ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΛΟΠΟΝΝΗΣΟΥ
University of the Peloponnese

Σχολή Οικονομίας και Τεχνολογίας
Τμήμα Πληροφορικής και Τηλεπικοινωνιών
Π.Μ.Σ. Επιστήμη Υπολογιστών

Μεταπτυχιακή Διπλωματική Εργασία

«Σχεδιασμός και Υλοποίηση Δυναμικού Συστήματος Διαχείρισης Κρατήσεων για Πολιτιστικούς Οργανισμούς»

Πεσλής Απόστολος

ΑΜ: 2022202402016

Επιβλέπων Καθηγητής: Βασιλάκης Κωνσταντίνος

Τρίπολη, 2026

ΠΕΡΙΛΗΨΗ

Η παρούσα πτυχιακή εργασία παρουσιάζει τον σχεδιασμό και την υλοποίηση της διαδικτυακής εφαρμογής, για ένα σύγχρονο σύστημα διαχείρισης κρατήσεων που απευθύνεται σε πολιτιστικούς οργανισμούς, όπως μουσεία και πολιτιστικά κέντρα. Στόχος της εφαρμογής είναι η αυτοματοποίηση και απλοποίηση των διαδικασιών που σχετίζονται με την ενοικίαση αιθουσών εκδηλώσεων και τον προγραμματισμό ομαδικών επισκέψεων (από σχολεία ή/και άλλους φορείς), μειώνοντας τη γραφειοκρατία και βελτιώνοντας την εμπειρία τόσο των διαχειριστών όσο και των επισκεπτών.

Η ανάπτυξη της εφαρμογής πραγματοποιήθηκε με βάση την αρχιτεκτονική Model - View - Controller (MVC), χωρίς τη χρήση πλαισίου ανάπτυξης (framework), προκειμένου να επιτευχθεί καθαρός, οργανωμένος και επεκτάσιμος κώδικας. Το τεχνολογικό υπόβαθρο περιλαμβάνει τη χρήση της PHP για τη λογική του backend, της MySQL για την αποθήκευση των δεδομένων και του Apache Server για την εξυπηρέτηση της εφαρμογής. Στο frontend χρησιμοποιήθηκε το έτοιμο πρότυπο Bootstrap v4 για τη διαμόρφωση του περιβάλλοντος χρήσης και η βιβλιοθήκη FullCalendar.js για την προβολή και διαχείριση του ημερολογίου κρατήσεων. Επιπλέον, ολόκληρο το περιβάλλον ανάπτυξης και εκτέλεσης είναι διαμορφωμένο σε περιέκτες (containerized) με χρήση του Docker, εξασφαλίζοντας τη φορητότητα, την ευκολία εγκατάστασης και τη συνέπεια στη λειτουργία και την παρακολούθηση μεταξύ διαφορετικών περιβαλλόντων.

Η εφαρμογή υποστηρίζει πολλαπλούς ρόλους χρηστών και παρέχει ολοκληρωμένες λειτουργίες για τη διαχείριση αιθουσών και (σχολικών) επισκέψεων. Περιλαμβάνει υποσύστημα καθορισμού κανόνων διαθεσιμότητας, με δυνατότητα ορισμού χρονικών περιόδων ισχύος και εξαιρέσεων, καθώς και μηχανισμό καταγραφής ενεργειών (logging) για την ενίσχυση της ασφάλειας και την προαγωγή της ιχνηλασιμότητας και της λογοδοσίας.

Το τελικό αποτέλεσμα είναι μια ολοκληρωμένη, ασφαλής και επεκτάσιμη πλατφόρμα που καλύπτει τις λειτουργικές ανάγκες πολιτιστικών οργανισμών, βελτιώνει την

οργάνωσή τους και προσφέρει στους επισκέπτες ευελιξία στον προγραμματισμό και τη διαχείριση των κρατήσεών τους.

Λέξεις Κλειδιά: Διαδικτυακή εφαρμογή, Διαχείριση κρατήσεων, Πολιτιστικοί οργανισμοί, Μουσεία, MVC αρχιτεκτονική, PHP, MySQL, Bootstrap, FullCalendar.js, Docker, Διαχείριση χρηστών, Διαθεσιμότητα, Καταγραφή ενεργειών, Ενοικίαση αιθουσών, Σχολικές επισκέψεις

ABSTRACT

This thesis presents the design and implementation of a web application for a modern reservation management system, aimed at cultural organizations, such as museums and cultural centers. The goal of the application is to automate and simplify processes related to the rental of event halls and the scheduling of visits (by schools or other organizations), reducing bureaucracy and improving the experience for both administrators and visitors.

The application was developed based on the Model–View–Controller (MVC) architecture, without the use of any development framework, in order to achieve clean, organized, and extensible code. The technological stack includes PHP for backend logic, MySQL for data storage, and Apache Server for serving the application. On the frontend, the Bootstrap v4 template was used to design the user interface, while the FullCalendar.js library enables the visualization and management of the reservation calendar. Furthermore, the entire development and execution environment is containerized using Docker, ensuring portability, easy installation, and consistency regarding operation and monitoring across different environments.

The application supports multiple user roles and provides comprehensive functionality for managing both event halls and (school) visits. It includes a subsystem for defining availability rules, allowing the configuration of validity periods and exceptions, as well as an activity logging mechanism to enhance security and promote traceability and accountability.

The result is a complete, secure, and extensible platform that meets the functional needs of cultural organizations, improves their organization, and offers visitors flexibility in scheduling and managing their reservations.

Keywords: Web application, Booking management, Cultural organizations, Museums, MVC architecture, PHP, MySQL, Bootstrap, FullCalendar.js, Docker, User management, Availability, Activity logging, Venues rental, School tours

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

ΠΕΡΙΛΗΨΗ *1*

ABSTRACT *3*

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ *4*

ΕΥΡΕΤΗΡΙΟ ΕΙΚΟΝΩΝ *9*

1 ΕΙΣΑΓΩΓΗ *15*

1.1 Παρουσίαση του Προβλήματος **15**

1.2 Δομή της Εργασίας **16**

2 ΑΝΑΛΥΣΗ ΑΠΑΙΤΗΣΕΩΝ *17*

2.1 Ανάλυση προβλήματος και αναγκών **17**

2.2 Λειτουργικές απαιτήσεις **19**

1. Διαχείριση Χρηστών και Αυθεντικοποίηση 19

2. Διαχείριση Αιθουσών 19

3. Διαχείριση Σχολικών Επισκέψεων 19

4. Διαχείριση Διαθεσιμότητας και Εξαιρέσεων 20

5. Σύστημα Κρατήσεων 20

6. Διαχείριση Συστήματος και Αναφορές 21

2.3 Μη λειτουργικές απαιτήσεις **21**

1. Ασφάλεια 21

2. Απόδοση 21

3. Ευχρηστία και Προσβασιμότητα 22

4. Συντήρηση και Επεκτασιμότητα 22

5. Φορητότητα 22

2.4 Ρόλοι χρηστών και δικαιώματα πρόσβασης **22**

2.4.1. Διαχειριστής Συστήματος (Administrator) 23

2. Υπεύθυνος Αιθουσών Εκδηλώσεων (Venue Manager) 23

3. Υπεύθυνος Επισκέψεων (Tour Manager) 23

4. Απλός Χρήστης (User) 23

2.5 Περιγραφή σεναρίων χρήσης (Use Cases) **24**

Σενάριο 1: Υποβολή Αιτήματος Κράτησης Αίθουσας 24

Σενάριο 2: Έγκριση Κράτησης από Διαχειριστή 25

Σενάριο 3: Μαζικός Ορισμός Αργιών (Εξαιρέσεις) 25

3. ΤΕΧΝΟΛΟΓΙΚΟ ΥΠΟΒΑΘΡΟ 27

3.1 Βάση δεδομένων MySQL 27

3.1.1 Ορισμός και χαρακτηριστικά 27

3.1.2 Το σχεσιακό μοντέλο 27

3.1.3 Μηχανή αποθήκευσης InnoDB και συναλλαγές 28

3.2 Γλώσσα προγραμματισμού PHP 28

3.2.1 Επισκόπηση και ιστορική αναδρομή 28

3.2.2 Λειτουργικά χαρακτηριστικά 29

3.2.3 Αντικειμενοστραφής προγραμματισμός (OOP) 29

3.2.4 Διασύνδεση με βάσεις δεδομένων και PDO 29

3.3 PHP Data Objects (PDO) 30

3.3.1 Ορισμός και ρόλος 30

3.3.2 Ασφάλεια και Prepared Statements 31

3.4 Αρχιτεκτονική Model – View – Controller (MVC) 31

3.4.1 Ορισμός και φιλοσοφία 31

3.4.2 Ανάλυση των μερών στην αρχιτεκτονική MVC 32

3.4.3 Οφέλη της αρχιτεκτονικής MVC για την εφαρμογή 33

3.4.4 Front Controller Pattern 33

3.5 Frontend τεχνολογίες 33

3.5.1 HTML5, CSS3 και JavaScript 33

3.5.2 Πλαίσιο Bootstrap 4 34

3.5.3 Βιβλιοθήκη FullCalendar.js 34

3.6 Docker και containerization 34

3.6.1 Ορισμός και σκοπός 34

3.6.2 Η έννοια των containers 35

3.6.3 Σύγκριση Docker με Εικονικές Μηχανές 35

3.6.4 Αρχιτεκτονική του Docker 38

3.6.5 Πώς λειτουργεί το Docker στον κύκλο ζωής μιας εφαρμογής 39

3.6.6 Εφαρμογή του Docker 40

4 ΣΧΕΔΙΑΣΗ ΤΗΣ ΕΦΑΡΜΟΓΗΣ 42

4.1 Αρχιτεκτονική συστήματος 42

4.2 Σχεδιασμός βάσης δεδομένων 42

1. Διαχείριση Χρηστών και Ρόλων 43

2. Διαχείριση Πόρων (Αίθουσες Εκδηλώσεων και Σχολικές Επισκέψεις) 44

3. Διαχείριση Διαθεσιμότητας και Εξαιρέσεων 44

4.3 Σχεδίαση διεπαφής χρήστη 45

4.4 Δομή των φακέλων και αρχείων 46

4.5 Ενδεικτικά παραδείγματα κώδικα 47

4.6 Δρομολόγηση και Σημείο Εισόδου 49

5 ΥΛΟΠΟΙΗΣΗ ΤΗΣ ΕΦΑΡΜΟΓΗΣ 51

5.1 Περιβάλλον ανάπτυξης 51

5.2 Υλοποίηση βασικών λειτουργιών 51

5.2.1 Αυθεντικοποίηση και Έλεγχος Πρόσβασης 51

5.2.2 Διαχείριση Προφίλ Χρήστη 56

5.2.3 Διαχείριση χρηστών και ρόλων 58

5.2.4 Δυναμική Διαχείριση Ρυθμίσεων και Περιεχομένου 62

5.2.5 Διαχείριση αιθουσών 64

5.2.6 Διαχείριση σχολικών επισκέψεων 66

5.2.7 Κανόνες διαθεσιμότητας και εξαιρέσεις 68

5.2.8 Προβολή και διαχείριση ημερολογίου 70

5.2.9 Σύστημα Στατιστικών και Αναφορών 75

5.3 Επικύρωση λειτουργικότητας και δοκιμές 76

1. Λειτουργικές Δοκιμές 76
2. Δοκιμές Διεπαφής (UI/UX Testing) 77
3. Δοκιμές Ασφαλείας (Security testing): 79

6 Οδηγίες εγκατάστασης και λειτουργίας 82

6.1 Περιγραφή δομής Docker Compose (containers, images, services) 82

Υπηρεσία 1: Web Server (web) 82

Υπηρεσία 2: Βάση Δεδομένων (db) 83

6.2 Διαδικασία εγκατάστασης της εφαρμογής 83

6.3 Εκτέλεση και πρόσβαση στην εφαρμογή 84

6.4 Προετοιμασία βάσης δεδομένων και αρχικών δεδομένων 84

6.5 Διαχείριση μέσω GitHub και ενημερώσεις 84

7 Παρουσίαση Λειτουργιών του Συστήματος 85

7.1 Επισκόπηση διεπαφής χρήστη 85

7.1.1 Αρχική Σελίδα (Landing Page) 85

7.1.2 Σελίδα Επικοινωνίας και Πληροφοριών 86

7.1.3 Παρακολούθηση Δραστηριότητας (Logs) 86

7.1.4 Πίνακας Ελέγχου και Στατιστικά (Dashboard) 87

7.1.5 Πίνακας Ελέγχου Χρήστη (User Dashboard) 88

7.2 Διαχείριση χρηστών και ρόλων ... **90**

7.3 Διαχείριση αιθουσών και σχολικών επισκέψεων ... **91**

Α. Προβολή διαχείρισης αιθουσών εκδηλώσεων ... 91

Β. Δημόσια προβολή αιθουσών εκδηλώσεων ... 92

Γ. Προβολή διαχείρισης επισκέψεων ... 93

Δ. Δημόσια προβολή επισκέψεων ... 94

7.4 Κανόνες διαθεσιμότητας και εξαιρέσεις ... **95**

7.5 Ημερολόγιο κρατήσεων ... **95**

8 ΕΠΙΛΟΓΟΣ ... ***98***

8.1 Συμπεράσματα ... **98**

8.2 Δυνατότητες Μελλοντικής Επέκτασης ... **99**

ΒΙΒΛΙΟΓΡΑΦΙΑ ... ***101***

ΠΑΡΑΡΤΗΜΑ Α': Docker Configuration Files ... ***104***

Α.1 Αρχείο Ρύθμισης Υπηρεσιών (compose.yml) ... **104**

Α.2 Αρχείο Δημιουργίας Εικόνας Web Server (Dockerfile) ... **105**

Α.3 Αρχείο Ρύθμισης Apache (apache.conf) ... **105**

ΠΑΡΑΡΤΗΜΑ Β': Σχεδίαση Βάσης Δεδομένων ... ***106***

ΠΑΡΑΡΤΗΜΑ Γ': Κώδικας της Εφαρμογής ... ***111***

Γ.1 Κεντρική Υποδομή και Routing ... **111**

Γ.2 Αυθεντικοποίηση ... **123**

Γ.3 Διαχείριση Χρηστών ... **132**

Γ.4 Διαχείριση Προφίλ ... **154**

Γ.5 Διαχείριση Αιθουσών (Venues) ... **160**

Γ.6 Διαχείριση Σχολικών Επισκέψεων (Tours) ... **190**

Γ.7 Σύστημα Κρατήσεων και API ... **214**

Γ.8 Καταγραφή Ενεργειών (Activity Logs) ... **268**

Γ.9 Διαχείριση Ρυθμίσεων (Settings) ... **273**

Γ.10 Δημόσιες Σελίδες και Στατιστικά ... **283**

ΠΑΡΑΡΤΗΜΑ Δ: Οδηγίες Εγκατάστασης και Εκτέλεσης ... ***294***

Δ.1 Προαπαιτούμενα Συστήματος 294

Δ.2 Λήψη και Προετοιμασία Αρχείων..... 294

Δ.3 Εκκίνηση της Εφαρμογής..... 295

Δ.4 Πρόσβαση στην Εφαρμογή 295

Δ.5 Στοιχεία Εισόδου 295

Δ.6 Τερματισμός Εφαρμογής 296

ΕΥΡΕΤΗΡΙΟ ΕΙΚΟΝΩΝ

Εικόνα 1. Σχεδιάγραμμα MVC 32

Εικόνα 2. Διαφορά Virtualization και Containerization 36

Εικόνα 3. Βασικές πτυχές σύγκρισης VMs – Containers 38

Εικόνα 4. Dockerfile 39

Εικόνα 5. Σχεδιάγραμμά Βάσης Δεδομένων MuseApp 43

Εικόνα 6. Δομή φακέλων εφαρμογής MuseApp 46

Εικόνα 7. Φόρμα εγγραφής 52

Εικόνα 8. Φόρμα επεξεργασίας προφίλ από χρήστη 57

Εικόνα 9. Φόρμα αλλαγής κωδικού πρόσβασης 58

Εικόνα 10. Στιγμιότυπο από τον πίνακα Users 58

Εικόνα 11. Στιγμιότυπο του πίνακα Χρηστών 59

Εικόνα 12. Φόρμα δημιουργίας χρήστη από τον διαχειριστή 60

Εικόνα 13. Επεξεργασία χρήστη από τον διαχειριστή 61

Εικόνα 14. Στιγμιότυπο πίνακα Logs 61

Εικόνα 15. Σελίδα διαχείρισης ρυθμίσεων ιστοσελίδας 63

Εικόνα 16. Σελίδα επικοινωνίας ιστοσελίδας 64

Εικόνα 17. Στιγμιότυπο από την επεξεργασία αίθουσας 65

Εικόνα 18. Φόρμα υποβολής νέου προγράμματος για σχολικές επισκέψεις 66

Εικόνα 19. Στιγμιότυπο σελίδας διαχείρισης εξαιρέσεων για σχολικές επισκέψεις ... 68

Εικόνα 20. Χρωματική κωδικοποίηση ημερολογίου 71

Εικόνα 21. Modal για προβολή ιστορικού, στο ημερολόγιο των διαχειριστών 72

Εικόνα 22. Στιγμιότυπο σελίδας στατιστικών 76

Εικόνα 23. Μήνυμα σφάλματος για διπλοεγγραφή κράτησης 76

Εικόνα 24. Προβολή νέου χρήστη 77

Εικόνα 25. Αποτελέσματα εγγραφής στον πίνακα Logs 77

Εικόνα 26. Σελίδα στατιστικών σε προβολή Monitor 77

Εικόνα 27. Σελίδα στατιστικών σε προβολή Tablet (1 από 2) 78

Εικόνα 28. Σελίδα στατιστικών σε προβολή Tablet (2 από 2) 78

Εικόνα 29. Σελίδα στατιστικών σε προβολή Κινητού (1 από 3) 78

Εικόνα 30. Σελίδα στατιστικών σε προβολή Κινητού (2 από 3) 79

Εικόνα 31. Σελίδα στατιστικών σε προβολή Κινητού (3 από 3) 79

Εικόνα 32. Σελίδα στατιστικών ενώ είναι συνδεδεμένος διαχειριστής 80

Εικόνα 33. Σελίδα στατιστικών ενώ είναι συνδεδεμένος απλός χρήστης 80

Εικόνα 34. Σελίδα στατιστικών ενώ δεν είναι συνδεδεμένος χρήστης 80

Εικόνα 35. Σελίδα στατιστικών ενώ είναι συνδεδεμένος διαχειριστής σχολικών επισκέψεων 80

Εικόνα 36. Προσπάθεια εισόδου στην φόρμα δημιουργίας κράτησης αίθουσας από διαχειριστή σχολικών επισκέψεων 81

Εικόνα 37. Αρχική σελίδα 85

Εικόνα 38. Σελίδα επικοινωνίας 86

Εικόνα 39. Σελίδα καταγραφής δραστηριότητας (Logs) 87

Εικόνα 40. Σελίδα στατιστικών 88

Εικόνα 41. Σελίδα επεξεργασίας προφίλ 89

Εικόνα 42. Σελίδα αλλαγής κωδικού πρόσβασης 89

Εικόνα 43. Σελίδα διαχείρισης χρηστών 91

Εικόνα 44. Σελίδα επεξεργασίας χρήστη 91

Εικόνα 45. Σελίδα διαχείρισης αιθουσών εκδηλώσεων 92

Εικόνα 46. Σελίδα επεξεργασίας αίθουσας εκδηλώσεων 92

Εικόνα 47. Σελίδα προβολής αιθουσών εκδηλώσεων 93

Εικόνα 48. Σελίδα προβολής λεπτομερειών αίθουσας εκδηλώσεων 93

Εικόνα 49. Σελίδα διαχείρισης προγραμμάτων σχολικών επισκέψεων 94

Εικόνα 50. Σελίδα επεξεργασίας προγράμματος σχολικών επισκέψεων 94

Εικόνα 51. Σελίδα προβολής επισκέψεων 94

Εικόνα 52. Σελίδα διαχείρισης εξαιρέσεων σχολικών επισκέψεων 95

Εικόνα 53. Σελίδα ημερολογίου σχολικών επισκέψεων 96

Εικόνα 54. Υποβολή αιτήματος κράτησης αίθουσας εκδηλώσεων 96

Εικόνα 55. Αποτελέσματα αιτήματος κράτησης αίθουσας εκδηλώσεων 97

Εικόνα 1. Σχεδιάγραμμα MVC 32

Εικόνα 2. Διαφορά Virtualization και Containerization 36

Εικόνα 3. Βασικές πτυχές σύγκρισης VMs – Containers 38

Εικόνα 4. Dockerfile 39

Εικόνα 5. Σχεδιάγραμμά Βάσης Δεδομένων MuseApp 43

Εικόνα 6. Δομή φακέλων εφαρμογής MuseApp 46

Εικόνα 7. Φόρμα εγγραφής 52

Εικόνα 8. Φόρμα επεξεργασίας προφίλ από χρήστη 57

Εικόνα 9. Φόρμα αλλαγής κωδικού πρόσβασης 58

Εικόνα 10. Στιγμιότυπο από τον πίνακα Users 58

Εικόνα 11. Στιγμιότυπο του πίνακα Χρηστών 59

Εικόνα 12. Φόρμα δημιουργίας χρήστη από τον διαχειριστή 60

Εικόνα 13. Επεξεργασία χρήστη από τον διαχειριστή 61

Εικόνα 14. Στιγμιότυπο πίνακα Logs 61

Εικόνα 15. Σελίδα διαχείρισης ρυθμίσεων ιστοσελίδας 63

Εικόνα 16. Σελίδα επικοινωνίας ιστοσελίδας 64

Εικόνα 17. Στιγμιότυπο από την επεξεργασία αίθουσας 65

Εικόνα 18. Φόρμα υποβολής νέου προγράμματος για σχολικές επισκέψεις 66

Εικόνα 19. Στιγμιότυπο σελίδας διαχείρισης εξαιρέσεων για σχολικές επισκέψεις ... 68

Εικόνα 20. Χρωματική κωδικοποίηση ημερολογίου 71

Εικόνα 21. Modal για προβολή ιστορικού, στο ημερολόγιο των διαχειριστών 72

Εικόνα 22. Στιγμιότυπο σελίδας στατιστικών 76

Εικόνα 23. Μήνυμα σφάλματος για διπλοεγγραφή κράτησης 76

Εικόνα 24. Προβολή νέου χρήστη 77

Εικόνα 25. Αποτελέσματα εγγραφής στον πίνακα Logs 77

Εικόνα 26. Σελίδα στατιστικών σε προβολή Monitor 77

Εικόνα 27. Σελίδα στατιστικών σε προβολή Tablet (1 από 2) 78

Εικόνα 28. Σελίδα στατιστικών σε προβολή Tablet (2 από 2) 78

Εικόνα 29. Σελίδα στατιστικών σε προβολή Κινητού (1 από 3) 78

Εικόνα 30. Σελίδα στατιστικών σε προβολή Κινητού (2 από 3) 79

Εικόνα 31. Σελίδα στατιστικών σε προβολή Κινητού (3 από 3) 79

Εικόνα 32. Σελίδα στατιστικών ενώ είναι συνδεδεμένος διαχειριστής 80

Εικόνα 33. Σελίδα στατιστικών ενώ είναι συνδεδεμένος απλός χρήστης 80

Εικόνα 34. Σελίδα στατιστικών ενώ δεν είναι συνδεδεμένος χρήστης 80

Εικόνα 35. Σελίδα στατιστικών ενώ είναι συνδεδεμένος διαχειριστής σχολικών επισκέψεων 80

Εικόνα 36. Προσπάθεια εισόδου στην φόρμα δημιουργίας κράτησης αίθουσας από διαχειριστή σχολικών επισκέψεων 81

Εικόνα 37. Αρχική σελίδα 85

Εικόνα 38. Σελίδα επικοινωνίας 86

Εικόνα 39. Σελίδα καταγραφής δραστηριότητας (Logs) 87

Εικόνα 40. Σελίδα στατιστικών 88

Εικόνα 41. Σελίδα επεξεργασίας προφίλ 89

Εικόνα 42. Σελίδα αλλαγής κωδικού πρόσβασης 89

Εικόνα 43. Σελίδα διαχείρισης χρηστών 91

Εικόνα 44. Σελίδα επεξεργασίας χρήστη 91

Εικόνα 45. Σελίδα διαχείρισης αιθουσών εκδηλώσεων 92

Εικόνα 46. Σελίδα επεξεργασίας αίθουσας εκδηλώσεων 92

Εικόνα 47. Σελίδα προβολής αιθουσών εκδηλώσεων 93

Εικόνα 48. Σελίδα προβολής λεπτομερειών αίθουσας εκδηλώσεων.....93

Εικόνα 49. Σελίδα διαχείρισης προγραμμάτων σχολικών επισκέψεων.....94

Εικόνα 50. Σελίδα επεξεργασίας προγράμματος σχολικών επισκέψεων 94

Εικόνα 51. Σελίδα προβολής επισκέψεων 94

Εικόνα 52. Σελίδα διαχείρισης εξαιρέσεων σχολικών επισκέψεων 95

Εικόνα 53. Σελίδα ημερολογίου σχολικών επισκέψεων.....96

Εικόνα 54. Υποβολή αιτήματος κράτησης αίθουσας εκδηλώσεων.....96

Εικόνα 55. Αποτελέσματα αιτήματος κράτησης αίθουσας εκδηλώσεων.....97

1 ΕΙΣΑΓΩΓΗ

Η ραγδαία εξέλιξη των διαδικτυακών τεχνολογιών έχει μετασχηματίσει ριζικά τον τρόπο με τον οποίο οργανισμοί και επιχειρήσεις αλληλοεπιδρούν με το κοινό τους. Στον τομέα του πολιτισμού, η ψηφιακή μετάβαση αποτελεί πλέον αναγκαιότητα και όχι πολυτέλεια. Η παρούσα διπλωματική εργασία πραγματεύεται τον σχεδιασμό και την ανάπτυξη μιας σύγχρονης διαδικτυακής εφαρμογής για τη διαχείριση κρατήσεων σε πολιτιστικούς οργανισμούς.

1.1 Παρουσίαση του Προβλήματος

Πολλοί πολιτιστικοί φορείς, όπως μικρά και μεσαία μουσεία ή πολιτιστικά κέντρα, εξακολουθούν να βασίζονται σε παραδοσιακές μεθόδους για τη διαχείριση των πόρων τους. Η διαδικασία κράτησης μιας αίθουσας για εκδήλωση ή ο προγραμματισμός μιας σχολικής επίσκεψης γίνεται συχνά τηλεφωνικά, με την καταγραφή να πραγματοποιείται σε χειρόγραφα ημερολόγια ή σε αρχεία υπολογιστή που τηρούνται διάσπαρτα και αποσυνδεδεμένα. Αυτή η προσέγγιση οδηγεί συχνά σε διοικητικό φόρτο, σφάλματα στη θεώρηση της διαθεσιμότητας, διπλοεγγραφές και αδυναμία παροχής άμεσης εξυπηρέτησης στους ενδιαφερόμενους εκτός ωραρίου λειτουργίας.

Σκοπός της παρούσας εργασίας είναι η δημιουργία ενός ολοκληρωμένου Πληροφοριακού Συστήματος που θα αυτοματοποιεί αυτές τις διαδικασίες. Η εφαρμογή στοχεύει να προσφέρει τα εξής. Αρχικά στους Διαχειριστές του συστήματος, ένα κεντρικό σημείο ελέγχου για τη διαχείριση αιθουσών, σχολικών επισκέψεων και κρατήσεων αυτών, με δυνατότητα εξαγωγής στατιστικών στοιχείων. Παράλληλα το κοινό θα αλληλοεπιδρά με ένα φιλικό περιβάλλον για την ενημέρωση διαθεσιμότητας σε πραγματικό χρόνο και την εύκολη υποβολή αιτημάτων κράτησης.

Η υλοποίηση βασίστηκε σε τεχνολογίες ανοιχτού κώδικα, ακολουθώντας την αρχιτεκτονική Model - View - Controller (MVC) για τη διασφάλιση της επεκτασιμότητας του κώδικα, ενώ για την ευκολία εγκατάστασης και φορητότητας της εφαρμογής χρησιμοποιήθηκε η τεχνολογία Docker.

1.2 Δομή της Εργασίας

Η παρούσα πτυχιακή εργασία διαρθρώνεται σε οκτώ κεφάλαια, τα οποία καλύπτουν τις φάσεις του κύκλο ζωής ανάπτυξης λογισμικού από την ανάλυση έως την υλοποίηση και την παρουσίαση.

Στο Κεφάλαιο 2 (Ανάλυση Απαιτήσεων), αναλύεται το πρόβλημα, καθορίζονται οι λειτουργικές και μη λειτουργικές απαιτήσεις του συστήματος και περιγράφονται οι ρόλοι των χρηστών και τα βασικά σενάρια χρήσης.

Στο Κεφάλαιο 3 (Τεχνολογικό Υπόβαθρο), παρουσιάζονται οι τεχνολογίες και τα εργαλεία που χρησιμοποιήθηκαν για την ανάπτυξη, όπως η γλώσσα PHP, η βάση δεδομένων MySQL, το πρότυπο MVC και η πλατφόρμα Docker.

Στο Κεφάλαιο 4 (Σχεδίαση της Εφαρμογής), περιγράφεται η αρχιτεκτονική του συστήματος, ο σχεδιασμός της βάσης δεδομένων και η δομή των αρχείων και φακέλων του κώδικα.

Στο Κεφάλαιο 5 (Υλοποίηση της Εφαρμογής), αναλύεται διεξοδικά η διαδικασία συγγραφής του κώδικα για τα βασικά υποσυστήματα (Χρήστες, Αίθουσες, Κρατήσεις, API) και εξηγούνται οι τεχνικές επιλογές που έγιναν.

Στο Κεφάλαιο 6 (Οδηγίες Εγκατάστασης και Λειτουργίας), παρέχονται αναλυτικές οδηγίες για την εγκατάσταση του περιβάλλοντος εκτέλεσης μέσω Docker.

Στο Κεφάλαιο 7 (Παρουσίαση Λειτουργιών του Συστήματος), παρουσιάζεται το τελικό αποτέλεσμα μέσω οθονών της εφαρμογής, με περιγραφή της αλληλεπίδρασης του χρήστη.

Τέλος, στο Κεφάλαιο 8 (Επίλογος), συνοψίζονται τα συμπεράσματα της εργασίας και προτείνονται κατευθύνσεις για μελλοντική επέκταση της εφαρμογής.

2 ΑΝΑΛΥΣΗ ΑΠΑΙΤΗΣΕΩΝ

Το στάδιο της ανάλυσης απαιτήσεων αποτελεί τον θεμέλιο λίθο για την επιτυχή ανάπτυξη οποιουδήποτε πληροφοριακού συστήματος, καθώς μεταφράζει τις επιχειρησιακές ανάγκες σε τεχνικές προδιαγραφές. Στο παρόν κεφάλαιο διενεργείται μια διεξοδική μελέτη του πλαισίου λειτουργίας της εφαρμογής MuseApp.

Αρχικά, περιγράφεται η υφιστάμενη κατάσταση και εντοπίζονται τα προβλήματα που καθιστούν αναγκαία τη δημιουργία του νέου συστήματος. Στη συνέχεια, καθορίζονται αναλυτικά οι λειτουργικές απαιτήσεις, δηλαδή τι πρέπει να κάνει το σύστημα, καθώς και οι μη λειτουργικές απαιτήσεις, τα ποιοτικά δηλαδή χαρακτηριστικά όπως η ασφάλεια και η απόδοση και το επιθυμητό επίπεδο του καθενός, όπου είναι απαραίτητο. Τέλος, προσδιορίζονται οι ρόλοι των χρηστών και αναλύονται τα βασικά σενάρια χρήσης, τα οποία θα αποτελέσουν τον οδηγό για τον σχεδιασμό και την υλοποίηση που ακολουθούν στα επόμενα κεφάλαια.

2.1 Ανάλυση προβλήματος και αναγκών

Η αποτελεσματική διαχείριση των πόρων και ο προγραμματισμός των δραστηριοτήτων αποτελούν κρίσιμους παράγοντες για την εύρυθμη λειτουργία των σύγχρονων πολιτιστικών οργανισμών, όπως τα μουσεία και τα πολιτιστικά κέντρα. Παρά την ψηφιακή μετάβαση σε πολλούς τομείς, παρατηρείται συχνά ότι οι διαδικασίες που αφορούν τις κρατήσεις αιθουσών για εκδηλώσεις και τον προγραμματισμό σχολικών επισκέψεων εξακολουθούν να βασίζονται σε παραδοσιακές, μη αυτοματοποιημένες μεθόδους.

Στην υφιστάμενη κατάσταση, η διαχείριση των κρατήσεων πραγματοποιείται κυρίως μέσω τηλεφωνικής επικοινωνίας ή ανταλλαγής μηνυμάτων ηλεκτρονικού ταχυδρομείου. Η καταγραφή των αιτημάτων γίνεται συχνά σε χειρόγραφα ημερολόγια ή σε απλά λογιστικά φύλλα, τα οποία δεν είναι προσβάσιμα ταυτόχρονα από όλα τα εμπλεκόμενα μέλη του προσωπικού. Αυτή η πρακτική οδηγεί σε σημαντικά λειτουργικά προβλήματα όπως παρακάτω.

Αρχικά παρατηρούνται ανθρώπινα λάθη και διπλοεγγραφές. Η έλλειψη ενός κεντρικού συστήματος ελέγχου διαθεσιμότητας σε πραγματικό χρόνο αυξάνει τον κίνδυνο

διπλοεγγραφών, όπου ο ίδιος πόρος δεσμεύεται από διαφορετικούς υπαλλήλους για την ίδια χρονική στιγμή.

Παράλληλα αυξάνεται ο διοικητικός φόρτος. Το προσωπικό αναλώνει σημαντικό χρόνο στη διαχείριση τηλεφωνικών κλήσεων και στην επιβεβαίωση στοιχείων, αντί να εστιάζει στο ουσιαστικό πολιτιστικό έργο του οργανισμού.

Ένα σημαντικό λειτουργικό πρόβλημα είναι η περιορισμένη προσβασιμότητα. Οι ενδιαφερόμενοι (σχολεία, εταιρείες, ιδιώτες) μπορούν να ενημερωθούν για τη διαθεσιμότητα και να πραγματοποιήσουν κράτηση μόνο κατά τις εργάσιμες ώρες του αρμόδιου προσωπικού, γεγονός που περιορίζει την εξυπηρέτηση του κοινού.

Μειονέκτημα αποτελεί και απουσία στατιστικών δεδομένων. Η έλλειψη ψηφιοποιημένων και δομημένων δεδομένων καθιστά δύσκολη την εξαγωγή συμπερασμάτων σχετικά με την επισκεψιμότητα, τη δημοφιλία των αιθουσών ή την εποχικότητα των σχολικών εκδρομών. Έτσι ο οργανισμός στερείται χρήσιμων δεδομένων τα οποία θα μπορούσαν να αξιοποιηθούν για τον καλύτερο σχεδιασμό και ανάπτυξή του.

Για την αντιμετώπιση των ανωτέρω προβλημάτων, κρίθηκε αναγκαία η σχεδίαση και υλοποίηση ενός ολοκληρωμένου διαδικτυακού συστήματος διαχείρισης κρατήσεων. Οι βασικές ανάγκες που καλείται να καλύψει το σύστημα είναι οι ακόλουθες.

1. Αρχικά θα υλοποιηθεί μια ενιαία βάση δεδομένων που θα αποτελεί το μοναδικό σημείο αναφοράς για τη διαθεσιμότητα των πόρων.
2. Επιπλέον θα επιτευχθεί η αυτοματοποίηση των διαδικασιών. Οι χρήστες θα έχουν τη δυνατότητα να ελέγχουν τη διαθεσιμότητα των αιθουσών και να υποβάλλουν αιτήματα κράτησης ηλεκτρονικά, μειώνοντας την ανάγκη τηλεφωνικής μεσολάβησης.
3. Επιπλέον θα παρέχεται η δυνατότητα στο διαχειριστή να ορίζει δυναμικά το ωράριο, τις αργίες και τις εξαιρέσεις διαθεσιμότητας, με διαισθητικό και εύληπτο τρόπο.
4. Τέλος, θα παρέχεται έλεγχος πρόσβασης. Τα δικαιώματα πρόσβασης θα διαβαθμίζονται ανάλογα με τον ρόλο του χρήστη, προκειμένου να παρέχονται εγγυήσεις για την ασφάλεια και την ακεραιότητα των δεδομένων.

2.2 Λειτουργικές απαιτήσεις

Οι λειτουργικές απαιτήσεις καθορίζουν τις συγκεκριμένες υπηρεσίες και λειτουργίες που οφείλει να παρέχει το σύστημα στους χρήστες του. Με βάση την ανάλυση των αναγκών, η εφαρμογή προδιαγράφεται να καλύπτει τις εξής λειτουργίες.

1. Διαχείριση Χρηστών και Αυθεντικοποίηση

- *Εγγραφή και Είσοδος*: Το σύστημα πρέπει να επιτρέπει τη δημιουργία λογαριασμού για απλούς χρήστες (εκπροσώπους σχολείων ή εταιρειών/ιδιωτών) και την ασφαλή είσοδό τους με χρήση κωδικού πρόσβασης.
- *Διαχείριση Προφίλ*: Οι χρήστες πρέπει να έχουν τη δυνατότητα επεξεργασίας των προσωπικών τους στοιχείων και αλλαγής του κωδικού πρόσβασης.
- *Ανάκτηση Κωδικού*: Θα πρέπει να υπάρχει διαδικασία (μέσω διαχειριστή) για την ανάκτηση πρόσβασης σε περίπτωση απώλειας κωδικού.
- *Ρόλοι Χρηστών*: Το σύστημα πρέπει να υποστηρίζει διακριτούς ρόλους (administrator, venue manager, tour manager, user) με διαφορετικά επίπεδα πρόσβασης.
- *Διαχείριση λογαριασμών χρηστών:* Ο διαχειριστής θα πρέπει να μπορεί να επιθεωρεί, να τροποποιεί και να απενεργοποιεί τους λογαριασμούς.

2. Διαχείριση Αιθουσών

- *Διαχείριση Οντοτήτων*: Οι διαχειριστές πρέπει να μπορούν να δημιουργούν, να επεξεργάζονται και να απενεργοποιούν αίθουσες, ορίζοντας χαρακτηριστικά όπως όνομα, χωρητικότητα, τιμή και περιγραφή.
- *Πολυμέσα*: Υποστήριξη μεταφόρτωσης πολλαπλών φωτογραφιών για κάθε αίθουσα και ορισμός κύριας φωτογραφίας (φωτογραφία εξωφύλλου).
- *Ωράριο Λειτουργίας*: Δυνατότητα ορισμού των ημερών λειτουργίας και της περιόδου ισχύος του ωραρίου για κάθε αίθουσα.

3. Διαχείριση Σχολικών Επισκέψεων

- *Χρονικές Ζώνες*: Το σύστημα πρέπει να επιτρέπει τη δημιουργία προγραμμάτων επισκέψεων με συγκεκριμένη ώρα έναρξης και λήξης.

- Εποχικότητα: Καθώς ο οργανισμός μπορεί να προσφέρει εκπαιδευτικά προγράμματα, το καθένα από τα οποία έχει ορισμένη ημερομηνία έναρξης και λήξης, θα πρέπει να είναι δυνατό να καθορίζονται τα χρονικά πλαίσια των εκπαιδευτικών προγραμμάτων και τα δυνατά χρονοπρογράμματα επίσκεψης.

4. Διαχείριση Διαθεσιμότητας και Εξαιρέσεων

- *Διαχείριση Εξαιρέσεων*: Οι διαχειριστές πρέπει να μπορούν να ορίζουν ημέρες αργίας ή συντήρησης, κατά τις οποίες οι πόροι δεν θα είναι διαθέσιμοι, είτε μαζικά είτε μεμονωμένα.
- *Έλεγχος σε Πραγματικό Χρόνο*: Το σύστημα θα παρέχει τη λειτουργία ελέγχου της διαθεσιμότητας των πόρων πριν από κάθε κράτηση, λαμβάνοντας υπόψη τις υφιστάμενες κρατήσεις, το ωράριο και τις εξαιρέσεις.

5. Σύστημα Κρατήσεων

- *Ημερολόγιο*: Προβολή διαθεσιμότητας μέσω διαδραστικού ημερολογίου, με χρωματική κωδικοποίηση για την κατάσταση κάθε ημέρας (Διαθέσιμη, Μερικώς Κλεισμένη, Πλήρης).
- *Υποβολή Αιτήματος*: Οι πιστοποιημένοι χρήστες πρέπει να μπορούν να υποβάλλουν αιτήματα κράτησης.
- *Ροή Έγκρισης*: Τα αιτήματα για αίθουσες πρέπει να τίθενται σε κατάσταση αναμονής (Pending) μέχρι να εγκριθούν ή να απορριφθούν από τον αρμόδιο διαχειριστή.
- *Κρατήσεις εκ μέρους τρίτων*: Οι διαχειριστές πρέπει να έχουν τη δυνατότητα να καταχωρούν κρατήσεις εκ μέρους πελατών που δεν έχουν πρόσβαση στο σύστημα (τηλεφωνικές κρατήσεις).
- *Περιορισμοί*: Εφαρμογή κανόνων, όπως η απαγόρευση κράτησης από απλούς χρήστες για ημερομηνίες πολύ κοντινές στην τρέχουσα. Το σύστημα πρέπει να διασφαλίζει ότι το προσωπικό του οργανισμού έχει επαρκή χρόνο για την προετοιμασία των χώρων για κρατήσεις αιθουσών και αντίστοιχα την προετοιμασία για την υποδοχή επισκέψεων. Ως εκ τούτου, οι χρήστες θα μπορούν να υποβάλλουν αιτήματα μόνο για ημερομηνίες που απέχουν τουλάχιστον έξι (6) ημέρες από την τρέχουσα ημερομηνία υποβολής.

6. Διαχείριση Συστήματος και Αναφορές

- *Ρυθμίσεις*: Δυνατότητα δυναμικής αλλαγής των στοιχείων επικοινωνίας και των ωραρίων του οργανισμού που εμφανίζονται στη δημόσια σελίδα.
- *Καταγραφή Ενεργειών (Logging)*: Καταγραφή όλων των κρίσιμων ενεργειών (δημιουργία, επεξεργασία, διαγραφή, Login/Logout) για λόγους ασφαλείας.
- *Στατιστικά*: Παροχή γραφημάτων και αναφορών σχετικά με το πλήθος των κρατήσεων ανά μήνα και τη δημοτικότητα των πόρων.

2.3 Μη λειτουργικές απαιτήσεις

Οι μη λειτουργικές απαιτήσεις καθορίζουν τα ποιοτικά χαρακτηριστικά του συστήματος και τους περιορισμούς υπό τους οποίους πρέπει να λειτουργεί. Για την ανάπτυξη της εφαρμογής, τέθηκαν οι εξής προδιαγραφές:

1. Ασφάλεια

- *Έλεγχος Πρόσβασης*: Πραγματοποιείται έλεγχος δικαιωμάτων σε κάθε Controller, ώστε να διασφαλίζεται ότι ο κάθε χρήστης έχει πρόσβαση μόνο στις λειτουργίες για τις οποίες είναι κατάλληλα εξουσιοδοτημένος. Ειδικότερα, οι απλοί χρήστες δεν θα πρέπει να έχουν πρόσβαση σε σελίδες διαχείρισης.
- *Προστασία Κωδικών*: Αποθήκευση κωδικών πρόσβασης αποκλειστικά σε κρυπτογραφημένη μορφή (Hashed) με χρήση του ισχυρού αλγορίθμου BCrypt.
- *Εμπιστευτικότητα και ακεραιότητα Δεδομένων*: Το σύστημα πρέπει να εγγυάται την εμπιστευτικότητα και την ακεραιότητα των δεδομένων. Προς αυτή την κατεύθυνση χρησιμοποιήθηκαν Prepared Statements σε όλα τα ερωτήματα SQL για την πλήρη εξάλειψη κινδύνων SQL Injection και ακολουθήθηκαν καλές προγραμματιστικές πρακτικές για την αποφυγή σφαλμάτων που διακυβεύουν την ασφάλεια σε εφαρμογές διαδικτύου [1].

2. Απόδοση

Προκειμένου να υποστηριχθεί καλή απόδοση του συστήματος, υιοθετήθηκαν οι ακόλουθες πρακτικές:

- *Ασύγχρονη Επικοινωνία*: Χρήση τεχνολογίας AJAX για τη φόρτωση των δεδομένων στο ημερολόγιο, ώστε να ελαχιστοποιείται ο όγκος μεταφοράς δεδομένων και να μην απαιτείται πλήρης επαναφόρτωση της σελίδας.
- *Βελτιστοποίηση Βάσης Δεδομένων*: Δημιουργία κατάλληλων ευρετηρίων και χρήση ξένων κλειδιών (Foreign Keys) για τη γρήγορη ανάκτηση δεδομένων και τη διατήρηση της αναφορικής ακεραιότητας.

3. Ευχρηστία και Προσβασιμότητα

- *Responsive Design*: Η διεπαφή χρήστη σχεδιάστηκε με το πλαίσιο Bootstrap 4, διασφαλίζοντας ότι η εφαρμογή είναι πλήρως λειτουργική και καλαίσθητη σε όλες τις συσκευές (Desktops, Tablets, Smartphones).
- *Φιλικότητα*: Παροχή άμεσης ανατροφοδότησης στον χρήστη μέσω ενημερωτικών μηνυμάτων (Flash Messages) για την επιτυχία ή αποτυχία κάθε ενέργειας.

4. Συντήρηση και Επεκτασιμότητα

- *Αρχιτεκτονική MVC*: Ο διαχωρισμός του κώδικα σε Μοντέλα, Ελεγκτές και Προβολές, σύμφωνα με το μοντέλο MVC (Model, View, Controller) επιτρέπει την εύκολη συντήρηση και επέκταση της εφαρμογής χωρίς να επηρεάζονται άλλα τμήματα του συστήματος.
- *Clean Code*: Τήρηση κανόνων καθαρού κώδικα και σχολιασμός των συναρτήσεων/μεθόδων.

5. Φορητότητα

- *Containerization*: Η εφαρμογή και η βάση δεδομένων είναι πλήρως ενθυλακωμένες σε περιέκτες (containers) Docker, επιτρέποντας την άμεση εγκατάσταση και εκτέλεση σε οποιοδήποτε λειτουργικό σύστημα υποστηρίζει Docker, χωρίς την ανάγκη πολύπλοκων ρυθμίσεων περιβάλλοντος.

2.4 Ρόλοι χρηστών και δικαιώματα πρόσβασης

Το σύστημα υποστηρίζει τέσσερις (4) διακριτούς ρόλους χρηστών, καθένας από τους οποίους έχει πρόσβαση σε συγκεκριμένο υποσύνολο λειτουργιών. Η υλοποίηση βασίζεται στο μοντέλο Ελέγχου Πρόσβασης Βάσει Ρόλων. Οι ρόλοι που έχουν

δημιουργηθεί είναι (α) Διαχειριστής Συστήματος (Administrator), (β) Υπεύθυνος Αιθουσών Εκδηλώσεων (Venue Manager), (γ) Υπεύθυνος Επισκέψεων (Tour Manager) και (δ) Απλός Χρήστης (User). Οι λειτουργικότητες που μπορεί να εκτελέσει ο κάθε ρόλος, περιγράφονται στις ακόλουθες παραγράφους.

2.4.1. Διαχειριστής Συστήματος (Administrator)

- Είναι ο χρήστης με τα υψηλότερα δικαιώματα (Super Admin).
- Έχει πλήρη πρόσβαση σε όλα τα υποσυστήματα.
- Διαχειρίζεται τους λογαριασμούς χρηστών (δημιουργία, επεξεργασία, απενεργοποίηση, επαναφορά κωδικών).
- Έχει πρόσβαση στα αρχεία καταγραφής (logs) ασφαλείας και στις γενικές ρυθμίσεις της εφαρμογής.
- Μπορεί να εκτελέσει χρέη υπεύθυνου αιθουσών εκδηλώσεων και υπευθύνου επισκέψεων, αν χρειαστεί.

2. Υπεύθυνος Αιθουσών Εκδηλώσεων (Venue Manager)

- Ο ρόλος αυτός εστιάζει στη διαχείριση των αιθουσών που ενοικιάζονται για εκδηλώσεις.
- Διαχειρίζεται τις αίθουσες (δημιουργία, επεξεργασία, φωτογραφίες).
- Ορίζει τις εξαιρέσεις διαθεσιμότητας για τις αίθουσες.
- Εγκρίνει ή απορρίπτει τα αιτήματα κράτησης αιθουσών.
- Καταχωρεί κρατήσεις εκ μέρους τρίτων.

3. Υπεύθυνος Επισκέψεων (Tour Manager)

- Ο ρόλος αυτός εστιάζει στις κρατήσεις για (σχολικές) επισκέψεις.
- Διαχειρίζεται τα προγράμματα επισκέψεων.
- Ορίζει τις εξαιρέσεις διαθεσιμότητας για τα προγράμματα.
- Επιβλέπει και διαχειρίζεται τις κρατήσεις σχολείων.

4. Απλός Χρήστης (User)

- Αποτελεί τον τελικό χρήστη της εφαρμογής (π.χ. εκπρόσωπος σχολείου ή εταιρείας).
- Μπορεί να βλέπει τη διαθεσιμότητα στο ημερολόγιο.

- Μπορεί να υποβάλει αιτήματα κράτησης.
- Έχει πρόσβαση στο ιστορικό και την διαχείριση των δικών του κρατήσεων.
- Μπορεί να επεξεργαστεί το προφίλ του.
- Δεν έχει πρόσβαση σε κανένα διαχειριστικό εργαλείο, ούτε σε στοιχεία άλλων χρηστών.

2.5 Περιγραφή σεναρίων χρήσης (Use Cases)

Για την καλύτερη κατανόηση της λειτουργικότητας του συστήματος, περιγράφονται παρακάτω τρία (3) αντιπροσωπευτικά σενάρια χρήσης που καλύπτουν τις κεντρικές διαδικασίες της εφαρμογής.

Σενάριο 1: Υποβολή Αιτήματος Κράτησης Αίθουσας

Actor:	Απλός Χρήστης (Εκπρόσωπος Εταιρείας)
Στόχος:	Η δέσμευση μιας αίθουσας για συγκεκριμένη ημερομηνία.
Προϋποθέσεις:	Ο χρήστης έχει συνδεθεί στο σύστημα και έχει συμπληρώσει τα στοιχεία της εταιρείας του στο προφίλ.
Ροή Γεγονότων:	1. Ο χρήστης πλοηγείται στη σελίδα «Ημερολόγιο Αιθουσών». 2. Το σύστημα εμφανίζει το ημερολόγιο με χρωματική κωδικοποίηση (Πράσινο για διαθέσιμες ημέρες, Κόκκινο για κατειλημμένες). 3. Ο χρήστης επιλέγει (κάνει κλικ) σε μια διαθέσιμη ημερομηνία. 4. Το σύστημα εμφανίζει ένα αναδυόμενο παράθυρο (Modal) και φορτώνει δυναμικά τις διαθέσιμες αίθουσες για τη συγκεκριμένη ημέρα. 5. Ο χρήστης επιλέγει την επιθυμητή αίθουσα από τη λίστα και πατάει «Υποβολή». 6. Το σύστημα ελέγχει ξανά τη διαθεσιμότητα (για αποφυγή ταυτόχρονης κράτησης), αποθηκεύει το αίτημα με κατάσταση «Σε Εκκρεμότητα» (Pending) και εμφανίζει μήνυμα επιτυχίας.

Αποτέλεσμα:	Ο διαχειριστής λαμβάνει ειδοποίηση για το νέο αίτημα και η ημερομηνία στο ημερολόγιο αλλάζει χρώμα για να δείξει ότι υπάρχει εκκρεμότητα.

Σενάριο 2: Έγκριση Κράτησης από Διαχειριστή

Actor:	Υπεύθυνος Αιθουσών (Venue Manager)
Στόχος:	Η αξιολόγηση και οριστικοποίηση ενός αιτήματος.
Προϋποθέσεις:	Υπάρχει τουλάχιστον ένα αίτημα σε κατάσταση «Pending».
Ροή Γεγονότων:	1. Ο διαχειριστής συνδέεται και βλέπει στο dashboard το πλήθος των εκκρεμών αιτημάτων. 2. Πλοηγείται στη σελίδα «Αιτήματα Κρατήσεων». 3. Το σύστημα εμφανίζει λίστα με τα αιτήματα, συμπεριλαμβανομένων των στοιχείων επικοινωνίας του αιτούντος και της ημερομηνίας. 4. Ο διαχειριστής εξετάζει το αίτημα και επιλέγει «Έγκριση». 5. Το σύστημα ενημερώνει την κατάσταση της κράτησης σε «Accepted» (Εγκεκριμένη).
Αποτέλεσμα:	Ο χρήστης λαμβάνει ειδοποίηση ότι η κράτησή του εγκρίθηκε. Η αίθουσα δεσμεύεται οριστικά για τη συγκεκριμένη ημερομηνία και πλέον δεν εμφανίζεται ως διαθέσιμη σε άλλους χρήστες.

Σενάριο 3: Μαζικός Ορισμός Αργιών (Εξαιρέσεις)

Actor:	Διαχειριστής (Administrator) ή Tour Manager
Στόχος:	Η ακύρωση διαθεσιμότητας λόγω εθνικής εορτής ή προγραμματισμένων εργασιών συντήρησης
Προϋποθέσεις:	Ο διαχειριστής ή ο tour manager έχει συνδεθεί στο σύστημα.
Ροή Γεγονότων:	1. Ο διαχειριστής πλοηγείται στη σελίδα «Εξαιρέσεις & Αργίες». 2. Επιλέγει την ημερομηνία της αργίας (π.χ. 25η Μαρτίου).

	3. Στο πεδίο επιλογής πόρων, επιλέγει το checkbox «Επιλογή Όλων» για να εφαρμόσει την αργία σε όλα τα προγράμματα επισκέψεων ταυτόχρονα. 4. Πληκτρολογεί την αιτιολογία (π.χ. «Εθνική Εορτή») και πατάει «Αποθήκευση».
Αποτέλεσμα:	Το σύστημα δημιουργεί εγγραφές εξαίρεσης για όλα τα προγράμματα. Στο δημόσιο ημερολόγιο, η συγκεκριμένη ημέρα εμφανίζεται πλέον ως «Μη Διαθέσιμη» για όλους τους χρήστες, αποτρέποντας μελλοντικές κρατήσεις.

3. ΤΕΧΝΟΛΟΓΙΚΟ ΥΠΟΒΑΘΡΟ

Η επιλογή των κατάλληλων εργαλείων και τεχνολογιών αποτελεί κρίσιμο παράγοντα για την επιτυχία ενός έργου λογισμικού. Στο παρόν κεφάλαιο παρουσιάζεται το τεχνολογικό οικοσύστημα πάνω στο οποίο αναπτύχθηκε η εφαρμογή. Αναλύονται τα χαρακτηριστικά της γλώσσας προγραμματισμού, της βάσης δεδομένων και των αρχιτεκτονικών προτύπων που υιοθετήθηκαν, τεκμηριώνοντας τις επιλογές που έγιναν.

3.1 Βάση δεδομένων MySQL

3.1.1 Ορισμός και χαρακτηριστικά

Η MySQL είναι το δημοφιλέστερο Σύστημα Διαχείρισης Σχεσιακών Βάσεων Δεδομένων ανοιχτού κώδικα στον κόσμο [2]. Αναπτύσσεται και υποστηρίζεται από την Oracle Corporation. Βασίζεται στη γλώσσα SQL (Structured Query Language), η οποία αποτελεί το διεθνές πρότυπο για την επικοινωνία με βάσεις δεδομένων.

Η MySQL διακρίνεται για την ταχύτητα, την αξιοπιστία και την ευκολία χρήσης της. Αποτελεί βασικό συστατικό της δημοφιλούς στοίβας τεχνολογιών LAMP (Linux, Apache, MySQL, PHP), πάνω στην οποία βασίζεται ένα μεγάλο μέρος του παγκόσμιου ιστού [3].

3.1.2 Το σχεσιακό μοντέλο

Ως Σύστημα Διαχείρισης Σχεσιακών Βάσεων Δεδομένων, η MySQL αποθηκεύει τα δεδομένα σε πίνακες (tables), οι οποίοι αποτελούνται από γραμμές (εγγραφές) και στήλες (πεδία). Η ισχύς του συστήματος έγκειται στη δυνατότητα σύνδεσης των πινάκων εγγραφών των πινάκων μεταξύ τους, μέσω εννοιολογικών συσχετίσεων (conceptual relationships).

Στην περίπτωση της εφαρμογής MuseApp, στο σχεσιακό μοντέλο αποτυπώθηκαν συσχετίσεις μεταξύ των οντοτήτων της εφαρμογής. Για παράδειγμα:

- Ένας χρήστης μπορεί να κάνει πολλές κρατήσεις, ενώ μία κράτηση συσχετίζεται με έναν μόνο χρήστη (Σχέση 1 - προς – πολλά μεταξύ των οντοτήτων *Χρήστης* και *Κράτηση*).

- Μια κράτηση αντιστοιχεί σε μία συγκεκριμένη Αίθουσα, ενώ μία αίθουσα μπορεί να συσχετίζεται με πολλές κρατήσεις (Σχέση πολλά-προς-1).

Η αποτύπωση αυτών των σχέσεων επιτυγχάνεται μέσω της χρήσης Πρωτευόντων Κλειδιών (Primary Keys) για τη μοναδική ταυτοποίηση κάθε εγγραφής και Ξένων Κλειδιών (Foreign Keys) για τη διασύνδεση των πινάκων. Με κατάλληλες δηλώσεις, διασφαλίζεται και η αναφορική ακεραιότητα [4].

3.1.3 Μηχανή αποθήκευσης InnoDB και συναλλαγές

Για την παρούσα διπλωματική εργασία, επιλέχθηκε η μηχανή αποθήκευσης InnoDB, η οποία είναι η προεπιλεγμένη μηχανή της MySQL. Το σημαντικότερο χαρακτηριστικό του InnoDB είναι η υποστήριξη δοσοληψιών (Transactions) που ακολουθούν το μοντέλο ACID (Atomicity, Consistency, Isolation, Durability) .

Η υποστήριξη δοσοληψιών είναι κρίσιμη για ένα σύστημα κρατήσεων. Εγγυάται ότι μια σύνθετη ενέργεια (π.χ. έλεγχος διαθεσιμότητας και αντίστοιχη εισαγωγή κράτησης) είτε θα ολοκληρωθεί είτε πλήρως και επιτυχώς, είτε δεν θα εκτελεστεί καθόλου (rollback) σε περίπτωση σφάλματος. Αυτό αποτρέπει την αλλοίωση των δεδομένων και το φαινόμενο των συνθηκών ανταγωνισμού (race conditions) που οδηγούν σε ασυνεπείς καταστάσεις της βάσης δεδομένων, π.χ. σε διπλοεγγραφές [5].

3.2 Γλώσσα προγραμματισμού PHP

3.2.1 Επισκόπηση και ιστορική αναδρομή

Η PHP είναι μια γλώσσα προγραμματισμού ανοιχτού κώδικα, η οποία εκτελείται στην πλευρά του διακομιστή (server-side scripting language) και είναι ειδικά σχεδιασμένη για την ανάπτυξη δυναμικών διαδικτυακών εφαρμογών. Δημιουργήθηκε αρχικά το 1994 από τον Rasmus Lerdorf και έκτοτε έχει εξελιχθεί σε μία από τις δημοφιλέστερες γλώσσες για τον Παγκόσμιο Ιστό, τροφοδοτώντας ένα σημαντικό ποσοστό των ιστοσελίδων παγκοσμίως [6].

Σε αντίθεση με τις γλώσσες που εκτελούνται στην πλευρά του πελάτη (client-side), όπως η JavaScript, ο κώδικας της PHP εκτελείται στον διακομιστή. Το αποτέλεσμα της εκτέλεσης είναι η παραγωγή κώδικα HTML, ο οποίος αποστέλλεται στον φυλλομετρητή (browser) του χρήστη. Αυτό σημαίνει ότι ο τελικός χρήστης βλέπει μόνο

το αποτέλεσμα και όχι τον πηγαίο κώδικα που παρήγαγε τη σελίδα, γεγονός που προσφέρει ένα επίπεδο ασφάλειας στην επιχειρησιακή λογική της εφαρμογής.

3.2.2 Λειτουργικά χαρακτηριστικά

Η PHP διακρίνεται για την ευελιξία και την ευκολία ενσωμάτωσής της με άλλες τεχνολογίες. Τα κυριότερα χαρακτηριστικά της είναι τα εξής.

- Ενσωμάτωση σε HTML: Ο κώδικας PHP μπορεί να ενσωματωθεί απευθείας μέσα σε αρχεία HTML, διευκολύνοντας τη δυναμική παραγωγή περιεχομένου.
- Πολλαπλές Πλατφόρμες (Cross-platform): Λειτουργεί αποτελεσματικά στα περισσότερα λειτουργικά συστήματα (Linux, Windows, macOS) και συνεργάζεται με τους δημοφιλέστερους διακομιστές ιστού (Apache, Nginx).
- Διερμηνευόμενη Γλώσσα (Interpreted): Δεν απαιτεί μεταγλώττιση (compilation) σε εκτελέσιμο αρχείο, καθώς ο κώδικας μεταφράζεται και εκτελείται σε πραγματικό χρόνο από τη μηχανή της PHP [5].

3.2.3 Αντικειμενοστραφής προγραμματισμός (OOP)

Αν και η PHP ξεκίνησε ως μια απλή διαδικαστική γλώσσα, οι σύγχρονες εκδόσεις της (PHP 7 και 8) προσφέρουν πλήρη υποστήριξη του Αντικειμενοστραφούς Προγραμματισμού (Object-Oriented Programming - OOP). Αυτό το χαρακτηριστικό ήταν καθοριστικό για την ανάπτυξη της εφαρμογής MuseApp, καθώς επέτρεψε:

- Τη δημιουργία Κλάσεων (Classes) για τις οντότητες του συστήματος (π.χ. User, Venue, Booking).
- Την εφαρμογή της αρχιτεκτονικής MVC (Model-View-Controller), διαχωρίζοντας τη λογική από την παρουσίαση.
- Τη χρήση προηγμένων εννοιών όπως η κληρονομικότητα, η ενθυλάκωση (encapsulation) και οι διεπαφές (interfaces), που καθιστούν τον κώδικα πιο δομημένο και συντηρήσιμο.

3.2.4 Διασύνδεση με βάσεις δεδομένων και PDO

Η PHP παρέχει εγγενή υποστήριξη για ένα ευρύ φάσμα συστημάτων διαχείρισης βάσεων δεδομένων. Στην παρούσα εργασία, για την επικοινωνία με τη βάση MySQL, επιλέχθηκε η χρήση του προτύπου PHP Data Objects (PDO).

Το PDO αποτελεί ένα επίπεδο αφαίρεσης (abstraction layer) για την πρόσβαση σε δεδομένα. Η χρήση του κρίνεται απαραίτητη σε σύγχρονες εφαρμογές διότι προσφέρει σημαντικά πλεονεκτήματα.

- Αρχικά ενισχύει την ασφάλεια καθώς υποστηρίζει τη χρήση προετοιμασμένων δηλώσεων (Prepared Statements), οι οποίες διαχωρίζουν τα δεδομένα από το ερώτημα SQL, εξαλείφοντας τον κίνδυνο επιθέσεων τύπου SQL Injection.
- Επιπρόσθετα παρέχει ευελιξία αφού επιτρέπει την εύκολη αλλαγή του συστήματος βάσης δεδομένων (π.χ. από MySQL σε PostgreSQL) χωρίς να απαιτούνται αλλαγές στον κώδικα της εφαρμογής, παρά μόνο στη διαμόρφωση του περιβάλλοντος εκτέλεσης (παροχή της κατάλληλης *συμβολοσειράς σύνδεσης / connect string*).
- Τέλος διευκολύνει την διαχείριση των σφαλμάτων. Παρέχει ισχυρούς μηχανισμούς διαχείρισης εξαιρέσεων (Exceptions), επιτρέποντας στην εφαρμογή να αντιδρά σωστά σε περίπτωση αποτυχίας σύνδεσης ή σε σφάλματα εκτέλεσης ερωτήματος [7].

3.3 PHP Data Objects (PDO)

3.3.1 Ορισμός και ρόλος

Το PHP Data Objects (PDO) είναι μια επέκταση (βιβλιοθήκη) της γλώσσας PHP που ορίζει μια ελαφριά και συνεπή διεπαφή για την πρόσβαση σε βάσεις δεδομένων. Σε αντίθεση με παλαιότερες προσεγγίσεις (όπως η βιβλιοθήκη MySQL Native Driver[1] [8]), το PDO λειτουργεί ως ένα Επίπεδο Αφαίρεσης Πρόσβασης Δεδομένων [9].

Αυτό σημαίνει ότι, ανεξάρτητα από τη βάση δεδομένων που χρησιμοποιείται στο παρασκήνιο (MySQL, PostgreSQL, SQLite, Oracle), οι εντολές PHP για την εκτέλεση ερωτημάτων και την ανάκτηση δεδομένων παραμένουν ίδιες. Αυτό το χαρακτηριστικό προσδίδει σημαντική ευελιξία και φορητότητα στον κώδικα της εφαρμογής.

[1] https://www.php.net/manual/en/book.mysqlnd.php

3.3.2 Ασφάλεια και Prepared Statements

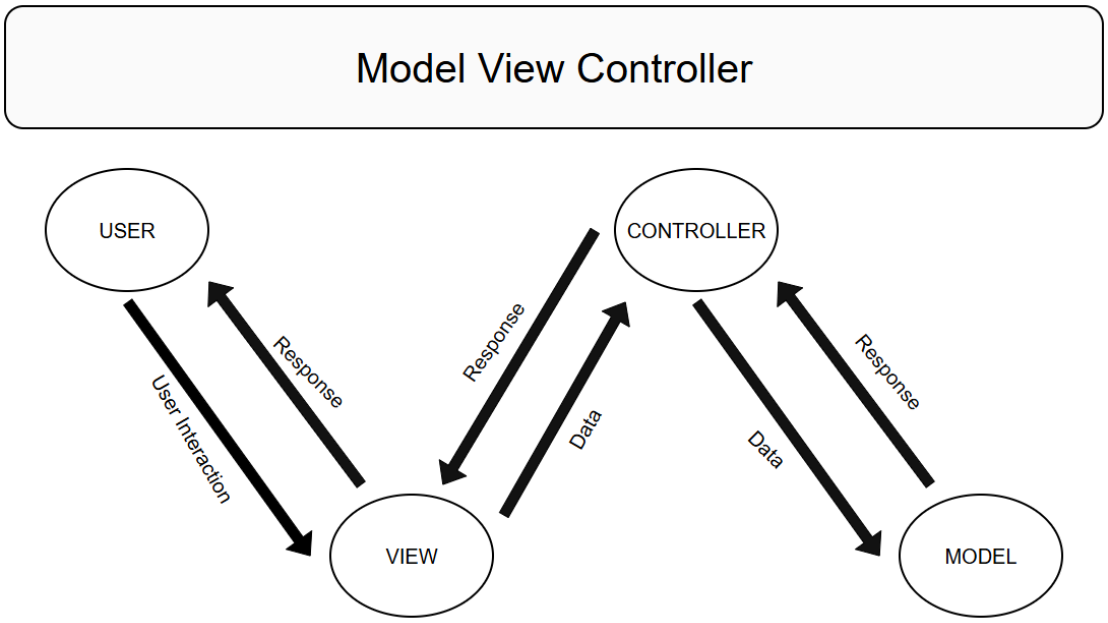
Ένας σημαντικός λόγος για την επιλογή του PDO για την ανάπτυξη του MuseApp είναι η ασφάλεια. Το PDO υποστηρίζει εγγενώς τις Προετοιμασμένες Δηλώσεις (Prepared Statements). Η βάση δεδομένων αναλύει, μεταγλωττίζει και βελτιστοποιεί το πλάνο εκτέλεσης του ερωτήματος SQL πριν εισαχθούν σε αυτό τα δεδομένα του χρήστη. Τα δεδομένα στέλνονται στη συνέχεια ξεχωριστά από το ερώτημα. Αυτή η διαδικασία εξουδετερώνει πλήρως τον κίνδυνο επιθέσεων τύπου SQL Injection, καθώς τα δεδομένα εισόδου αντιμετωπίζονται αυστηρά ως παράμετροι, χωρίς να υπάρχει κίνδυνος να παρερμηνευθούν ως εκτελέσιμος κώδικας SQL [10].

3.4 Αρχιτεκτονική Model – View – Controller (MVC)

3.4.1 Ορισμός και φιλοσοφία

Το Model-View-Controller (MVC) (Μοντέλο-Προβολή-Ελεγκτής) είναι ένα αρχιτεκτονικό πρότυπο λογισμικού που χρησιμοποιείται ευρέως στη μηχανική λογισμικού για την ανάπτυξη εφαρμογών που περιλαμβάνουν διεπαφές χρήστη. Η βασική φιλοσοφία του είναι ο Διαχωρισμός Αρμοδιοτήτων. Με βάση τη φιλοσοφία αυτή, η εφαρμογή διαχωρίζεται σε τρία αλληλένδετα συστατικά, επιτρέποντας την ανεξάρτητη ανάπτυξη, δοκιμή και συντήρηση της επιχειρησιακής λογικής, των δεδομένων και της παρουσίασης [11].

Η ροή της πληροφορίας σε μια διαδικτυακή εφαρμογή MVC ακολουθεί έναν συγκεκριμένο κύκλο: ο χρήστης αλληλοεπιδρά με τη διεπαφή, ο Ελεγκτής δέχεται το αίτημα, ενημερώνει το Μοντέλο και τελικά επιλέγει την κατάλληλη Προβολή για να εμφανίσει το αποτέλεσμα στον χρήστη. Η ροή αυτή αποτυπώνεται στην Εικόνα 1.



Εικόνα 1. Σχεδιάγραμμα MVC

3.4.2 Ανάλυση των μερών στην αρχιτεκτονική MVC

Όπως αναφέρθηκε προηγουμένως, σε μία αρχιτεκτονική MVC υπάρχουν τρεις συνιστώσες, το μοντέλο (model), η προβολή (view) και ο ελεγκτής (controller).Οι συνιστώσες αυτές αναλύονται συνοπτικά στη συνέχεια.

1. Μοντέλο (Model)

Το Μοντέλο αποτελεί τον πυρήνα της εφαρμογής. Διαχειρίζεται τα δεδομένα, την επιχειρησιακή λογική (Business Logic) και τους κανόνες του συστήματος. Είναι υπεύθυνο για την επικοινωνία με τη βάση δεδομένων (μέσω ερωτημάτων SQL) και την επιστροφή των αποτελεσμάτων.

2. Προβολή (View)

Η Προβολή είναι υπεύθυνη για την οπτική απεικόνιση των δεδομένων στον χρήστη. Δεν περιέχει λογική επεξεργασίας δεδομένων, παρά μόνο κώδικα παρουσίασης (HTML, CSS) και απλή λογική εμφάνισης (π.χ. βρόχους (loops) για λίστες). Η Προβολή λαμβάνει τα έτοιμα δεδομένα από τον Ελεγκτή και τα μορφοποιεί.

3. Ελεγκτής (Controller)

Ο Ελεγκτής λειτουργεί ως ο ενδιάμεσος μεταξύ του Μοντέλου και της Προβολής. Δέχεται τα αιτήματα του χρήστη (User Input), τα επεξεργάζεται, καλεί τις κατάλληλες μεθόδους του Μοντέλου και επιλέγει ποια Προβολή θα επιστραφεί ως απάντηση [11].

3.4.3 Οφέλη της αρχιτεκτονικής MVC για την εφαρμογή

Η υιοθέτηση του προτύπου MVC στην ανάπτυξη του MuseApp προσέφερε σημαντικά πλεονεκτήματα. Αρχικά, στην οργάνωση του κώδικα, επέτρεψε τον σαφή διαχωρισμό των αρχείων σε φακέλους (`/models`, `/views`, `/controllers`), καθιστώντας τον κώδικα ευανάγνωστο και εύκολα διαχειρίσιμο. Επιπλέον συνεισέφερε και στη βελτίωση της συντήρησης, καθώς αλλαγές στην εμφάνιση μπορούν να γίνουν χωρίς να επηρεαστεί η λογική της βάσης δεδομένων και αντίστροφα. Τέλος, παρείχε την δυνατότητα επαναχρησιμοποίησης του κώδικα. Το ίδιο Μοντέλο μπορεί να χρησιμοποιηθεί από διαφορετικούς Ελεγκτές, αποφεύγοντας την επανάληψη κώδικα.

3.4.4 Front Controller Pattern

Σε συνδυασμό με το MVC, η εφαρμογή χρησιμοποιεί το πρότυπο Front Controller. Όλα τα αιτήματα HTTP δρομολογούνται σε ένα μοναδικό αρχείο εισόδου (public/index.php), το οποίο αναλαμβάνει την αρχικοποίηση του περιβάλλοντος και τη δρομολόγηση (Routing) του αιτήματος στον κατάλληλο Controller. Αυτή η τεχνική εξασφαλίζει την κεντρική διαχείριση στον έλεγχο και ενισχύει την ασφάλεια της εφαρμογής.

3.5 Frontend τεχνολογίες

3.5.1 HTML5, CSS3 και JavaScript

Η υλοποίηση της διεπαφής χρήστη (Frontend) βασίστηκε στη συνδυαστική χρήση των τριών θεμελιωδών τεχνολογιών του παγκόσμιου ιστού, οι οποίες συνεργάζονται αρμονικά για να προσφέρουν μια ολοκληρωμένη και λειτουργική εμπειρία πλοήγησης.

Συγκεκριμένα, η HTML αποτέλεσε τον σκελετό της εφαρμογής, παρέχοντας την απαραίτητη σημασιολογική δομή στο περιεχόμενο. Πάνω σε αυτή τη δομή εφαρμόστηκε το CSS, το οποίο ανέλαβε τον ρόλο της μορφοποίησης και της διαμόρφωσης της αισθητικής της παρουσίασης, επιτυγχάνοντας τον διαχωρισμό του

περιεχομένου από την εμφάνιση. Τέλος, η JavaScript προσέδωσε την απαραίτητη δυναμική συμπεριφορά στο σύστημα, επιτρέποντας την αλληλεπίδραση του χρήστη σε πραγματικό χρόνο, τη διαχείριση συμβάντων όπως η εμφάνιση αναδυόμενων παραθύρων (Modals), καθώς και την ασύγχρονη επικοινωνία με τον εξυπηρετητή χωρίς την ανάγκη για επαναφόρτωση της σελίδας.

3.5.2 Πλαίσιο Bootstrap 4

Για να διασφαλιστεί η ταχύτητα ανάπτυξης και η προσαρμοστικότητα της εφαρμογής σε διαφορετικές συσκευές (Responsive Design), επιλέχθηκε το πλαίσιο Bootstrap 4. Το Bootstrap παρέχει ένα σύστημα πλέγματος (Grid System) 12 στηλών, το οποίο επιτρέπει τη δημιουργία ευέλικτων διατάξεων. Επιπλέον, προσφέρει έτοιμα, καλοσχεδιασμένα συστατικά όπως μπάρες πλοήγησης, κάρτες, κουμπιά και φόρμες, διασφαλίζοντας μια συνεπή και επαγγελματική εμφάνιση σε όλη την εφαρμογή [12].

3.5.3 Βιβλιοθήκη FullCalendar.js

Για την κεντρική για την εφαρμογή λειτουργία της προβολής και διαχείρισης των κρατήσεων, ενσωματώθηκε η βιβλιοθήκη JavaScript FullCalendar. Πρόκειται για μια ισχυρή βιβλιοθήκη που επιτρέπει την εμφάνιση συμβάντων σε μορφή ημερολογίου (Μήνας/Εβδομάδα/Ημέρα). Στο MuseApp, το FullCalendar ρυθμίστηκε ώστε να αντλεί δεδομένα μέσω AJAX από το API της εφαρμογής (ApiController). Η βιβλιοθήκη προσφέρει πλούσια διαδραστικότητα, επιτρέποντας στον χρήστη να κάνει κλικ σε ημερομηνίες για να δει λεπτομέρειες ή να ξεκινήσει μια νέα κράτηση, χωρίς να απαιτείται επαναφόρτωση της σελίδας [13].

3.6 Docker και containerization

3.6.1 Ορισμός και σκοπός

Το Docker είναι μία πλατφόρμα ανοιχτού κώδικα που επιτρέπει την αυτοματοποιημένη δημιουργία, ανάπτυξη και εκτέλεση εφαρμογών μέσα σε απομονωμένα περιβάλλοντα, γνωστά ως *περιέκτες* (containers). Τα containers αποτελούν ελαφριά, φορητά και απομονωμένα συστήματα που περιλαμβάνουν όλα τα απαραίτητα στοιχεία για την εκτέλεση μίας εφαρμογής, όπως ο πηγαίος κώδικας, βιβλιοθήκες, εξαρτήσεις και παραμετροποιήσεις.

Η κύρια αποστολή του Docker είναι να προσφέρει έναν τρόπο για τη συσκευασία (Packaging) και διανομή λογισμικού, διασφαλίζοντας ότι η εφαρμογή θα εκτελείται με τον ίδιο τρόπο ανεξαρτήτως περιβάλλοντος. Αυτή η προσέγγιση λύνει ένα από τα πιο συχνά προβλήματα στην ανάπτυξη λογισμικού, γνωστό ως “it works on my machine” φαινόμενο [14].

Το Docker δημιουργήθηκε από τον Solomon Hykes και παρουσιάστηκε για πρώτη φορά το 2013 ως εσωτερικό εργαλείο της εταιρείας dotCloud. Από τότε έχει εξελιχθεί σε μία από τις βασικότερες τεχνολογίες στον χώρο του DevOps, του cloud computing και της σύγχρονης αρχιτεκτονικής λογισμικού [15].

3.6.2 Η έννοια των containers

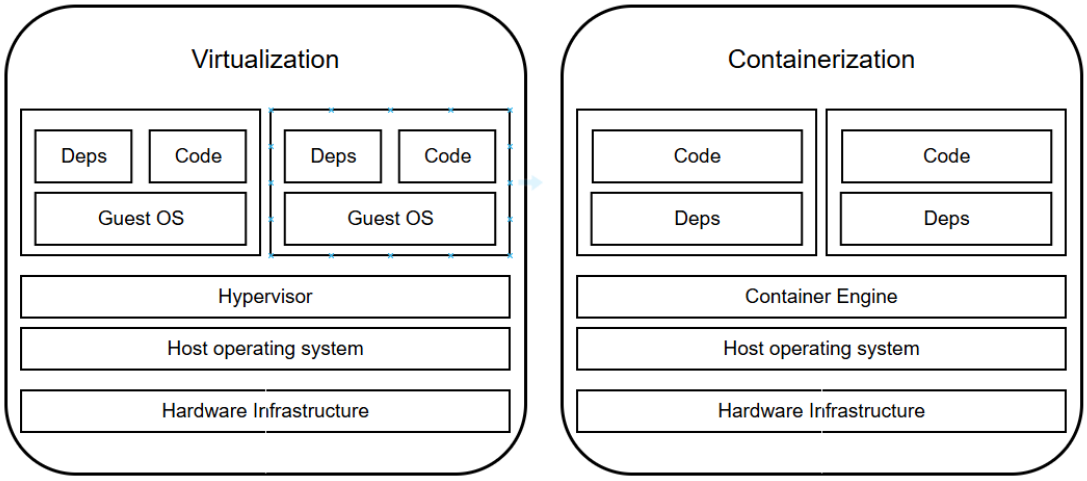
Τα containers είναι απομονωμένα περιβάλλοντα εκτέλεσης εφαρμογών, τα οποία χρησιμοποιούν τον πυρήνα του λειτουργικού συστήματος του host (κοινή χρήση του kernel), αλλά διατηρούν πλήρη απομόνωση σε επίπεδο αρχείων συστήματος, βιβλιοθηκών και διεργασιών. Αυτό τα καθιστά σημαντικά πιο ελαφριά και γρήγορα σε σύγκριση με τις εικονικές μηχανές. Η δημιουργία και εκκίνηση ενός container μπορεί να γίνει μέσα σε δευτερόλεπτα, καθώς δεν απαιτείται φόρτωση πλήρους λειτουργικού συστήματος.

Η λειτουργία των containers βασίζεται σε τεχνολογίες του Linux όπως τα namespaces και τα *cgroups*, οι οποίες επιτρέπουν την απομόνωση πόρων και διεργασιών, ενώ διατηρείται η αποτελεσματική χρήση του hardware [16].

3.6.3 Σύγκριση Docker με Εικονικές Μηχανές

Παραδοσιακά, η ανάπτυξη εφαρμογών σε διαφορετικά περιβάλλοντα γινόταν με τη χρήση εικονικών μηχανών (VMs). Οι VMs προσφέρουν πλήρη απομόνωση μέσω της εξομοίωσης ενός ολόκληρου λειτουργικού συστήματος, συνοδεύονται ωστόσο από υψηλό υπολογιστικό κόστος, καθυστερήσεις στην εκκίνηση, και δυσκολίες στη διανομή.

Ο παρακάτω πίνακας συνοψίζει τις βασικές διαφορές:



Εικόνα 2. Διαφορά Virtualization και Containerization

Στα αριστερά, όπου παρουσιάζεται η εικονικοποίηση δύο εφαρμογών σε έναν υπολογιστή με την χρήση των εικονικών μηχανών, παρατηρούμε ότι κάθε μία από τις εικονικές μηχανές απαιτεί ξεχωριστή εγκατάσταση λειτουργικού συστήματος (Guest OS). Αντιθέτως, στην αρχιτεκτονική χρησιμοποιεί το Docker (απεικονίζεται στα δεξιά), παρατηρείται ότι κάθε container που περιέχει μία εφαρμογή χρειάζεται μόνο τις εξαρτήσεις και τον κώδικά της.

Το Docker προσφέρει μία πιο αποδοτική προσέγγιση, ειδικά για εφαρμογές μικροϋπηρεσιών (microservices) και σύγχρονα DevOps pipelines.

Εμβαθύνοντας τη συγκριτική ανάλυση μεταξύ των Εικονικών Μηχανών και των Docker Containers μπορούμε να εστιάσουμε στις εξής πτυχές: την απομόνωση (Isolation), το μέγεθος/επιβάρυνση (Size/Overhead), τη φορητότητα (Portability) και τις ενδεικνυόμενες περιπτώσεις χρήσης (when to use). Οι Εικονικές Μηχανές προσφέρουν ισχυρή απομόνωση, καθώς κάθε VM διαθέτει το δικό της λειτουργικό σύστημα, αλλά έχουν μεγαλύτερο μέγεθος λόγω της ανάγκης για πλήρως virtualized hardware και λειτουργικό σύστημα του φιλοξενούμενου host. Αντιθέτως, τα Docker Containers παρέχουν απομόνωση σε επίπεδο διεργασιών, μοιράζονται τον πυρήνα του host OS, και είναι πιο ελαφριά, με ελάχιστη επιβάρυνση μεγέθους. Επιπλέον, τα Containers διακρίνονται για την υψηλή τους φορητότητα, καθώς είναι ανεξάρτητα από την πλατφόρμα και εκτελούνται με συνέπεια σε διαφορετικά περιβάλλοντα, ενώ οι VMs είναι λιγότερο φορητές λόγω της εξάρτησής τους από συγκεκριμένους hypervisors και διαμορφώσεις του λειτουργικού συστήματος. Όσον αφορά τις περιπτώσεις χρήσης, οι VMs συνιστώνται για εφαρμογές που απαιτούν ισχυρή

απομόνωση ή όταν αντιμετωπίζονται παλαιότερες εφαρμογές που δεν μπορούν εύκολα να διαμορφωθούν σε περιβάλλον περιεκτών, καθώς και για περιβάλλοντα δοκιμών ή ανάπτυξης που απαιτούν πλήρη συστήματα. Αντιθέτως, τα Containers προτείνονται για σύγχρονες, cloud-native εφαρμογές μικροϋπηρεσιών (microservices), όπου η γρήγορη κλιμάκωση και η αποδοτικότητα αποτελούν προτεραιότητα. Η παρακάτω εικόνα αποτελεί ένα χρήσιμο οδηγό για την κατανόηση των διαφορών και των κατάλληλων εφαρμογών των δύο τεχνολογιών στο πλαίσιο της διαχείρισης υποδομών πληροφορικής [17].

Χαρακτηριστικό	Εικονικές Μηχανές (VMs)	Docker Containers
Απομόνωση	**Ισχυρή απομόνωση:** Κάθε VM έχει το δικό του Λειτουργικό Σύστημα (OS), παρέχοντας πλήρη απομόνωση.	**Απομόνωση επιπέδου διεργασίας:** Τα containers μοιράζονται τον πυρήνα του κεντρικού Λειτουργικού Συστήματος (host OS).
Μέγεθος / Επιβάρυνση	**Μεγαλύτερο:** Τα VMs έχουν μεγαλύτερο αποτύπωμα λόγω του λειτουργικού συστήματος (guest OS) και του εικονικού υλικού.	**Ελαφρύ:** Τα containers έχουν ελάχιστη επιβάρυνση (overhead), καθώς μοιράζονται τον ίδιο πυρήνα.
Φορητότητα	**Λιγότερο φορητές:** Τα VMs μπορεί να δεσμεύονται σε συγκεκριμένους hypervisors και διαμορφώσεις του λειτουργικού συστήματος.	**Εξαιρετικά φορητά:** Τα containers είναι ανεξάρτητα από πλατφόρμες (platform-agnostic) και εκτελούνται με συνέπεια.
Ενδεικτικές περιπτώσεις χρήσης	Είναι δόκιμο να χρησιμοποιηθούν containers όταν 1. Χρειάζεται ισχυρή απομόνωση μεταξύ	Δημιουργείτε σύγχρονες, cloud-native εφαρμογές χρησιμοποιώντας αρχιτεκτονική μικροϋπηρεσιών (microservices).Πρέπει να

	διαφορετικών περιβαλλόντων. 2. Πρέπει να ενσωματωθούν εφαρμογές «παλαιού τύπου» (legacy), οι οποίες ενδέχεται να μην μετατρέπονται εύκολα σε containers. 3. Απαιτείται η αναπαραγωγή ενός πλήρους περιβάλλον συστήματος για δοκιμές ή ανάπτυξη.	κλιμακώσετε τις εφαρμογές σας γρήγορα και αποτελεσματικά. Η φορητότητα σε διαφορετικά περιβάλλοντα αποτελεί κορυφαία προτεραιότητα.

Εικόνα 3. Βασικές πτυχές σύγκρισης VMs – Containers

3.6.4 Αρχιτεκτονική του Docker

Η αρχιτεκτονική του Docker βασίζεται στα εξής βασικά συστατικά:

A. Docker Engine

Πρόκειται για τη βασική μηχανή που επιτρέπει τη δημιουργία και διαχείριση containers. Αποτελείται από:

- Docker Deamon (υπηρεσία Docker): Υλοποιεί τις εντολές και διαχειρίζεται images, containers, δίκτυα και volumes.
- Docker Client: Δέχεται εντολές από τον χρήστη (π.χ. `docker build`, `docker run`) και τις προωθεί στην υπηρεσία του Docker (Docker daemon).
- REST API: Επιτρέπει την επικοινωνία μεταξύ client και daemon.

B. Docker Image

Είναι ένα στιγμιότυπο μόνο για ανάγνωση (read-only) που περιλαμβάνει όλα τα απαραίτητα για την εκτέλεση της εφαρμογής. Τα images είναι πολύ-επίπεδα, γεγονός που επιτρέπει την επαναχρησιμοποίηση και αποδοτική αποθήκευση.

Γ. Docker Container

Είναι ένα εκτελέσιμο στιγμιότυπο ενός Docker image. Ο container (περιέκτης) δημιουργείται από το image και εκτελείται ως απομονωμένη οντότητα με δικό του σύστημα αρχείων, δίκτυο και πόρους.

Δ. Dockerfile

Αποτελεί αρχείο οδηγιών για τη δημιουργία ενός image. Προδιαγράφει τα βήματα που πρέπει να ακολουθηθούν βήμα ώστε να εγκατασταθεί ή να ρυθμιστεί (π.χ. ποια έκδοση Node.js, ποιος φάκελος να αντιγραφεί σε ποιο σημείο στο σύστημα αρχείων του container κ.λπ.).

Στην Εικόνα 4 παρουσιάζεται ένα Dockerfile το οποίο χρησιμοποιεί ως βασική εικόνα, από το Docker Hub, ένα image που εκτός από τη βασική έκδοση του Linux περιλαμβάνει την έκδοση 8.2 της PHP και έναν Apache web server που έχει ρυθμιστεί να εκτελεί εφαρμογές PHP και επιπλέον εκτελεί μία εντολή κατά την διαδικασία δημιουργίας του image, μέσω της οποίας εγκαθιστά δύο επεκτάσεις, τις pdo και pdo_mysql, έτσι ώστε να είναι δυνατή η λειτουργικότητα για διεπαφή με βάσεις δεδομένων και ειδικότερα της MySQL.

```
1   FROM php:8.2-apache
2
3   RUN docker-php-ext-install pdo pdo_mysql
```

Εικόνα 4. Dockerfile

Ε. Docker Hub / Registry

Αποτελεί τον «χώρο αποθήκευσης» των images. Το Docker Hub είναι η επίσημη δημόσια registry, αλλά μπορούν να δημιουργηθούν και ιδιωτικά registries για επιχειρησιακή χρήση [18].

3.6.5 Πώς λειτουργεί το Docker στον κύκλο ζωής μιας εφαρμογής

Το Docker ενσωματώνεται ιδανικά στον κύκλο ζωής ανάπτυξης λογισμικού, από την ανάπτυξη (development) έως την παραγωγή (production), υποστηρίζοντας μοντέρνες πρακτικές όπως το Continuous Integration/Continuous Deployment (CI/CD).

Οι προγραμματιστές μπορούν να δημιουργούν και να πραγματοποιούν ελέγχους λειτουργίας εφαρμογών τοπικά σε containers, διασφαλίζοντας ότι το περιβάλλον εκτέλεσης είναι ακριβώς το ίδιο με αυτό που θα χρησιμοποιηθεί στην παραγωγή. Αυτό μειώνει τα σφάλματα, διευκολύνει την ομαδική εργασία και επιταχύνει τον κύκλο ανάπτυξης.

Επιπλέον, το Docker υποστηρίζει τη φιλοσοφία “Infrastructure as Code” (IaC), μέσω αρχείων όπως το `Dockerfile` και το `docker-compose.yml`, καθιστώντας τον καθορισμό και την επανάληψη περιβαλλόντων εύκολο και διαφανή [19].

3.6.6 Εφαρμογή του Docker

Το Docker, ως μία από τις πιο δημοφιλείς τεχνολογίες περιεκτών, έχει προσελκύσει μεγάλες εταιρείες όπως η Google και η RedHat. Η μέθοδος απομόνωσης sandbox[2] προσφέρει μεγαλύτερη ασφάλεια στις εφαρμογές του Docker, επιτρέποντας το containerization και τη μεταφόρτωση της εφαρμογής και του περιβάλλοντος στον διακομιστή. Οι διαχειριστές χρειάζεται απλώς να κατεβάσουν την εικόνα (image) και να εκκινήσουν το κοντέινερ για να την αναπτύξουν.

Α. Υψηλή Αποδοτικότητα και Αξιοποίηση Πόρων

Στην παραδοσιακή προσέγγιση, τα πληροφοριακά κέντρα, όπως αυτά των εταιρειών και των πανεπιστημίων, χρησιμοποιούν διακομιστές ή εικονικές μηχανές (VMs) για την εκτέλεση εφαρμογών. Αυτή η διαδικασία απαιτεί την αρχική εγκατάσταση λειτουργικού συστήματος και στη συνέχεια την ανάπτυξη των απαιτούμενων υπηρεσιών, γεγονός που καταναλώνει χρόνο και ανθρώπινους πόρους, μειώνοντας έτσι την αποδοτικότητα.

Το Docker διαθέτει μηχανισμό απομόνωσης πόρων παρόμοιο με αυτόν των VM, όπου κάθε container μπορεί να διαθέτει επεξεργαστές, μνήμη, δίκτυο και αποθήκευση ανεξάρτητα. Ωστόσο, οι τεχνολογίες Linux Containers (LxC)[3] [20], containerd[4] [21]

---

[2] Το sandbox είναι ένα απομονωμένο και ασφαλές περιβάλλον που χρησιμοποιείται για δοκιμές, πειραματισμούς ή εκτέλεση λογισμικού χωρίς να επηρεάζει τα συστήματα παραγωγής ή να έχει τη δυνατότητα να βλάψει τον κεντρικό υπολογιστή.
[3] https://linuxcontainers.org/
[4] https://containerd.io/

και runc[5] [22] που χρησιμοποιούνται από το Docker, του επιτρέπουν να χρησιμοποιεί πόρους χωρίς να απαιτείται η πλήρης του υλικού, μειώνοντας έτσι την επιβάρυνση για τη διαχείριση των περιεκτών. Έτσι, το Docker είναι πιο αποδοτικό και εξοικονομεί περισσότερους πόρους σε σχέση με τις παραδοσιακές εικονικές μηχανές.

Β. Κοινή Χρήση Εικόνας Λειτουργικού Συστήματος και Γρήγορη Εκκίνηση

Το Docker επιτρέπει την κοινή χρήση εικόνων του λειτουργικού συστήματος, αποφεύγοντας την ανάγκη πολλαπλών εγκαταστάσεων, όπως συμβαίνει με τις εικονικές μηχανές. Αυτό έχει ως αποτέλεσμα ταχύτερη εκκίνηση, μικρότερη κατανάλωση χώρου και ευκολότερη συντήρηση.

Γ. Εύκολη Ανάπτυξη και Μεταφορά Υπηρεσιών

Στην παραδοσιακή ανάπτυξη εφαρμογών σε ένα πληροφοριακό κέντρο, η ανάπτυξη ή η μεταφορά μιας υπηρεσίας απαιτεί την εγκατάσταση λειτουργικού συστήματος, λογισμικού περιβάλλοντος και ρυθμίσεων, διαδικασία που είναι περίπλοκη και μπορεί να οδηγήσει σε προβλήματα συμβατότητας.

Το Docker λύνει το παραπάνω πρόβλημα, καθώς απλοποιεί τη διαδικασία ανάπτυξης και μεταφοράς υπηρεσιών. Οι προγραμματιστές μπορούν να δημιουργήσουν ένα container με το απαραίτητο περιβάλλον ανάπτυξης και να το αποθηκεύσουν ως εικόνα (image) στην αποθήκη εικόνων (image repository). Όταν χρειάζεται ανάπτυξη ή μεταφορά μιας υπηρεσίας, απλώς γίνεται φόρτωση της αντίστοιχης εικόνας, χωρίς την ανάγκη πολύπλοκης εγκατάστασης και διαμόρφωσης [23].

[5] https://www.docker.com/blog/runc/

4 ΣΧΕΔΙΑΣΗ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

Ο σχεδιασμός προηγείται της υλοποίησης και η ποιότητά του είναι καθοριστική για την ποιότητα του τελικού λογισμικού. Σε αυτό το κεφάλαιο περιγράφεται η αρχιτεκτονική δομή του συστήματος MuseApp. Παρουσιάζεται το διάγραμμα του σχήματος της βάσης δεδομένων, η οργάνωση των αρχείων σύμφωνα με το πρότυπο MVC και ο σχεδιασμός της διεπαφής χρήστη, που στοχεύει στη βέλτιστη εμπειρία πλοήγησης.

4.1 Αρχιτεκτονική συστήματος

Η αρχιτεκτονική του συστήματος MuseApp βασίστηκε στο πρότυπο σχεδίασης Model - View - Controller (MVC). Η επιλογή αυτή έγινε για να διαχωριστεί η λογική της εφαρμογής από την παρουσίαση των δεδομένων, επιτρέποντας ευκολότερη συντήρηση και μελλοντική επέκταση.

Η ροή της πληροφορίας στο σύστημα ακολουθεί τα εξής βήματα:

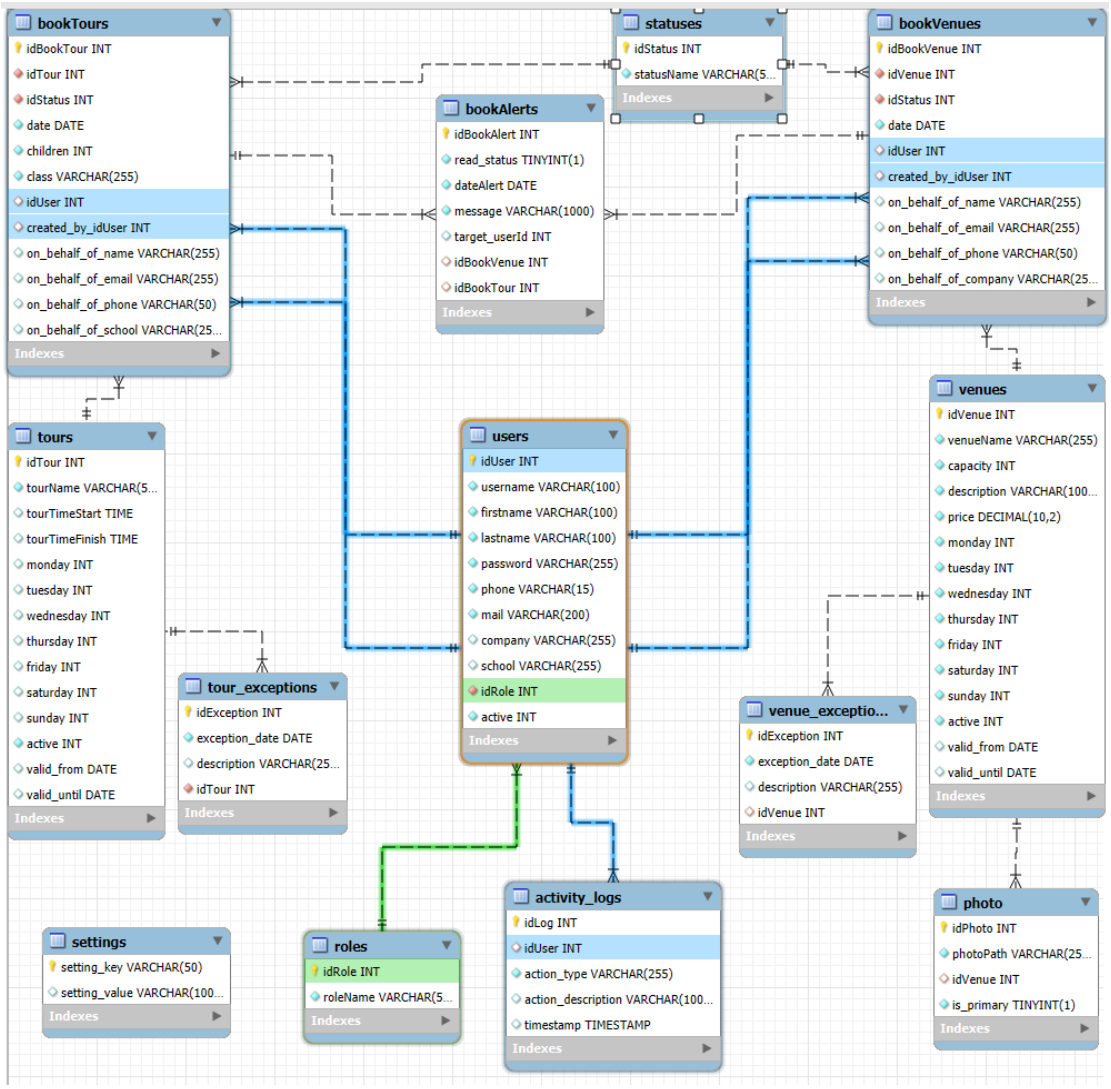
- Controller (Ελεγκτής): Δέχεται τα αιτήματα του χρήστη (HTTP Requests) μέσω της συνιστώσας του Router, επεξεργάζεται την είσοδο και καλεί τις κατάλληλες μεθόδους των Μοντέλων. Εφαρμόζει τους κανόνες επιχειρησιακής λογικής (Business Logic) και ελέγχει τα δικαιώματα πρόσβασης.
- Model (Μοντέλο): Αναλαμβάνει την επικοινωνία με τη βάση δεδομένων, εκτελεί ερωτήματα SQL (Select, Insert, Update, Delete) και επιστρέφει τα δεδομένα στον Controller.
- View (Προβολή): Λαμβάνει τα δεδομένα από τον Controller και τα παρουσιάζει στον χρήστη σε μορφή HTML. Τα Views δεν περιέχουν λογική της εφαρμογής, παρά μόνο λογική παρουσίασης (π.χ. βρόχους για λίστες).

4.2 Σχεδιασμός βάσης δεδομένων

Ο σχεδιασμός της βάσης δεδομένων αποτελεί ένα κεντρικό στοιχείο της εφαρμογής, καθώς καθορίζει τον τρόπο με τον οποίο αποθηκεύονται και συσχετίζονται οι πληροφορίες για τους χρήστες, τις αίθουσες εκδηλώσεων, τις επισκέψεις και τις κρατήσεις αυτών. Η βάση δεδομένων υλοποιήθηκε σε MySQL (έκδοση 8.0) με χρήση της μηχανής αποθήκευσης InnoDB, η οποία υποστηρίζει δοσοληψίες (transactions) και

περιορισμούς ξένων κλειδιών, παρέχοντας εγγυήσεις για την ακεραιότητα των δεδομένων.

Το σχήμα της βάσης δεδομένων (musedb) οργανώνεται σε λογικές ενότητες πινάκων οι οποίες απεικονίζονται στην Εικόνα 5 και περιγράφονται στις επόμενες παραγράφους.



Εικόνα 5. Σχεδιάγραμμά Βάσης Δεδομένων MuseApp

1. Διαχείριση Χρηστών και Ρόλων

- *users*: Ο κεντρικός πίνακας αποθήκευσης χρηστών. Περιλαμβάνει πεδία για τα προσωπικά στοιχεία (όνομα, τηλέφωνο, email), διαπιστευτήρια πρόσβασης (κρυπτογραφημένο password) και στοιχεία οργανισμού (εταιρεία ή σχολείο).

- *roles*: Καθορίζει τους υπαρκτούς ρόλους και τα επίπεδα πρόσβασης για τον καθένα από αυτούς (Administrator, Venue Manager, Tour Manager, User).
- *Συσχέτιση*: Ο πίνακας users συνδέεται με τον πίνακα roles με σχέση 1 - προς - πολλά (ένας ρόλος μπορεί να ανατεθεί σε πολλούς χρήστες), επιτρέποντας την εφαρμογή του μοντέλου ελέγχου πρόσβασης.

2. Διαχείριση Πόρων (Αίθουσες Εκδηλώσεων και Σχολικές Επισκέψεις)

- *venues*: Αποθηκεύει τις πληροφορίες για τις αίθουσες προς ενοικίαση/παραχώρηση (χωρητικότητα, τιμή, περιγραφή). Περιλαμβάνει πεδία τύπου boolean (όπως Monday, Tuesday) για τον ορισμό των ημερών λειτουργίας.
- *tours*: Αποθηκεύει τις διαθέσιμες ζώνες επισκέψεων για σχολεία, με συγκεκριμένη ώρα έναρξης και λήξης.
- *photo*: Πίνακας για την αποθήκευση φωτογραφιών των αιθουσών. Συνδέεται με την οντότητα της αίθουσας (venues) με σχέση 1 - προς - πολλά, με δυνατότητα ορισμού κύριας φωτογραφίας (is_primary). Αποθηκεύεται η διαδρομή της φωτογραφίας μέσα στον φάκελο της εφαρμογής.

3. Διαχείριση Διαθεσιμότητας και Εξαιρέσεων

Για την κάλυψη της απαίτησης για ευέλικτο ωράριο, δημιουργήθηκαν πίνακες για την καταχώρηση των εξαιρέσεων, (α) για τις αίθουσες (*venue_exceptions*) και τις ζώνες επισκέψεων (*tour_exceptions*). Επιτρέπουν στον διαχειριστή να ορίζει συγκεκριμένες ημερομηνίες κατά τις οποίες μια αίθουσα ή ένα πρόγραμμα επίσκεψης δεν είναι διαθέσιμα (όπως αργίες, συντήρηση, άδεια ξεναγού), παρακάμπτοντας τον γενικό κανόνα διαθεσιμότητας.

4. Διαχείριση Κρατήσεων

Οι κρατήσεις διαχωρίζονται σε δύο οντότητες λόγω των διαφορετικών πεδίων που απαιτούν:

- *bookVenues*: Κρατήσεις αιθουσών εκδηλώσεων.

- *bookTours*: Κρατήσεις επισκέψεων (περιλαμβάνει πεδία όπως αριθμό παιδιών και τάξη).
- *statuses*: Πίνακας αναφοράς για την κατάσταση της κράτησης (Pending, Accepted, Rejected, Cancelled).

Η δομή των πινάκων υποστηρίζει και τη λειτουργικότητα *on_behalf_of*, για καταχώρηση κράτησης «εκ μέρους» τρίτου. Στο πλαίσιο αυτό, και οι δύο πίνακες κρατήσεων περιλαμβάνουν πεδία για καταχώρηση κράτησης «εκ μέρους» τρίτου, επιτρέποντας στους διαχειριστές να δημιουργούν κρατήσεις για πελάτες που δεν έχουν λογαριασμό στην εφαρμογή.

5. Καταγραφή και Ρυθμίσεις

- *activity_logs*: Καταγράφει κρίσιμες ενέργειες των χρηστών για λόγους ασφαλείας, ελέγχου, και καταλογισμού/λογοδοσίας.
- *settings*: Πίνακας ζευγών key - value για δυναμική διαχείριση ρυθμίσεων της εφαρμογής (όπως στοιχεία επικοινωνίας, ωράριο λειτουργίας τμημάτων του οργανισμού), ώστε αυτά να καταγράφονται δυναμικά και να μην απαιτείται αλλαγή στον κώδικα της εφαρμογής ή σε αρχεία διαμόρφωσής της.

4.3 Σχεδίαση διεπαφής χρήστη

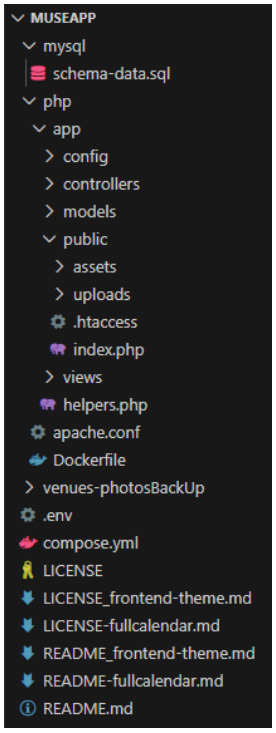
Η σχεδίαση της διεπαφής χρήστη (User Interface) είχε ως κύριο γνώμονα την ευχρηστία (Usability) και την προσαρμοστικότητα (Responsiveness). Χρησιμοποιήθηκε το πλαίσιο Bootstrap 4, το οποίο εξασφαλίζει ότι η εφαρμογή εμφανίζεται σωστά σε οθόνες υπολογιστών, tablets και κινητών τηλεφώνων. Χρησιμοποιήθηκε το σύστημα Grid του Bootstrap για τη διάταξη των στοιχείων. Επιλέχθηκε μια «καθαρή» χρωματική παλέτα για να παρέχεται ξεκούραστη ανάγνωση, ενώ για την οπτική αναπαράσταση ενεργειών και καταστάσεων χρησιμοποιήθηκε η βιβλιοθήκη εικονιδίων FontAwesome 5 (όπως εικονίδια κάδου για διαγραφή, στυλό για επεξεργασία).

Η πλοήγηση υλοποιήθηκε με μια σταθερή μπάρα (Navbar) και ένα πλευρικό μενού (Sidebar) για το διαχειριστικό περιβάλλον, ώστε ο χρήστης να έχει πάντα πρόσβαση στις βασικές λειτουργίες.

Για ενέργειες όπως η "Κράτηση" ή η "Διαγραφή", χρησιμοποιήθηκαν αναδυόμενα παράθυρα (Modals) ώστε ο χρήστης να μην χάνει την επαφή με την τρέχουσα σελίδα.

4.4 Δομή των φακέλων και αρχείων

Η οργάνωση των αρχείων στον διακομιστή ακολουθεί αυστηρά τη λογική του MVC, διαχωρίζοντας τον δημόσιο φάκελο από τον πυρήνα της εφαρμογής για λόγους οργάνωσης του κώδικα και ασφαλείας. Η δομή του έργου απεικονίζεται στην Εικόνα 6, ενώ το περιεχόμενο των κύριων φακέλων περιγράφεται στη συνέχεια.



Εικόνα 6. Δομή φακέλων εφαρμογής MuseApp

- `/mysql`: Περιέχει τα αρχεία αρχικοποίησης της βάσης δεδομένων (`schema-data.sql`).
- `/php`: Ο κεντρικός φάκελος της εφαρμογής.
- `/app`: Περιέχει όλη τη λογική (δεν είναι προσβάσιμος απευθείας από τον browser).
- `/config`: Αρχεία ρυθμίσεων (σύνδεση με βάση).
- `/controllers`: Οι κλάσεις των Ελεγκτών (π.χ. BookingController.php).
- `/models`: Οι κλάσεις των Μοντέλων (π.χ. User.php).
- `/views`: Τα αρχεία HTML/PHP για την προβολή (π.χ. home.php).

- `/public`: Ο μόνος δημόσια προσβάσιμος φάκελος.
- `index.php`: Το σημείο εισόδου της εφαρμογής.
- `/assets`: Στατικά αρχεία (CSS, JS, Images).
- `/uploads`: Φάκελος αποθήκευσης φωτογραφιών αιθουσών.
- `docker-compose.yml`: Αρχείο διαμόρφωσης των υπηρεσιών Docker.

4.5 Ενδεικτικά παραδείγματα κώδικα

Για την κατανόηση της αρχιτεκτονικής του συστήματος, παρουσιάζεται η ροή μιας τυπικής λειτουργίας (Request - Response Cycle) και συγκεκριμένα η διαδικασία προβολής της λίστας των χρηστών. Η διαδικασία ακολουθεί αυστηρά το πρότυπο MVC.

1. Το αίτημα του χρήστη φτάνει αρχικά στον UserController. Η μέθοδος index() ελέγχει αν ο χρήστης έχει δικαίωμα πρόσβασης (εν προκειμένω εάν είναι Super Admin). Αν ο έλεγχος είναι επιτυχής, καλεί το Model για να ζητήσει τα δεδομένα, ειδάλλως εμφανίζει μήνυμα σφάλματος.

```php
    /**
     * Εμφανίζει τη λίστα όλων των χρηστών.
     */
    public function index()
    {
        // Μόνο ο super admin μπορεί να δει τη λίστα των χρηστών.
        if (!is_super_admin()) {
            http_response_code(403);
            require_once BASE_PATH . '/views/errors/403.php';
            exit();
        }

        $users = $this->userModel->getAll();
        require_once BASE_PATH . '/views/users/index.php';
    }
```

2. Στο model η κλάση User εκτελεί την ερώτηση SQL έναντι της βάσης δεδομένων, ανακτά τους χρήστες με τους ρόλους τους και επιστρέφει τα αποτελέσματα στον Controller.

```php
    /**
     * Επιστρέφει όλους τους χρήστες μαζί με το όνομα του ρόλου τους.
     */
    public function getAll()
    {
        $sql = "SELECT
                    u.idUser, u.username, u.firstname, u.lastname,
u.mail, u.phone, u.company, u.active, r.roleName
                FROM
                    users u
                JOIN
                    roles r ON u.idRole = r.idRole
                ORDER BY
                    u.lastname ASC, u.firstname ASC";

        $stmt = $this->pdo->query($sql);
        return $stmt->fetchAll();
    }
```

3. Ο Controller φορτώνει το αρχείο index.php (View), περνώντας τα δεδομένα (μέσω της μεταβλητής $users). Το View αναλαμβάνει την τελική μορφοποίηση HTML για τον browser.

```php
        <div class="card-header py-3">
            <h6 class="m-0 font-weight-bold text-primary">Εγγεγραμμένοι
Χρήστες</h6>
        </div>
        <div class="card-body">
            <div class="table-responsive">
                <table class="table table-bordered" id="dataTable"
width="100%" cellspacing="0">
                    <thead>
                        <tr>
                            <th>ID</th>
                            <th>Ονοματεπώνυμο</th>
                            <th>Username</th>
                            <th>Email</th>
                            <th>Ρόλος</th>
                            <th>Κατάσταση</th>
                            <th>Ενέργειες</th>
                        </tr>
                    </thead>
                    <tbody>
                        <?php foreach ($users as $user): ?>
                            <tr>
```

```
                                <td><?=
htmlspecialchars($user['idUser']) ?></td>
                                <td><?=
htmlspecialchars($user['lastname'] . ' ' . $user['firstname']) ?></td>
                                <td><?=
htmlspecialchars($user['username']) ?></td>
                                <td><?= htmlspecialchars($user['mail'])
?></td>
                                <td>
                                    <?php
                                        $roleName =
htmlspecialchars($user['roleName']);
                                        $badgeClass = '';
                                        switch ($roleName) {
                                            case 'administrator':
                                                $badgeClass = 'badge-
danger';
                                                break;
                                            case 'tour_manager':
                                            ...
                                            ...
                                            ...
```

4.6 Δρομολόγηση και Σημείο Εισόδου

Η εφαρμογή ακολουθεί το πρότυπο σχεδίασης *Front Controller*. Αυτό σημαίνει ότι όλα τα αιτήματα (HTTP Requests) προς την εφαρμογή κατευθύνονται σε ένα μοναδικό αρχείο εισόδου, το `public/index.php`.

Στο αρχείο `index.php` πραγματοποιείται η αρχικοποίηση του περιβάλλοντος. Αναλυτικότερα γίνεται η διαχείριση σφαλμάτων (Error Handling). Ορίζεται ένας κεντρικός μηχανισμός που καταγράφει τα σφάλματα και εμφανίζει φιλική σελίδα με κωδικό σφάλματος τύπου 500 στον χρήστη, για τις περιπτώσεις όπου υπάρχει δυσλειτουργία του διακομιστή ή της εφαρμογής. Επιπλέον γίνεται η διασύνδεση με την Βάση Δεδομένων. Δημιουργείται το αντικείμενο σύνδεσης με τη βάση (PDO).

Τέλος, όσο αφορά των Μηχανισμό Δρομολόγησης (Router), δεν χρησιμοποιήθηκε έτοιμη βιβλιοθήκη, αλλά αναπτύχθηκε ένας μηχανισμός βασισμένος σε κανονικές εκφράσεις (Regular Expressions). Ο πίνακας `$routes` αντιστοιχίζει μοτίβα URL (π.χ. `GET /venue/{id}`) σε ανώνυμες συναρτήσεις που καλούν την κατάλληλη μέθοδο του Controller. Ο router διαβάζει το URL, αντικαθιστά τις δυναμικές παραμέτρους

(όπως {id}) αξιοποιώντας μοτίβα κανονικών εκφράσεων (regex patterns), και αν βρεθεί αντιστοιχία, εκτελεί τον αντίστοιχο κώδικα.

5 ΥΛΟΠΟΙΗΣΗ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

Το παρόν κεφάλαιο περιγράφει τη μετατροπή του σχεδιασμού σε λειτουργικό κώδικα. Εδώ αναλύεται διεξοδικά ο τρόπος ανάπτυξης των βασικών υποσυστημάτων της εφαρμογής, όπως η διαχείριση χρηστών, ο μηχανισμός κρατήσεων και το σύστημα ειδοποιήσεων, παραθέτοντας τμήματα κώδικα και εξηγώντας τη λογική πίσω από αυτά.

5.1 Περιβάλλον ανάπτυξης

Η ανάπτυξη της εφαρμογής πραγματοποιήθηκε σε τοπικό περιβάλλον, χρησιμοποιώντας ένα σύνολο σύγχρονων εργαλείων που εξασφαλίζουν ταχύτητα, οργάνωση και συμβατότητα. Συγκεκριμένα χρησιμοποιήθηκαν:

- Λειτουργικό Σύστημα: Windows 11 (ως host machine).
- Επεξεργαστής Κώδικα (IDE): Visual Studio Code, εμπλουτισμένο με επεκτάσεις για PHP , Docker και SQL, το οποίο προσφέρει δυνατότητες debugging και αυτόματης συμπλήρωσης κώδικα.
- Πλατφόρμα Containerization: Docker Desktop, για τη δημιουργία και διαχείριση των containers της εφαρμογής και της βάσης δεδομένων.
- Version Control: Git, για την καταγραφή του ιστορικού αλλαγών και τη διαχείριση του πηγαίου κώδικα, ο οποίος έχει αναρτηθεί σε δημόσιο αποθετήριο στο GitHub [24].
- Web Browser: Google Chrome για τον έλεγχο της ορθής απεικόνισης και τη χρήση των DevTools (Network tab, Console) κατά την ανάπτυξη του Frontend και του API.

5.2 Υλοποίηση βασικών λειτουργιών

Στην ενότητα αυτή αναλύεται ο τρόπος υλοποίησης των κεντρικών υποσυστημάτων της εφαρμογής, εστιάζοντας στη λογική, την ασφάλεια και τη διαχείριση δεδομένων.

5.2.1 Αυθεντικοποίηση και Έλεγχος Πρόσβασης

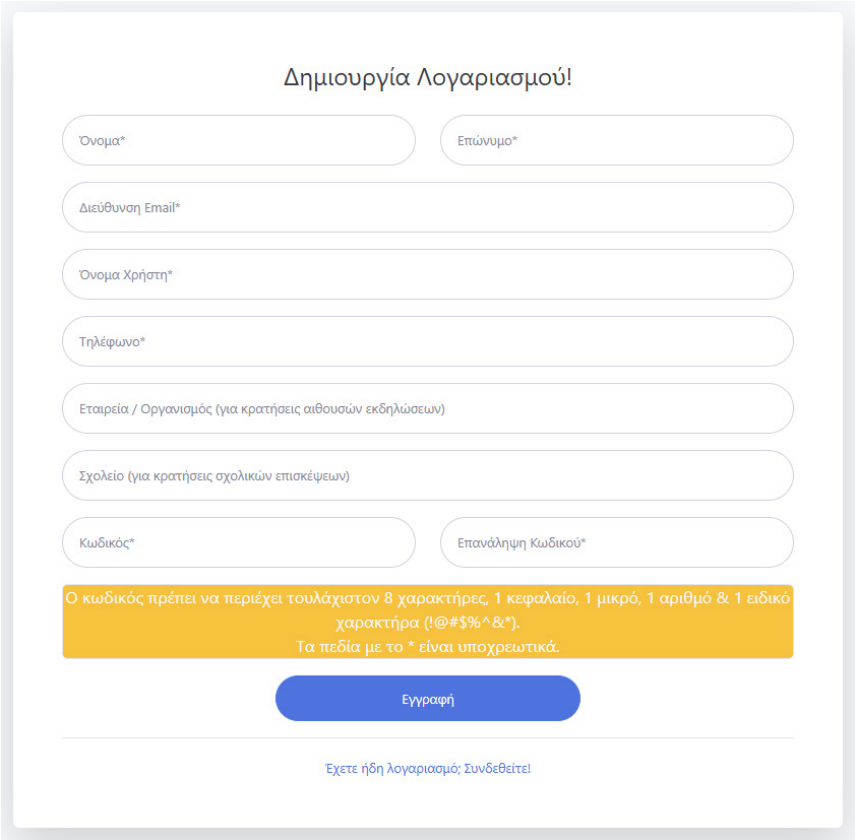
Η ασφάλεια της εφαρμογής βασίζεται σε δύο παράγοντες, (α) την αυθεντικοποίηση, την εξέταση δηλαδή του ποιος είναι ο χρήστης και (β) την εξουσιοδότηση, τον έλεγχο

δηλαδή του τι επιτρέπεται να κάνει και εάν έχει τη δυνατότητα να εκτελέσει την αιτούμενη ενέργεια.

Α. Διαχείριση Συνόδου (session)

Η λειτουργία της αυθεντικοποίησης υλοποιείται στην ενότητα AuthController. Οι δύο σελίδες που υλοποιεί ο συγκεκριμένος ελεγκτής είναι η σελίδα εγγραφής και η σελίδα σύνδεσης.

Κατά την εγγραφή, εφαρμόζεται αυστηρή πολιτική στον ορισμό των συνθηματικών, δηλαδή απαιτούνται τουλάχιστον 8 χαρακτήρες, που να περιλαμβάνουν κεφαλαία και μικρά γράμματα, αριθμοί και σύμβολα. Η πολιτική αυτή εφαρμόζεται επίσης σε κάθε αλλαγή συνθηματικού.



Εικόνα 7. Φόρμα εγγραφής

Στη σελίδα σύνδεσης, ο χρήστης εισάγει τα στοιχεία του. Το σύστημα ελέγχει αν υπάρχει ο χρήστης, αν είναι ενεργός και επαληθεύει τον κωδικό. Σε περίπτωση επιτυχίας, αποθηκεύονται στη μεταβλητή `$_SESSION` το ID, το username και ο ρόλος του χρήστη.

Β. Ρόλοι και Δικαιώματα

Για τον έλεγχο πρόσβασης, δημιουργήθηκαν στο αρχείο `helpers.php`, το οποίο περιέχει βοηθητικές συναρτήσεις, οι ακόλουθες που ελέγχουν τη σύνοδο (Session) του χρήστη. Οι βασικές βοηθητικές συναρτήσεις είναι οι ακόλουθες:

- `is_logged_in()`: Ελέγχει αν υπάρχει συνδεδεμένος χρήστης.
- `has_role($role)`: Ελέγχει αν ο χρήστης έχει συγκεκριμένο ρόλο.
- `is_super_admin()`, `is_venue_manager()`, `is_tour_manager()`, `is_simple_user()`: Η κάθε συνάρτηση ελέγχει αν ο χρήστης ανήκει στον συγκεκριμένο ρόλο. Οι συναρτήσεις αυτές χρησιμοποιούνται σε όλη την έκταση του κώδικα (Controllers και Views) για να επιτρέψουν ή να απαγορεύσουν ενέργειες με βάση τον ρόλο του χρήστη.

```
// ====================================================================
// HELPERS ΓΙΑ AUTHENTICATION & ROLES
// ====================================================================

if (!function_exists('is_logged_in')) {
    /**
     * Ελέγχει αν ο χρήστης είναι συνδεδεμένος.
     * @return bool
     */
    function is_logged_in(): bool
    {
        return isset($_SESSION['user_id']);
    }
}

if (!function_exists('has_role')) {
    /**
     * (Internal Helper) Ελέγχει αν ο συνδεδεμένος χρήστης έχει έναν
συγκεκριμένο ρόλο.
     * @param string $roleName Το όνομα του ρόλου όπως είναι στη βάση
δεδομένων.
     * @return bool
     */
    function has_role(string $roleName): bool
    {
        return isset($_SESSION['user_role']) && $_SESSION['user_role']
=== $roleName;
    }
}
```

```php
if (!function_exists('is_super_admin')) {
    /**
     * Ελέγχει αν ο χρήστης είναι Super Administrator.
     * @return bool
     */
    function is_super_admin(): bool
    {
        return has_role('administrator');
    }
}

if (!function_exists('is_venue_manager')) {
    /**
     * Ελέγχει αν ο χρήστης είναι Venue Manager.
     * @return bool
     */
    function is_venue_manager(): bool
    {
        return has_role('venue_manager');
    }
}

if (!function_exists('is_tour_manager')) {
    /**
     * Ελέγχει αν ο χρήστης είναι Tour Manager.
     * @return bool
     */
    function is_tour_manager(): bool
    {
        return has_role('tour_manager');
    }
}

if (!function_exists('is_simple_user')) {
    /**
     * Ελέγχει αν ο χρήστης είναι απλός User.
     * @return bool
     */
    function is_simple_user(): bool
    {
        return has_role('user');
    }
}
```

Γ. Μηχανισμός Flash Messages

Για την αλληλεπίδραση με τον χρήστη, με μηνύματα όπως «Η εγγραφή πέτυχε» ή «Λάθος κωδικός», υλοποιήθηκε η συνάρτηση flash(). Αυτή αποθηκεύει προσωρινά μηνύματα στο Session, τα οποία εμφανίζονται μόνο στην επόμενη αίτηση και στη συνέχεια διαγράφονται, βελτιώνοντας την εμπειρία χρήστη (UX).

```
// ====================================================================
// HELPERS ΓΙΑ ΜΗΝΥΜΑΤΑ (FLASH MESSAGES)
// ====================================================================

if (!function_exists('flash')) {
    /**
     * Ορίζει, ανακτά και εμφανίζει flash messages.
     * - ΟΡΙΣΜΟΣ: flash('success', 'Το μήνυμά σου εδώ.');
     * - ΕΜΦΑΝΙΣΗ: flash();
     * @param string|null $type 'success', 'danger', 'info'
     * @param string|null $message Το μήνυμα προς εμφάνιση.
     * @param bool $allow_html Αν θα επιτρέπεται η εμφάνιση HTML tags
στο μήνυμα.
     */
    function flash($type = null, $message = null, $allow_html = false):
void
    {
        // Ορισμός μηνύματος
        if ($type && $message) {
            $_SESSION['flash_messages'][] = [
                'type' => $type,
                'message' => $message,
                'allow_html' => $allow_html
            ];
            return;
        }

        // Εμφάνιση μηνυμάτων
        if (isset($_SESSION['flash_messages']) &&
is_array($_SESSION['flash_messages'])) {
            foreach ($_SESSION['flash_messages'] as $flash) {
                $display_message = $flash['allow_html'] ?
$flash['message'] : htmlspecialchars($flash['message']);
                echo '<div class="alert alert-' .
htmlspecialchars($flash['type']) . '">' . $display_message . '</div>';
            }
            unset($_SESSION['flash_messages']);
        }
    }
```

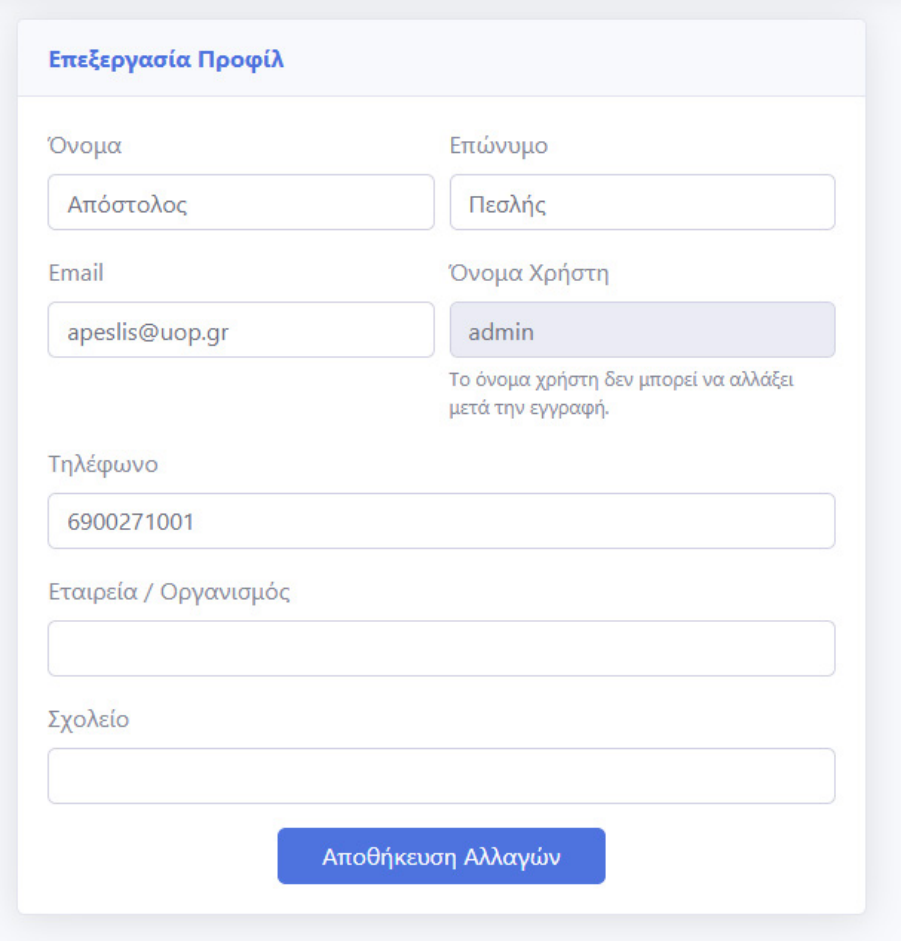
```
}
```

5.2.2 Διαχείριση Προφίλ Χρήστη

Η δυνατότητα αυτοδιαχείρισης του λογαριασμού είναι κρίσιμη για την εμπειρία χρήστη και την ασφάλεια. Η λειτουργία αυτή υλοποιείται μέσω του ProfileController και επιτρέπει στους συνδεδεμένους χρήστες να ενημερώνουν τα προσωπικά τους στοιχεία και να αλλάζουν τον κωδικό πρόσβασής τους.

Α. Ενημέρωση Στοιχείων

Η μέθοδος `updateProfile` διαχειρίζεται την αλλαγή στοιχείων όπως Όνομα, Επώνυμο, Email, Τηλέφωνο και, προαιρετικά, την Εταιρεία ή το Σχολείο. Κατά την υποβολή της φόρμας (`views/profile/edit.php`) πραγματοποιείται έλεγχος εγκυρότητας (validation) για τα υποχρεωτικά πεδία. Επιπλέον, το σύστημα αποτρέπει τη χρήση email που έχει ήδη καταχωρηθεί από άλλον χρήστη, διασφαλίζοντας έτσι τη μοναδικότητα.



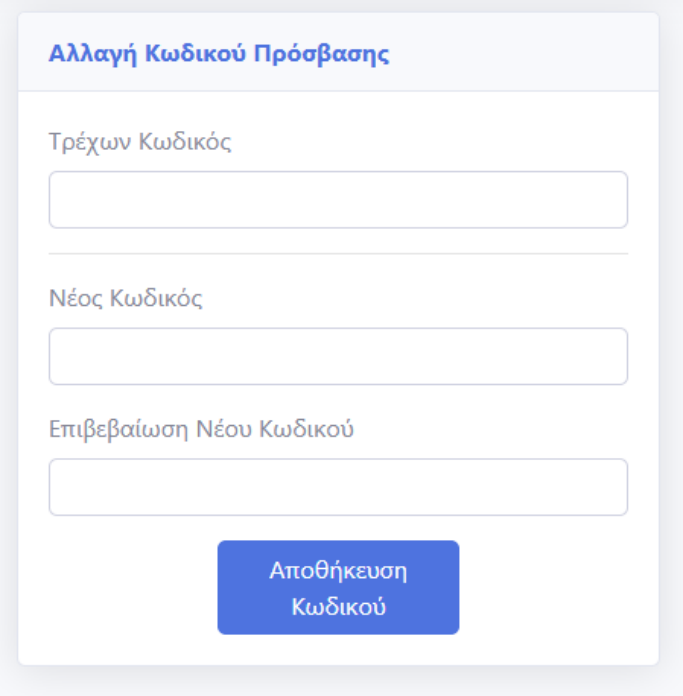
Εικόνα 8. Φόρμα επεξεργασίας προφίλ από χρήστη

Β. Ασφαλής Αλλαγή Κωδικού

Η διαδικασία αλλαγής κωδικού (`updatePassword`) εφαρμόζει τις ακόλουθες τεχνικές για τη διασφάλιση της διαδικασίας.

- Επαλήθευση Τρέχοντος Κωδικού: Πριν επιτραπεί η αλλαγή, ο χρήστης πρέπει να εισάγει τον τρέχοντα κωδικό του, ο οποίος ελέγχεται με την `password_verify()`. Αυτό αποτρέπει την αλλαγή κωδικού σε περίπτωση που κάποιος βρει τον υπολογιστή του χρήστη ξεκλείδωτο.
- Πολιτική Ισχυρού Κωδικού: Ο νέος κωδικός ελέγχεται μέσω ώστε να πληροί τις ίδιες προϋποθέσεις με την εγγραφή.
- Επιβεβαίωση: Απαιτείται διπλή εισαγωγή του νέου κωδικού για την αποφυγή τυπογραφικών λαθών.

Κάθε αλλαγή στο προφίλ ή στον κωδικό καταγράφεται στο ActivityLog, ενημερώνοντας το σύστημα για το πότε και από ποιον έγιναν οι τροποποιήσεις.



Εικόνα 9. Φόρμα αλλαγής κωδικού πρόσβασης

5.2.3 Διαχείριση χρηστών και ρόλων

Το υποσύστημα διαχείρισης χρηστών αποτελεί τη βάση για τον έλεγχο πρόσβασης στην εφαρμογή (RBAC – Role Based Access Control). Η υλοποίηση βασίστηκε στην κλάση `UserController` και το μοντέλο User.

Α. Ασφάλεια και Αυθεντικοποίηση

Κατά την εγγραφή ή τροποποίηση ενός χρήστη, δόθηκε ιδιαίτερη έμφαση στην ασφάλεια των δεδομένων. Οι κωδικοί πρόσβασης δεν αποθηκεύονται ως απλό κείμενο, αλλά κρυπτογραφούνται μέσω της συνάρτησης `password_hash()` της PHP, η οποία χρησιμοποιεί τον αλγόριθμο BCRYPT, όπως φαίνεται στη μέθοδο `create` του μοντέλου, αλλά και στον πίνακα που αποθηκεύεται στη βάση δεδομένων (Εικόνα 10).

```
$password_hash = password_hash($data['password'], PASSWORD_BCRYPT);
```

idUser	username	firstname	lastname	password
1	admin	Απόστολος	Πεσλής	$2y$10$CS1Zq7jM8GLmwLyDl5cFl.XMLfOlevooOim0cJL1VNDzwK.ETHWMW
2	adminvenues	Νίκος	Νικολάου	$2y$10$CS1Zq7jM8GLmwLyDl5cFl.XMLfOlevooOim0cJL1VNDzwK.ETHWMW
3	admintours	Παύλος	Παύλου	$2y$10$CS1Zq7jM8GLmwLyDl5cFl.XMLfOlevooOim0cJL1VNDzwK.ETHWMW
4	g.pappas	Γιώργος	Παππάς	$2y$10$vEsmtLt1xZj6AK0bizlAF.KX6hwTMaYdiqm7yPs3ABku/JcK34Bs.

Εικόνα 10. Στιγμιότυπο από τον πίνακα Users

Επιπλέον, όλα τα ερωτήματα προς τη βάση δεδομένων γίνονται αποκλειστικά μέσω Prepared Statements της βιβλιοθήκης PDO, αποτρέποντας επιθέσεις τύπου SQL Injection.

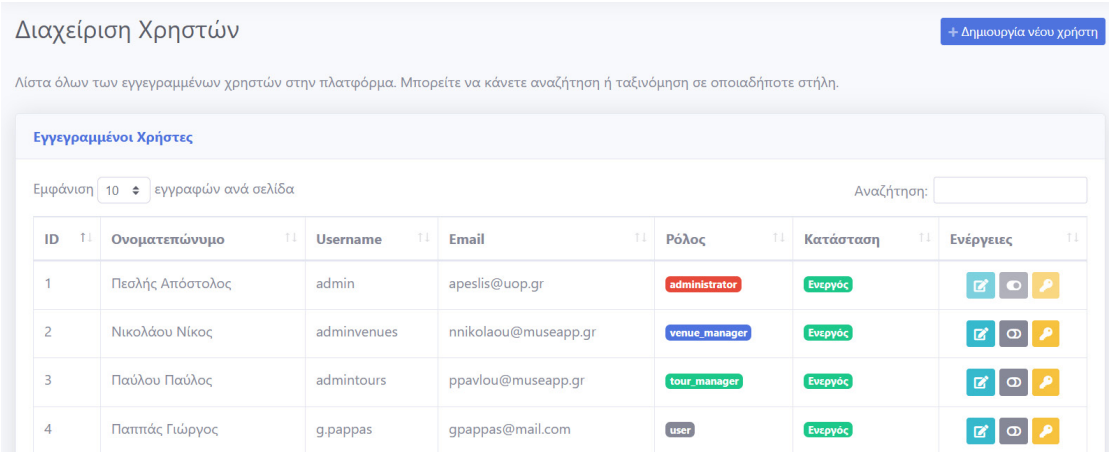
Παράλληλα κάθε κρίσιμη ενέργεια στον Controller που αφορά τους χρήστες προστατεύεται από έλεγχο δικαιωμάτων, διασφαλίζοντας ότι μόνο εξουσιοδοτημένοι χρήστες μπορούν να δημιουργήσουν ή να επεξεργαστούν λογαριασμούς.

```php
/**
 * Εμφανίζει τη φόρμα επεξεργασίας για έναν χρήστη.
 */
public function edit(int $id)
{
    if (!is_super_admin()) {
        http_response_code(403);
        require_once BASE_PATH . '/views/errors/403.php';
        exit();        }
```

Β. Λειτουργίες CRUD (Create, Read, Update, Delete/Deactivate)

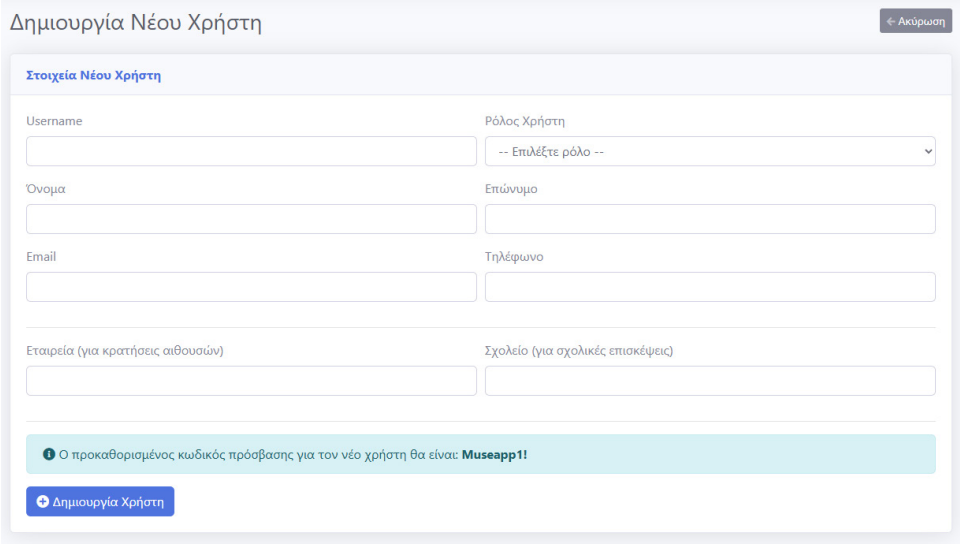
Η εφαρμογή παρέχει πλήρες περιβάλλον διαχείρισης:

1. Προβολή (Read): Η λίστα χρηστών υλοποιείται στο View `users/index.php` με χρήση της βιβλιοθήκης DataTables (jQuery), προσφέροντας δυνατότητες άμεσης αναζήτησης, σελιδοποίησης και ταξινόμησης χωρίς επιπλέον φόρτο στον διακομιστή.



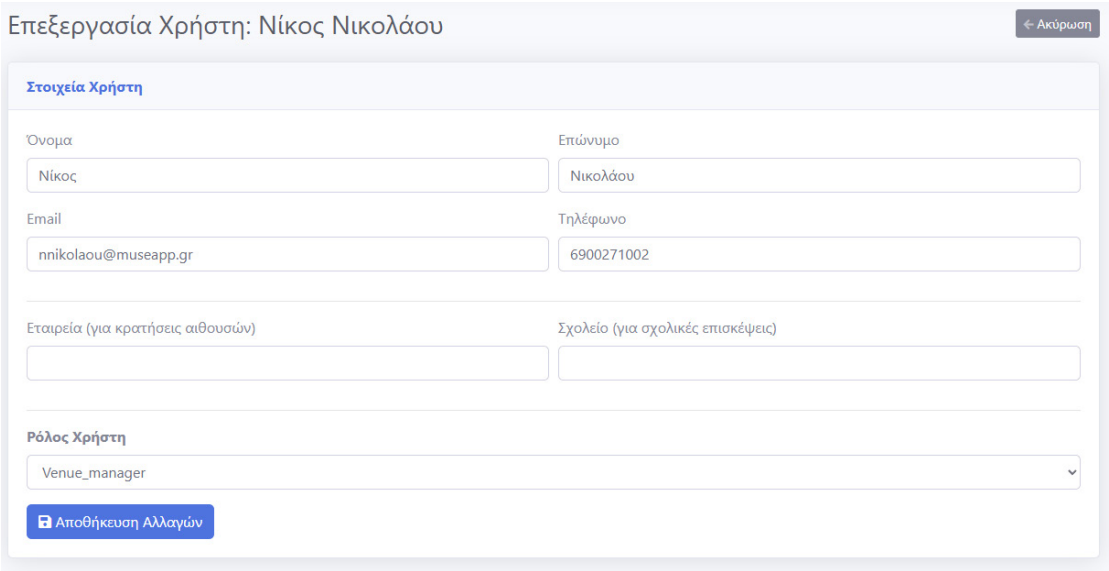
Εικόνα 11. Στιγμιότυπο του πίνακα Χρηστών

2. Δημιουργία (Create): Η φόρμα δημιουργίας (`users/create.php`) περιλαμβάνει ελέγχους εγκυρότητας τόσο στην πλευρά του client όσο και στον server. Ειδική μέριμνα λήφθηκε για τον ορισμό προκαθορισμένου κωδικού (λαμβάνει την τιμή *Museapp1!*) κατά τη δημιουργία χρήστη από τον διαχειριστή.



Εικόνα 12. Φόρμα δημιουργίας χρήστη από τον διαχειριστή

3. Ενημέρωση και Κατάσταση (Update/Status): Ο διαχειριστής έχει τη δυνατότητα να αλλάξει τα στοιχεία ενός χρήστη ή να τον απενεργοποιήσει («Soft Delete») μέσω της μεθόδου `toggleActiveStatus`. Το σύστημα αποτρέπει την κατά λάθος απενεργοποίηση του ιδίου του διαχειριστή για λόγους ασφάλειας.



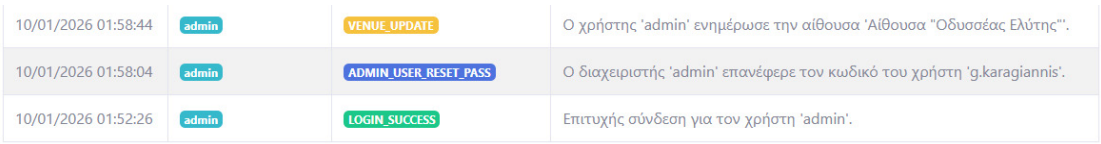
Εικόνα 13. Επεξεργασία χρήστη από τον διαχειριστή

Γ. Καταγραφή Ενεργειών

Σημαντικό στοιχείο της υλοποίησης είναι η διασύνδεση του `UserController` με το μοντέλο `ActivityLog`. Κάθε σημαντική ενέργεια (όπως δημιουργία χρήστη, επαναφορά κωδικού) καταγράφεται αυτόματα στον πίνακα `activity_logs`, ενισχύοντας την ιχνηλασιμότητα του συστήματος.

Παράδειγμα κώδικα καταγραφής:

```
$this->logModel->logAction(
                'ADMIN_USER_CREATE',
                "Ο διαχειριστής '{$adminUsername}' δημιούργησε τον νέο
χρήστη '{$data['username']}'.",
                $_SESSION['user_id']
            );
```



Εικόνα 14. Στιγμιότυπο πίνακα Logs

5.2.4 Δυναμική Διαχείριση Ρυθμίσεων και Περιεχομένου

Μια βασική απαίτηση του συστήματος ήταν η ευελιξία στη διαχείριση των πληροφοριών του οργανισμού, χωρίς την ανάγκη παρέμβασης στον πηγαίο κώδικα ή σε αρχεία διαμόρφωσης. Για τον σκοπό αυτό, υλοποιήθηκε το υποσύστημα Settings, το οποίο λειτουργεί ως ένα αποθετήριο ζευγών Κλειδιού-Τιμής (key – value store).

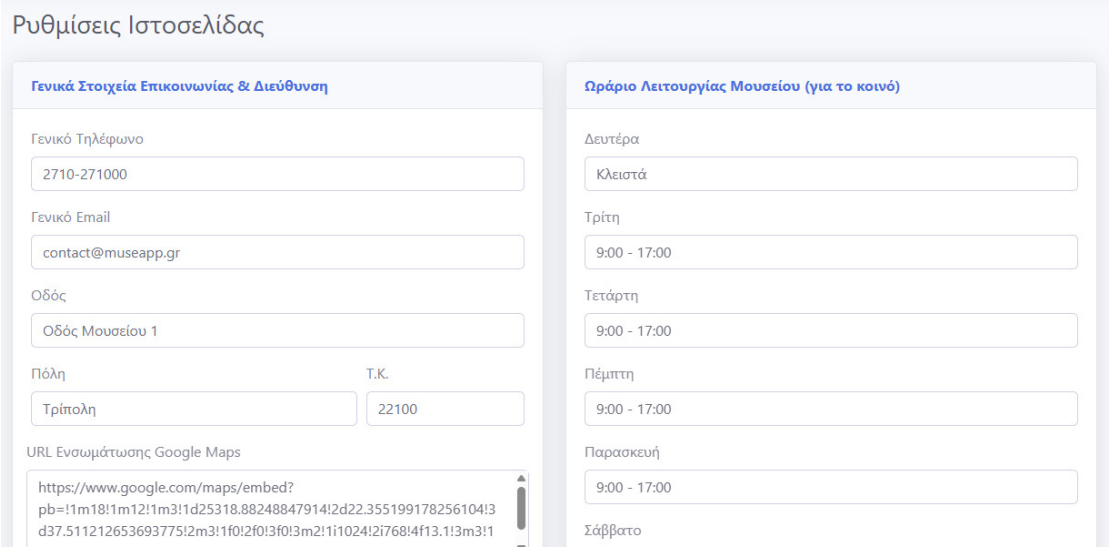
Α. Μοντέλο Δεδομένων και Ακεραιότητα

Οι ρυθμίσεις αποθηκεύονται στον πίνακα `settings` με δύο στήλες: `setting_key` και `setting_value`. Το μοντέλο `Settings.php` αναλαμβάνει την ανάκτηση όλων των ρυθμίσεων σε έναν συσχετιστικό πίνακα για γρήγορη πρόσβαση.

Ιδιαίτερη έμφαση δόθηκε στην ακεραιότητα των δεδομένων κατά την αποθήκευση. Η μέθοδος update χρησιμοποιεί δοσοληψίες (Transactions) της βάσης δεδομένων:

```
        $sql = "UPDATE settings SET setting_value = :value WHERE
setting_key = :key";
        $stmt = $this->pdo->prepare($sql);

        $this->pdo->beginTransaction();
        try {
            foreach ($settings as $key => $value) {
                $stmt->execute([':key' => $key, ':value' => $value]);
            }
            $this->pdo->commit();
            return true;
        } catch (PDOException $e) {
            $this->pdo->rollBack();
            error_log($e->getMessage());
            return false;
        }
```

Με αυτόν τον τρόπο διασφαλίζεται η ατομικότητα της ενέργειας, είτε θα αποθηκευτούν όλες οι ρυθμίσεις επιτυχώς, είτε καμία, αποτρέποντας τη μερική ενημέρωση της βάσης.



Εικόνα 15. Σελίδα διαχείρισης ρυθμίσεων ιστοσελίδας

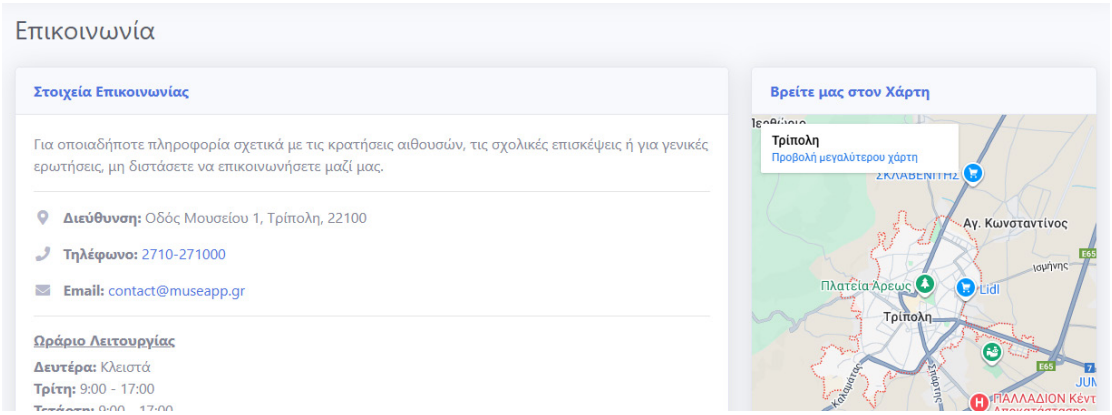
Β. Δυναμική Προβολή στο Frontend

Η πρακτική εφαρμογή του υποσυστήματος των ρυθμίσεων φαίνεται στη σελίδα «Επικοινωνία» (`pages/contact.php`). Για την άντληση των δεδομένων από τη βάση, το μοντέλο Settings χρησιμοποιεί την παρακάτω μέθοδο, η οποία επιστέφει όλες τις ρυθμίσεις με τη μορφή ενός συσχετιστικού πίνακα (key-value array):

```
    public function getAllAsArray(): array
    {
        $settings = [];
        $stmt = $this->pdo->query("SELECT setting_key, setting_value
FROM settings");

        while ($row = $stmt->fetch()) {
            $settings[$row['setting_key']] = $row['setting_value'];
        }

        return $settings;
    }
```

Στη συνέχεια, η διεπαφή (Frontend) χρησιμοποιεί αυτόν τον πίνακα για να προβάλει δυναμικά τα στοιχεία, αντί να τα αναγράφει στατικά:

```
<a href="mailto:<?= htmlspecialchars($settings['contact_email'])
?>"><?= htmlspecialchars($settings['contact_email']) ?></a>
```



Εικόνα 16. Σελίδα επικοινωνίας ιστοσελίδας

Επιπλέον, ενσωματώθηκε δυναμικά ο χάρτης της Google μέσω της ρύθμισης `Maps_url`, επιτρέποντας στον διαχειριστή να αλλάξει την τοποθεσία του οργανισμού απλώς επικολλώντας το νέο URL στο κατάλληλο πεδίο ρυθμίσεων στο διαχειριστικό περιβάλλον.

5.2.5 Διαχείριση αιθουσών

Η οντότητα «Αίθουσα Εκδηλώσεων» (Venue) αποτελεί έναν τον κεντρικό πόρο του συστήματος. Η διαχείρισή της υλοποιήθηκε μέσω του `VenueController` και περιλαμβάνει σύνθετες λειτουργίες πέραν των απλών CRUD (Create, Read, update, Delete).

Αρχικά, κατά τη δημιουργία ή επεξεργασία μιας αίθουσας, το σύστημα αποθηκεύει τα βασικά χαρακτηριστικά (χωρητικότητα, τιμή) αλλά και παραμετρικά στοιχεία, όπως τις ημέρες λειτουργίας (boolean πεδία Monday, Tuesday κ.λπ.). Ιδιαίτερη έμφαση δόθηκε στην ακεραιότητα αναφοράς. Όπως φαίνεται στη μέθοδο `deleteIfUnused`, το σύστημα απαγορεύει τη διαγραφή αίθουσας αν υπάρχουν συνδεδεμένες κρατήσεις.

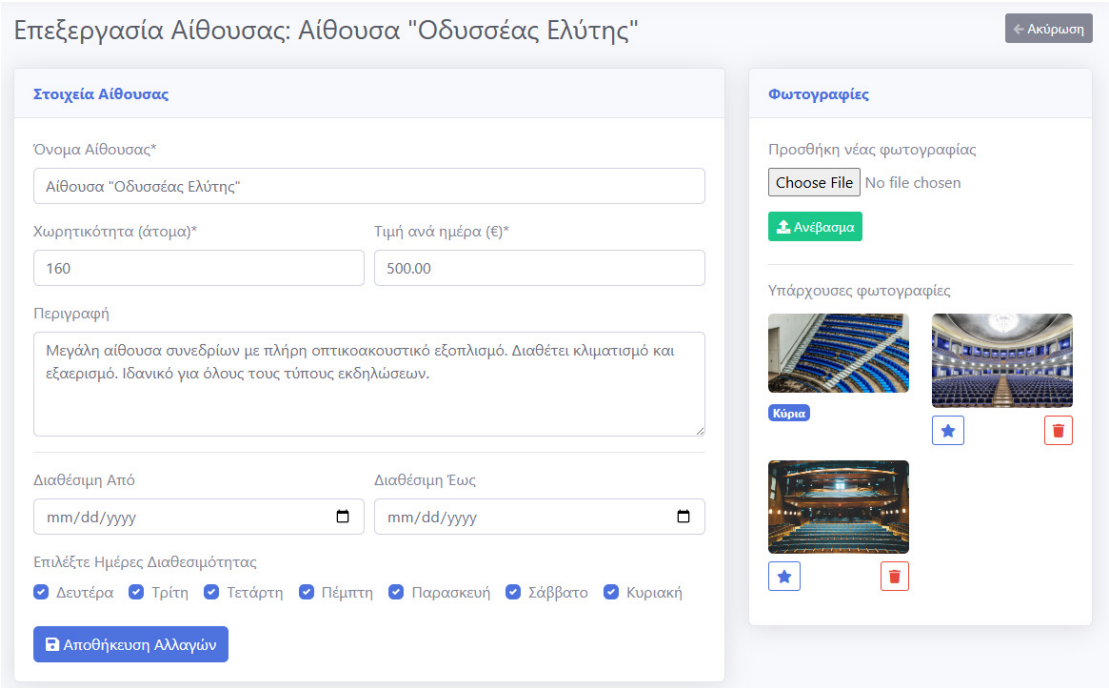
```
$stmt_check = $this->pdo->prepare("SELECT COUNT(*) FROM bookVenues
WHERE idVenue = ?");
$stmt_check->execute([$id]);

if ($stmt_check->fetchColumn() > 0) {
```

```
            flash('danger', 'Αυτή η αίθουσα δεν μπορεί να διαγραφεί
οριστικά γιατί υπάρχουν συνδεδεμένες κρατήσεις. Μπορείτε όμως να την
απενεργοποιήσετε.');
            return false;
        }
```

Σε αυτή την περίπτωση, προτείνεται η «Απενεργοποίηση» ώστε να διατηρείται το ιστορικό των κρατήσεων αλλά η αίθουσα να μην εμφανίζεται σε νέες αναζητήσεις.

Επιπλέον η εφαρμογή υποστηρίζει πολλαπλές φωτογραφίες ανά αίθουσα (σχέση 1-προς-Πολλά). Η υλοποίηση περιλαμβάνει:

- *Upload*: Έλεγχος τύπου αρχείου (jpg, png) και δημιουργία μοναδικού ονόματος (μέσω hashing) για την αποφυγή συγκρούσεων στο σύστημα αρχείων.
- *Primary Photo*: Δυνατότητα ορισμού μιας «Κύριας» φωτογραφίας που εμφανίζεται στις λίστες. Η μέθοδος setAsPrimary στο μοντέλο Photo χρησιμοποιεί δοσοληψία για να διασφαλίσει ότι μόνο μία φωτογραφία είναι επιλεγμένη ως κύρια κάθε φορά (ατομική ενημέρωση).



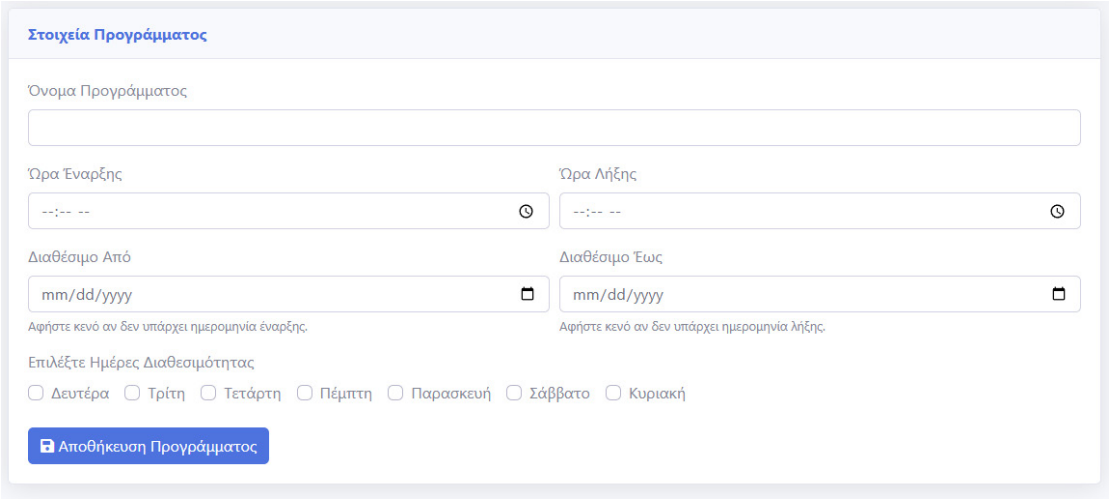
Εικόνα 17. Στιγμιότυπο από την επεξεργασία αίθουσας

5.2.6 Διαχείριση σχολικών επισκέψεων

Η δεύτερη βασική οντότητα του συστήματος αφορά τις επισκέψεις (Tours). Σε αντίθεση με τις αίθουσες που αφορούν ενοικίαση χώρου, οι επισκέψεις αντιπροσωπεύουν συγκεκριμένες χρονικές ζώνες κατά τις οποίες το μουσείο δέχεται ομάδες επισκεπτών.

Η κλάση `Tour` διαχειρίζεται τα προγράμματα επισκέψεων. Κάθε εγγραφή περιλαμβάνει:

- *Χρονικό Παράθυρο*: Τα πεδία `tourTimeStart` και `tourTimeFinish` καθορίζουν τη διάρκεια της επίσκεψης (π.χ. 09:00 - 10:30), ανάλογα με τον προγραμματισμό του πολιτιστικού οργανισμού για υποδοχή ομαδικών επισκέψεων.
- *Περιοδικότητα*: Μέσω των boolean πεδίων (Monday, Tuesday, κλπ.), ορίζεται ποιες ημέρες της εβδομάδας εκτελείται το συγκεκριμένο πρόγραμμα.
- *Περίοδος Ισχύος*: Τα πεδία valid_from και valid_until επιτρέπουν στον διαχειριστή να δημιουργεί εποχιακά προγράμματα (π.χ. «Εαρινό Πρόγραμμα 2026»).



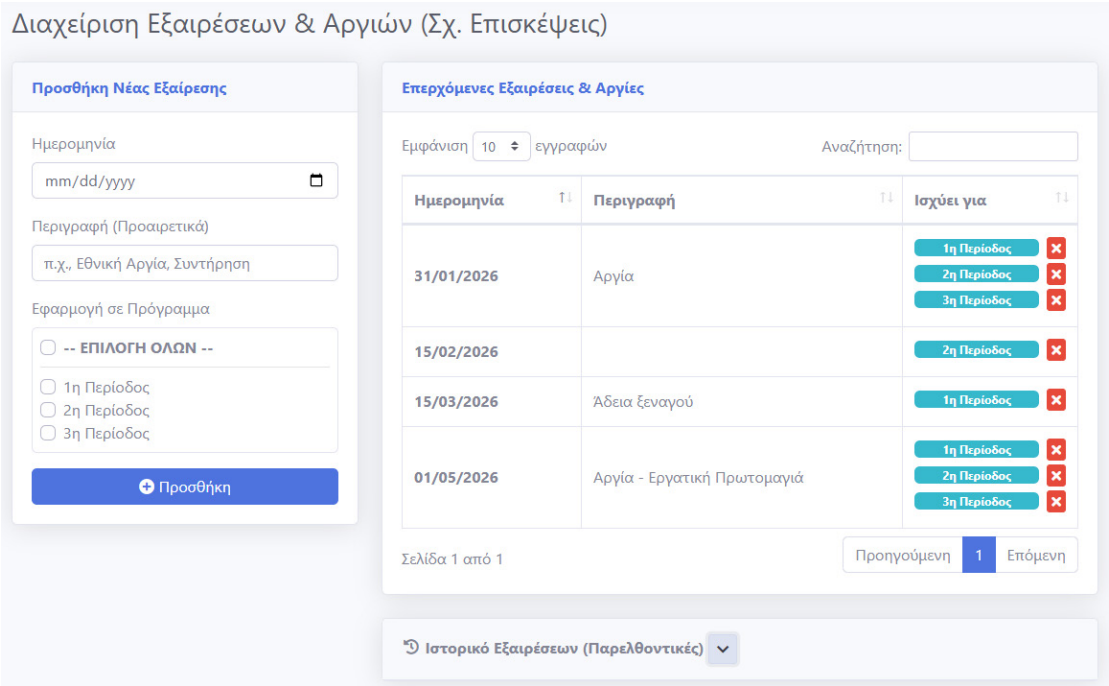
Εικόνα 18. Φόρμα υποβολής νέου προγράμματος για σχολικές επισκέψεις

Για τη διαχείριση των αργιών ή των ημερών που το μουσείο δεν δέχεται σχολεία, υλοποιήθηκε η κλάση `TourException`. Εδώ, δόθηκε έμφαση στην ευχρηστία για τον διαχειριστή. Η μέθοδος `createMultiple` επιτρέπει την ταυτόχρονη εφαρμογή μιας εξαίρεσης σε πολλαπλά προγράμματα επισκέψεων μέσω μίας μόνο ενέργειας,

χρησιμοποιώντας δοσοληψίες στο επίπεδο της βάσης δεδομένων για τη διασφάλιση της ακεραιότητας:

```
$sql = "INSERT INTO tour_exceptions (exception_date, description,
idTour) VALUES (:exception_date, :description, :idTour)";
$stmt = $this->pdo->prepare($sql);


$this->pdo->beginTransaction();
            try {
                //Το $tourIds είναι ένας πίνακας (array) που περιέχει
//τα ID των προγραμμάτων που επέλεξε ο διαχειριστής μέσω των checkboxes
//στη φόρμα
                $tourIds = $data['tours'];

                // Απλά κάνουμε loop για κάθε επιλεγμένο tour ID
                foreach ($tourIds as $tourId) {
                    $stmt->execute([
                        ':exception_date' => $data['exception_date'],
                        ':description' => $data['description'],
                        ':idTour' => (int)$tourId
                    ]);
                }

                $this->pdo->commit();
                return true;
            } catch (PDOException $e) {
                $this->pdo->rollBack();
                flash('danger', 'Παρουσιάστηκε σφάλμα κατά την
αποθήκευση. Πιθανόν κάποια εξαίρεση για αυτή την ημέρα/πρόγραμμα να
υπάρχει ήδη.');
                error_log($e->getMessage());
                return false;
            }
```

Όπως φαίνεται από το παραπάνω κομμάτι κώδικα, η βάση δεδομένων αποτρέπει τις διπλές εγγραφές. Συγκεκριμένα, στον πίνακα tour_exceptions υπάρχει ένας κανόνας μοναδικότητας που συνδυάζει την ημερομηνία (exception_date) με το πρόγραμμα (idTour). Επομένως, είναι αδύνατο να αποθηκευτεί δεύτερη φορά μια εξαίρεση για το ίδιο πρόγραμμα την ίδια ακριβώς μέρα. Με αυτόν τον τρόπο εξασφαλίζεται η ορθότητα των δεδομένων μας, ακόμα κι αν υπάρξει κάποιο σφάλμα από την πλευρά του χρήστη.



Εικόνα 19. Στιγμιότυπο σελίδας διαχείρισης εξαιρέσεων για σχολικές επισκέψεις

Η λογική διαθεσιμότητας (`isTourAvailableOnDate`) ακολουθεί παρόμοια ροή με τις αίθουσες, αλλά εστιάζει στη χρονική εγκυρότητα του slot. Ένα πρόγραμμα θεωρείται διαθέσιμο μόνο αν είναι ενεργό, ισχύει για την τρέχουσα ημέρα της εβδομάδας, βρίσκεται εντός της περιόδου ισχύος του και δεν υπάρχει καταχωρημένη εξαίρεση για τη συγκεκριμένη ημερομηνία.

5.2.7 Κανόνες διαθεσιμότητας και εξαιρέσεις

Ένα από τα πιο κρίσιμα σημεία της εφαρμογής είναι ο αλγόριθμος που αποφασίζει αν μια αίθουσα είναι διαθέσιμη σε μια συγκεκριμένη ημερομηνία. Η λογική αυτή ενθυλακώνεται στη μέθοδο `isVenueAvailableOnDate` του μοντέλου Venue, η οποία εκτελεί ελέγχους σε πέντε επίπεδα:

1. *Έλεγχος Ενεργής Κατάστασης*: Αν η αίθουσα είναι ανενεργή (`active = 0`), αποκλείεται άμεσα.
2. *Έλεγχος Ημέρας Εβδομάδας:* Ελέγχεται αν η αίθουσα λειτουργεί τη συγκεκριμένη ημέρα (π.χ. Κυριακή).
3. *Έλεγχος Περιόδου Ισχύος*: Ελέγχεται αν η ημερομηνία εμπίπτει στο διάστημα που ορίζουν τα πεδία `valid_from` και `valid_until`.
4. *Έλεγχος Εξαιρέσεων (Exceptions):* Το σύστημα ελέγχει τον πίνακα venue_exceptions για:

a. Γενικές αργίες (όπου `idVenue` είναι `NULL`).
b. Ειδικές εξαιρέσεις για τη συγκεκριμένη αίθουσα (π.χ. συντήρηση).

5. *Έλεγχος Υπαρχουσών Κρατήσεων:* Τέλος, ελέγχεται αν υπάρχει ήδη επιβεβαιωμένη ή εκκρεμής κράτηση (`idStatus IN (1, 2)`) στον πίνακα bookVenues.

Σημείωση Υλοποίησης: Στην παρούσα αρχική έκδοση του συστήματος, υιοθετήθηκε η προσέγγιση "First-Come, First-Served" για την απλούστευση της ροής, όπου μία εκκρεμής κράτηση δεσμεύει προσωρινά την αίθουσα για ολόκληρη την ημέρα. Αυτή η επιλογή αποτρέπει προσωρινά άλλους χρήστες από το να αιτηθούν την ίδια ημερομηνία, μειώνοντας την πολυπλοκότητα διαχείρισης ανταγωνιστικών αιτημάτων από τον Venue Manager, ωστόσο αναγνωρίζεται ως σημείο προς εξέλιξη σε επόμενες εκδόσεις.

Μόνο αν περάσει επιτυχώς και τους 5 ελέγχους, η αίθουσα επιστρέφεται ως διαθέσιμη.

```
    /**
     * Ελέγχει ολοκληρωμένα αν μια συγκεκριμένη αίθουσα είναι διαθέσιμη
σε μια συγκεκριμένη ημερομηνία.
     * @param int $idVenue
     * @param string $date (Y-m-d)
     * @return bool
     */
    public function isVenueAvailableOnDate(int $idVenue, string $date):
bool
    {
        // 1. Βρίσκουμε τα στοιχεία της αίθουσας
        $venue = $this->findById($idVenue);
        if (!$venue || !$venue['active']) {
            return false; // Δεν υπάρχει ή δεν είναι ενεργή
        }

        $dt = new DateTime($date);
        $day_of_week = strtolower($dt->format('l'));

        // 2. Έλεγχος ημέρας εβδομάδας
        if (empty($venue[$day_of_week])) {
            return false;
        }

        // 3. Έλεγχος περιόδου ισχύος
        if (($venue['valid_from'] && $date < $venue['valid_from']) ||
($venue['valid_until'] && $date > $venue['valid_until'])) {
```

```
            return false;
        }

        // 4. Έλεγχος για εξαιρέσεις
        // Έλεγχος για γενική αργία (idVenue IS NULL)
        $ex_sql_general = "SELECT 1 FROM venue_exceptions WHERE
exception_date = ? AND idVenue IS NULL LIMIT 1";
        $stmt_ex_general = $this->pdo->prepare($ex_sql_general);
        $stmt_ex_general->execute([$date]);
        if ($stmt_ex_general->fetch()) {
            return false;
        }

        // Έλεγχος για ειδική εξαίρεση για τη συγκεκριμένη αίθουσα
        $ex_sql_specific = "SELECT 1 FROM venue_exceptions WHERE
exception_date = ? AND idVenue = ? LIMIT 1";
        $stmt_ex_specific = $this->pdo->prepare($ex_sql_specific);
        $stmt_ex_specific->execute([$date, $idVenue]);
        if ($stmt_ex_specific->fetch()) {
            return false;
        }

        // 5. Έλεγχος αν είναι ήδη κλεισμένη (pending ή accepted)
        $booking_sql = "SELECT 1 FROM bookVenues WHERE idVenue = ? AND
date = ? AND idStatus IN (1, 2) LIMIT 1";
        $stmt_booking = $this->pdo->prepare($booking_sql);
        $stmt_booking->execute([$idVenue, $date]);
        if ($stmt_booking->fetch()) {
            return false;
        }

        // Αν πέρασε όλους τους ελέγχους, είναι διαθέσιμη!
        return true;
    }
```

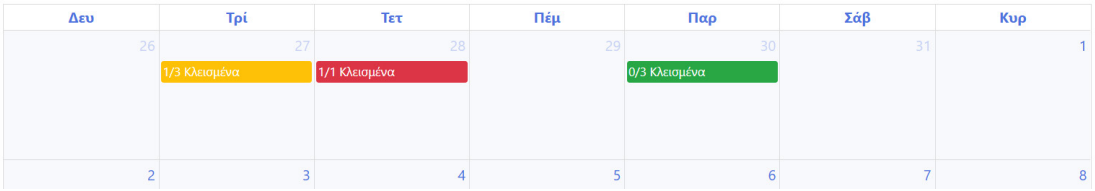
5.2.8 Προβολή και διαχείριση ημερολογίου

Η διαχείριση των κρατήσεων υλοποιήθηκε μέσω ενός διαδραστικού ημερολογίου, το οποίο βασίζεται στη βιβλιοθήκη FullCalendar.js. Για την επικοινωνία του Frontend με το Backend χωρίς την ανάγκη για επαναφόρτωση της σελίδας, αναπτύχθηκε ένα εσωτερικό API.

Α. Αρχιτεκτονική API (ApiController)

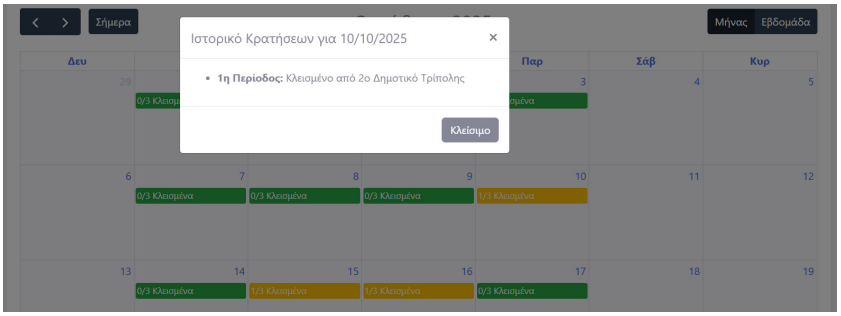
Ο ApiController λειτουργεί ως ενδιάμεσος σταθμός που δέχεται αιτήματα HTTP (GET) και επιστρέφει δεδομένα σε μορφή JSON. Η μέθοδος getVenueEvents (και αντίστοιχα getTourEvents) εκτελεί την εξής ροή εργασίας:

1. *Λήψη Δεδομένων*: Ανακτά τις ενεργές αίθουσες, τις υπάρχουσες κρατήσεις και τις εξαιρέσεις για το ζητούμενο χρονικό εύρος.
2. *Επεξεργασία Λογικής* (Business Logic): Για κάθε ημέρα του μήνα, υπολογίζει τη διαθεσιμότητα.
    a. Αν όλες οι αίθουσες είναι ελεύθερες εμφανίζεται με πράσινο χρώμα.
    b. Αν κάποιες είναι κλεισμένες εμφανίζεται με πορτοκαλί χρώμα.
    c. Αν όλες είναι κατειλημμένες εμφανίζεται με κόκκινο χρώμα.



Εικόνα 20. Χρωματική κωδικοποίηση ημερολογίου

3. *Διαχείριση Δικαιωμάτων*: Το API επιστρέφει διαφορετικά δεδομένα ανάλογα με τον ρόλο του χρήστη.
    a. Στους Διαχειριστικούς Ρόλους (Administrator, Venue Manager, Tour Manager), επιστρέφει αναλυτικά στοιχεία για κάθε κράτηση (ποιος έκλεισε τι, όνομα σχολείου/εταιρείας).
    b. Στους Απλούς Χρήστες (Users), επιστρέφει μόνο την κατάσταση διαθεσιμότητας και αποκρύπτει παρελθοντικές ημερομηνίες ή ημερομηνίες που εμπίπτουν στον κανόνα ελάχιστης προθεσμίας κρατήσεων (6 ημέρες).



Εικόνα 21. Modal για προβολή ιστορικού, στο ημερολόγιο των διαχειριστών

Β. Διαδικασία Κράτησης (BookingController)

Η υποβολή μιας κράτησης γίνεται μέσω της μεθόδου `store`. Πριν την καταχώρηση της εγγραφής στη βάση δεδομένων, εκτελούνται οι παρακάτω έλεγχοι:

1. *Validation*: Έλεγχος πληρότητας στοιχείων.
2. *Profile check*: Έλεγχος αν ο χρήστης έχει συμπληρώσει τα απαραίτητα στοιχεία στο προφίλ του (Σχολείο ή Εταιρεία).
3. *Availability check*: Επαλήθευση διαθεσιμότητας του ζητούμενου πόρου. Ο έλεγχος αυτός είναι διαζευκτικός και εξαρτάται από τον τύπο του αιτήματος. Εάν η κράτηση αφορά αίθουσα, καλείται η σχετική μέθοδος του μοντέλου `Venue`, ενώ εάν αφορά προγράμματα επίσκεψης, χρησιμοποιείται το μοντέλο `Tour`.
4. *Race Condition Prevention*: Έλεγχος "τελευταίας στιγμής" (`doesBookingExist`) για την αποφυγή διπλοεγγραφών, σε περίπτωση που δύο χρήστες προσπαθήσουν να κλείσουν τον ίδιο πόρο ταυτόχρονα.

Αν η κράτηση είναι επιτυχής, το σύστημα:

1. Αποθηκεύει την εγγραφή με κατάσταση *Pending* (για αίθουσες) ή *Accepted* (για tours).
2. Καταγράφει την ενέργεια στο `ActivityLog`.
3. Δημιουργεί μια ειδοποίηση (`BookAlert`) για τον διαχειριστή.

```
    /**
     * Αποθηκεύει μια νέα κράτηση για σχολική επίσκεψη.
     */
    public function storeTourBooking()
    {
        // Έλεγχος δικαιωμάτων
```

```
        if (!is_logged_in()) {
            http_response_code(403);
            exit('Access Denied');
        }

        // --- Validation ---
        $errors = [];
        if (empty($_POST['booking_date'])) $errors[] = "Η ημερομηνία
λείπει.";
        if (empty($_POST['idTour'])) $errors[] = "Πρέπει να επιλέξετε
πρόγραμμα.";
        if (empty($_POST['children']) ||
!is_numeric($_POST['children']) || $_POST['children'] < 1) {
            $errors[] = "Ο αριθμός παιδιών πρέπει να είναι έγκυρος
αριθμός μεγαλύτερος του 0.";
        }
        if (empty($_POST['class'])) $errors[] = "Το πεδίο 'Τάξη' είναι
υποχρεωτικό.";

        $user = $this->userModel->findById($_SESSION['user_id']);
        if (empty($user['school'])) {
            $errors[] = 'Πρέπει να έχετε δηλώσει το σχολείο σας στο
προφίλ σας για να κάνετε κράτηση. <a href="' . url('/profile/edit') .
'">Επεξεργασία Προφίλ</a>';
        }

        if (!empty($errors)) {
            $errorMessage = "<ul>";
            foreach ($errors as $error) { $errorMessage .= "<li>" .
$error . "</li>"; }
            $errorMessage .= "</ul>";
            flash('danger', $errorMessage, true);
            header('Location: ' . url('/tours-calendar'));
            exit();
        }

        // --- ΠΛΗΡΗΣ ΕΛΕΓΧΟΣ ΔΙΑΘΕΣΙΜΟΤΗΤΑΣ ---
        $isAvailable = $this->tourModel-
>isTourAvailableOnDate((int)$_POST['idTour'], $_POST['booking_date']);
        if (!$isAvailable) {
            flash('danger', 'Το πρόγραμμα που επιλέξατε δεν είναι
διαθέσιμο για την συγκεκριμένη ημερομηνία.');
            header('Location: ' . url('/tours-calendar'));
            exit();
        }

        // Έλεγχος σε περίπτωση που κάποιος πρόλαβε να κάνει κράτηση.
```

```
        $isAlreadyBooked = $this->bookTourModel-
>doesBookingExist((int)$_POST['idTour'], $_POST['booking_date']);
        if ($isAlreadyBooked) {
            flash('danger', 'Αυτή η θέση μόλις κρατήθηκε από κάποιον
άλλον. Παρακαλώ επιλέξτε μια άλλη ημέρα ή πρόγραμμα.');
            header('Location: ' . url('/tours-calendar'));
            exit();
        }

        // --- Προετοιμασία Δεδομένων ---
        $data = [
            'idTour'   => (int)$_POST['idTour'],
            'idStatus' => 2,
            'idUser'   => (int)$_SESSION['user_id'],
            'date'     => $_POST['booking_date'],
            'children' => (int)$_POST['children'],
            'class'    => trim($_POST['class'])
        ];

        // --- ΔΗΜΙΟΥΡΓΙΑ ΚΡΑΤΗΣΗΣ ---
        // Η μέθοδος create επιστρέφει το ID της νέας κράτησης ή false.
        $newBookingId = $this->bookTourModel->create($data);

        if ($newBookingId) {
            // Η κράτηση ήταν επιτυχής!
            flash('success', "Η κράτησή σας για τις " . date('d/m/Y',
strtotime($data['date'])) . " καταχωρήθηκε με επιτυχία!");

            // 1. LOGGING
            $this->logModel->logAction(
                'TOUR_BOOKING_CREATE',
                "Ο χρήστης '{$user['username']}' έκανε κράτηση για
σχολική επίσκεψη στις {$data['date']}.",
                $user['idUser']
            );

            // 2. ΔΗΜΙΟΥΡΓΙΑ ΕΙΔΟΠΟΙΗΣΗΣ
            $message = "Νέα σχολική επίσκεψη από το '{$user['school']}'
για τις " . date('d/m/Y', strtotime($data['date'])) . ".";
            $this->bookAlertModel->createForManager($message,
$newBookingId, null); // Χρησιμοποιούμε τη σωστή μέθοδο

        } else {
            // Η κράτηση απέτυχε.
            flash('danger', 'Παρουσιάστηκε ένα σφάλμα. Η κράτηση δεν
καταχωρήθηκε.');
        }
```

```
        header('Location: ' . url('/tours-calendar'));
        exit();
    }
```

5.2.9 Σύστημα Στατιστικών και Αναφορών

Για την υποστήριξη της λήψης αποφάσεων από τη διοίκηση του οργανισμού, υλοποιήθηκε ένα υποσύστημα στατιστικής ανάλυσης. Το σύστημα συγκεντρώνει δεδομένα από τις κρατήσεις και τα παρουσιάζει οπτικοποιημένα.

Α. Συγκέντρωση Δεδομένων

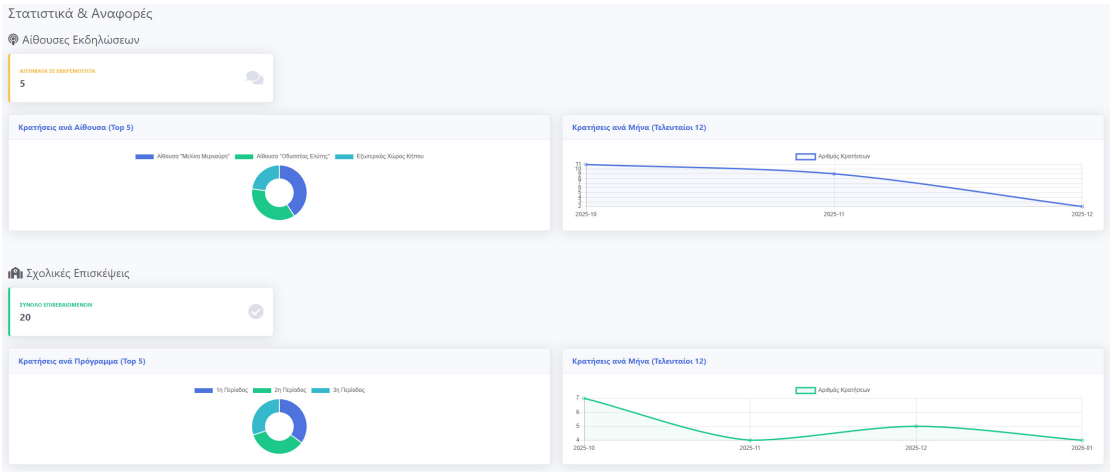
Η λογική της στατιστικής ανάλυσης εκτελείται σε επίπεδο βάσης δεδομένων για βέλτιστη απόδοση. Τα Models (`BookVenue`, `BookTour`) διαθέτουν τη μέθοδο `getStats()`, η οποία χρησιμοποιεί σύνθετα SQL ερωτήματα με συναθροιστικές συναρτήσεις (`COUNT`, `GROUP BY`). Στη συνέχεια παρουσιάζεται ένα παράδειγμα ερωτήματος για τις κρατήσεις αιθουσών εκδηλώσεων ανά ημερολογιακό μήνα για το τελευταίο δωδεκάμηνο.

```
SELECT DATE_FORMAT(date, '%Y-%m') as month, COUNT(*) as count
FROM bookVenues
WHERE idStatus = 2 AND date >= DATE_SUB(CURDATE(), INTERVAL 12 MONTH)
GROUP BY month
ORDER BY month ASC
```

Β. Οπτικοποίηση

Για την παρουσίαση των δεδομένων χρησιμοποιήθηκε η βιβλιοθήκη Chart.js. Ο ελεγκτής `StatsController` μεταβιβάζει τα επεξεργασμένα δεδομένα (JSON) στο κατάλληλο View (`stats/index.php`), όπου η JavaScript αναλαμβάνει να δημιουργήσει δυναμικά:

- *Pie charts* (Γραφήματα Πίτας): Για την απεικόνιση της δημοτικότητας των αιθουσών και των εκπαιδευτικών προγραμμάτων (Top 5).
- *Line charts* (Γραμμικά Γραφήματα): Για την απεικόνιση της τάσης των κρατήσεων τους τελευταίους 12 μήνες (εποχικότητα).



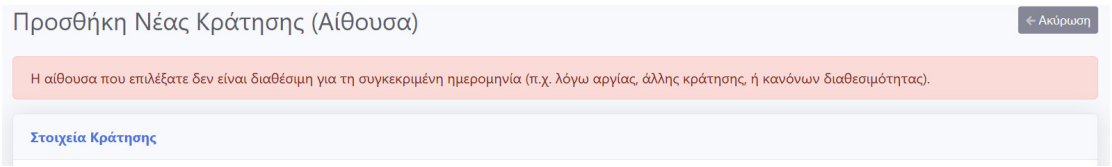
Εικόνα 22. Στιγμιότυπο σελίδας στατιστικών

5.3 Επικύρωση λειτουργικότητας και δοκιμές

Η διασφάλιση της ορθής λειτουργίας του συστήματος επιτεύχθηκε μέσω μιας σειράς δοκιμών που πραγματοποιήθηκαν καθ' όλη τη διάρκεια της ανάπτυξης. Οι δοκιμές εστίασαν σε τρεις άξονες:

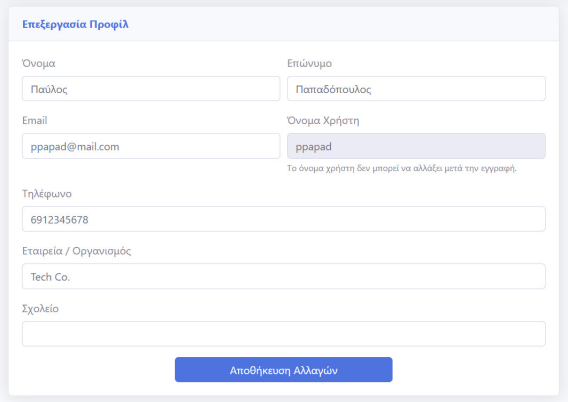
1. Λειτουργικές Δοκιμές

Ελέγχθηκαν όλα τα σενάρια χρήσης. Για παράδειγμα, επιβεβαιώθηκε ότι το σύστημα απαγορεύει την κράτηση αίθουσας αν αυτή είναι ήδη δεσμευμένη.



Εικόνα 23. Μήνυμα σφάλματος για διπλοεγγραφή κράτησης

Ένα άλλο παράδειγμα χρήσης ήταν ο έλεγχος της ροής εγγραφής χρήστη και η ορθή αποθήκευση των στοιχείων του.



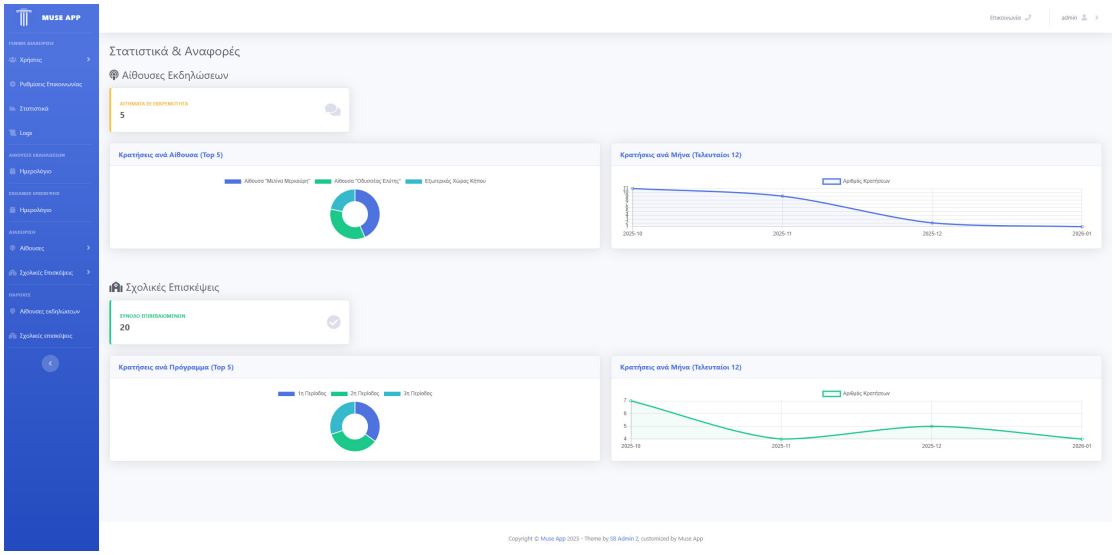
Εικόνα 24. Προβολή νέου χρήστη

12/01/2026 00:39:47	ppapad	LOGOUT_SUCCESS	Ο χρήστης 'ppapad' αποσυνδέθηκε.
12/01/2026 00:39:25	ppapad	LOGIN_SUCCESS	Επιτυχής σύνδεση για τον χρήστη 'ppapad'.
12/01/2026 00:39:13	Σύστημα	USER_REGISTER_SUCCESS	Ένας νέος χρήστης εγγράφηκε με username 'ppapad'.

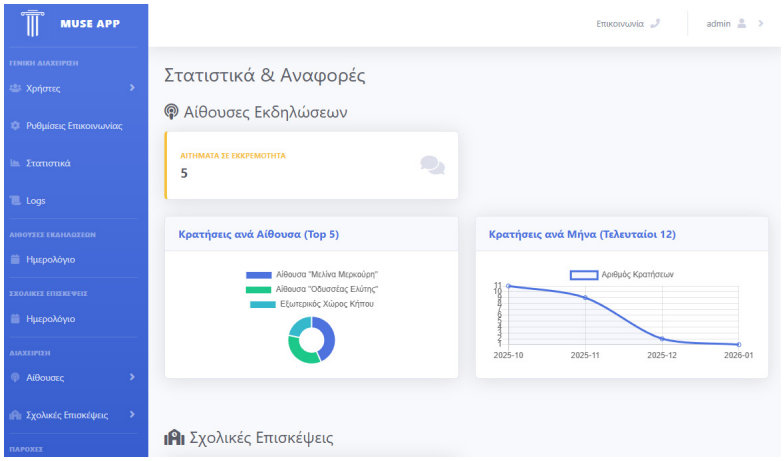
Εικόνα 25. Αποτελέσματα εγγραφής στον πίνακα Logs

2. Δοκιμές Διεπαφής (UI/UX Testing)

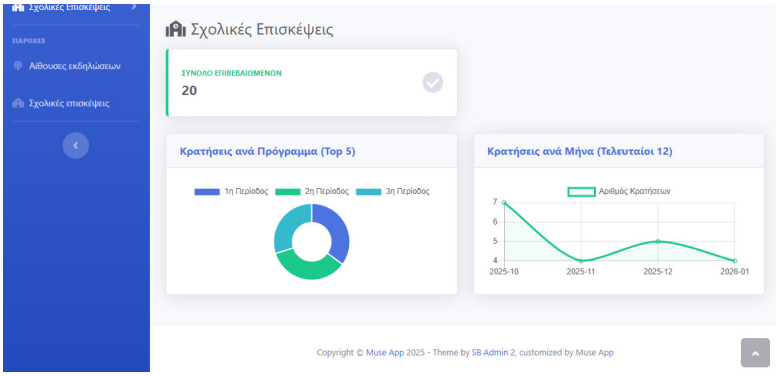
Ελέγχθηκε η ανταπόκριση της εφαρμογής (Responsiveness) σε διαφορετικές αναλύσεις οθόνης (Mobile, Tablet, Desktop) για να διασφαλιστεί ότι το πλαίσιο Bootstrap λειτουργεί σωστά.



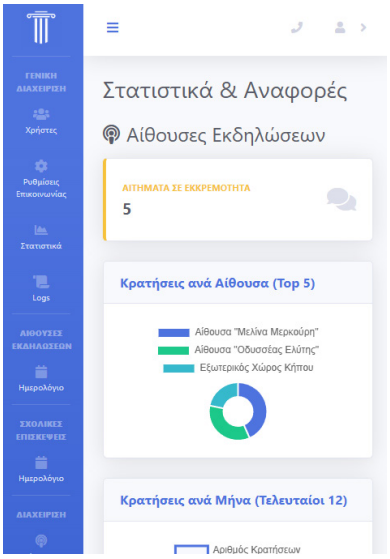
Εικόνα 26. Σελίδα στατιστικών σε προβολή Monitor



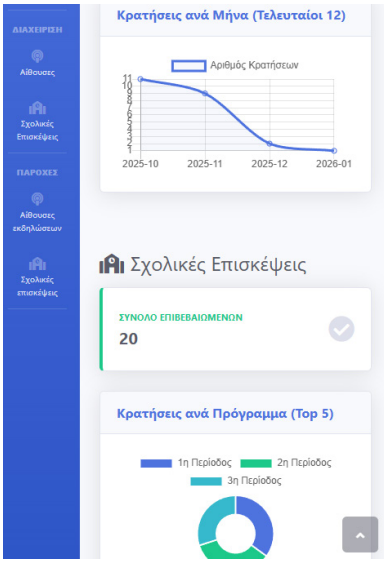
Εικόνα 27. Σελίδα στατιστικών σε προβολή Tablet (1 από 2)



Εικόνα 28. Σελίδα στατιστικών σε προβολή Tablet (2 από 2)



Εικόνα 29. Σελίδα στατιστικών σε προβολή Κινητού (1 από 3)

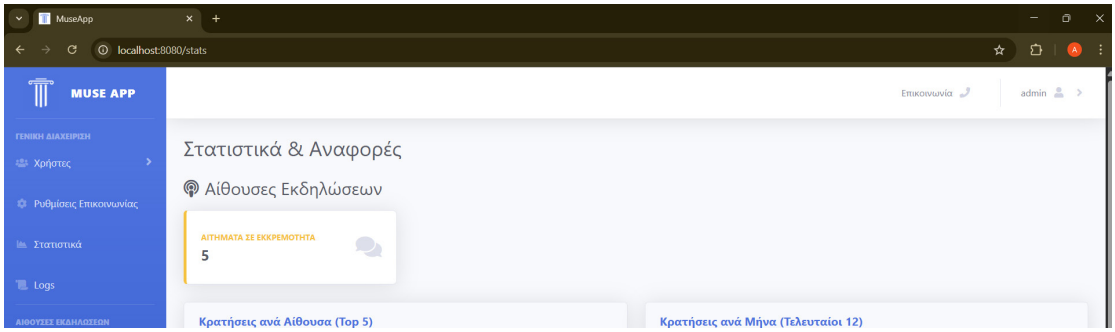


Εικόνα 30. Σελίδα στατιστικών σε προβολή Κινητού (2 από 3)

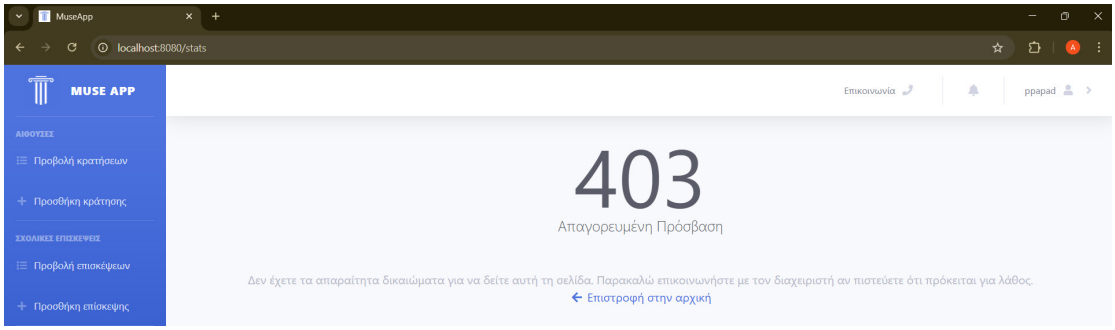
Εικόνα 31. Σελίδα στατιστικών σε προβολή Κινητού (3 από 3)

3. Δοκιμές Ασφαλείας (Security testing):

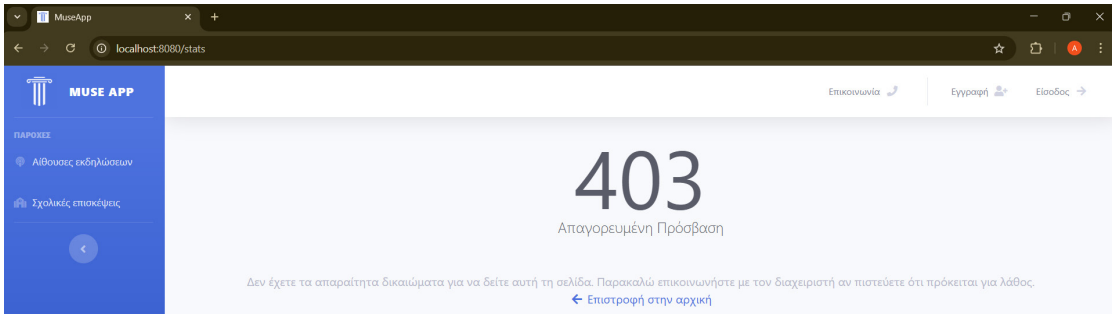
Πραγματοποιήθηκαν απόπειρες μη εξουσιοδοτημένης πρόσβασης ώστε να επιβεβαιωθεί η ορθή λειτουργία του ελέγχου ρόλων. Οι δοκιμές αυτές επικεντρώθηκαν στο επίπεδο της Διεπαφής Χρήστη (UI). Συγκεκριμένα ελέγχθηκε η αποτροπή πλοήγησης σε σελίδες διαχειριστών (μέσω απευθείας πληκτρολόγησης του URL) από μη εξουσιοδοτημένου λογαριασμούς.



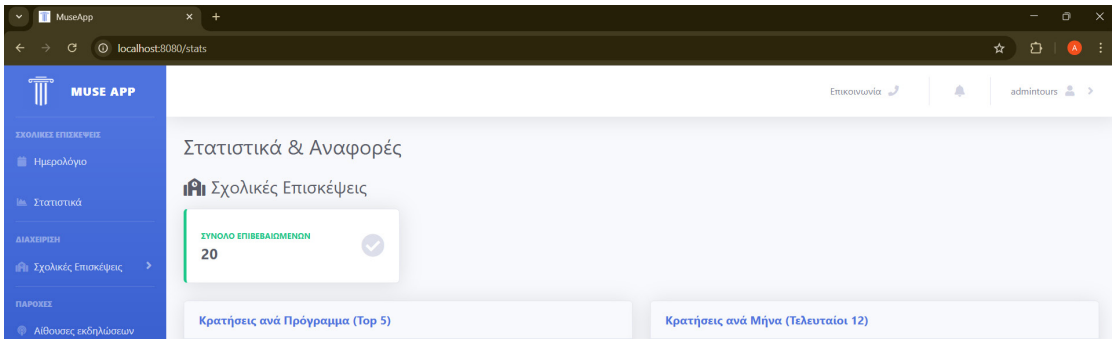
Εικόνα 32. Σελίδα στατιστικών ενώ είναι συνδεδεμένος διαχειριστής



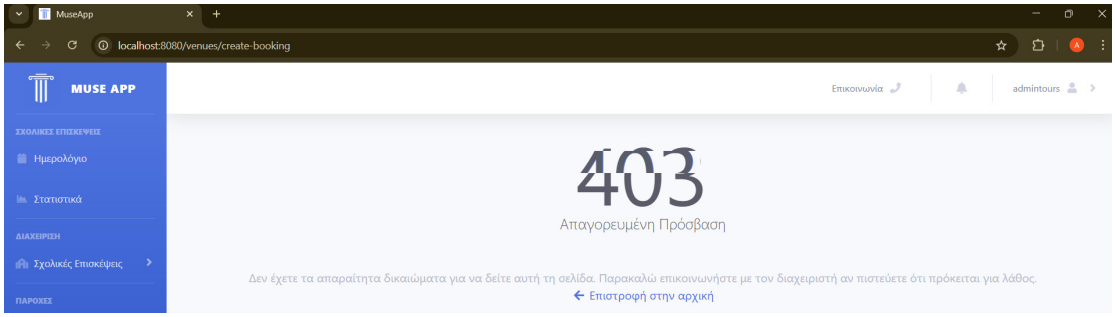
Εικόνα 33. Σελίδα στατιστικών ενώ είναι συνδεδεμένος απλός χρήστης



Εικόνα 34. Σελίδα στατιστικών ενώ δεν είναι συνδεδεμένος χρήστης



Εικόνα 35. Σελίδα στατιστικών ενώ είναι συνδεδεμένος διαχειριστής σχολικών επισκέψεων



Εικόνα 36. Προσπάθεια εισόδου στην φόρμα δημιουργίας κράτησης αίθουσας από διαχειριστή σχολικών επισκέψεων

6 Οδηγίες εγκατάστασης και λειτουργίας

Για να είναι ένα λογισμικό αξιοποιήσιμο, πρέπει να συνοδεύεται από σαφείς οδηγίες εγκατάστασης. Στο κεφάλαιο αυτό περιγράφεται η διαδικασία προετοιμασίας του περιβάλλοντος εκτέλεσης. Χάρη στη χρήση της τεχνολογίας Docker, η διαδικασία έχει απλοποιηθεί σημαντικά, επιτρέποντας την άμεση λειτουργία της εφαρμογής σε οποιοδήποτε σύστημα, ανεξαρτήτως της υποκείμενης υποδομής.

6.1 Περιγραφή δομής Docker Compose (containers, images, services)

Η εφαρμογή αναπτύχθηκε με την αρχή του containerization χρησιμοποιώντας το Docker, εξασφαλίζοντας ένα φορητό, αναπαραγώγιμο και ανεξάρτητο περιβάλλον ανάπτυξης και εκτέλεσης. Η διαχείριση των υπηρεσιών (services) γίνεται μέσω του αρχείου ρύθμισης `compose.yml`. Το σύστημα αποτελείται από δύο βασικές υπηρεσίες, τον διακομιστή ιστού (PHP) και τη βάση δεδομένων MySQL.

Υπηρεσία 1: Web Server (web)

Η υπηρεσία web είναι υπεύθυνη για την παροχή του περιβάλλοντος εντός του οποίου εκτελείται η διαδικτυακή εφαρμογή.

- Δημιουργία Εικόνας: Η εικόνα (image) της υπηρεσίας χτίζεται με βάση το `Dockerfile`, το οποίο ξεκινά από την επίσημη εικόνα `php:8.2-apache`. Στο στάδιο κατασκευής, ενεργοποιούνται κρίσιμες επεκτάσεις της PHP, όπως οι `mysqli` (βασικά στοιχεία σύνδεσης με τη MySQL), `pdo` (PHP Data Objects), και `pdo_mysql` (διεπαφή μεταξύ pdo και MySQL), απαραίτητες για τη σύνδεση με τη βάση δεδομένων. Επιπλέον, ενεργοποιείται το module `rewrite` του Apache, το οποίο είναι απαραίτητο για την εφαρμογή του μοτίβου Model-View-Controller (MVC), επιτρέποντας τη χρήση «καθαρών» (clean) URLs.
- Ρύθμιση Apache: Το προεπιλεγμένο αρχείο ρυθμίσεων του Apache (`000-default.conf`) αντικαθίσταται δυναμικά από το αρχείο `apache.conf`, διασφαλίζοντας ότι το Document Root του διακομιστή ορίζεται στον φάκελο `/var/www/html/app/public`. Αυτό επιβάλλει την πρόσβαση στο σύστημα μόνο μέσω του κεντρικού σημείου εισόδου της εφαρμογής

(`public/index.php`), ενισχύοντας την ασφάλεια και την εφαρμογή του MVC.

- Διασύνδεση: Η θύρα 80 εντός του container αντιστοιχίζεται στη θύρα 8080 του host μηχανήματος (`ports: "8080:80"`).
- Εξαρτήσεις: Εξαρτάται από την υπηρεσία `db` (`depends_on: db`), διασφαλίζοντας ότι ο περιέκτης/διακομιστής της βάσης δεδομένων έχει ξεκινήσει πριν επιχειρήσει να ξεκινήσει η εφαρμογή.

Υπηρεσία 2: Βάση Δεδομένων (db)

Η υπηρεσία `db` χρησιμοποιεί την επίσημη εικόνα `mysql:8.0` και λειτουργεί ως ο αποθηκευτικός χώρος της εφαρμογής.

- Αρχικοποίηση: Μέσω ενός volume mount (`./mysql/schema-data.sql:/docker-entrypoint-initdb.d/init.sql`), το αρχείο SQL με τις εντολές για τη δημιουργία του σχήματος της βάσης δεδομένων και την εισαγωγή των αρχικών δεδομένων εκτελείται αυτόματα κατά την πρώτη εκκίνηση του container, αυτοματοποιώντας την προετοιμασία.
- Κωδικοποίηση: Η κωδικοποίηση ορίζεται ρητά σε `utf8mb4` για πλήρη υποστήριξη ελληνικών χαρακτήρων.
- Δεδομένα: Χρησιμοποιείται ένα named volume (`db_data`) για την μόνιμη αποθήκευση των δεδομένων της βάσης, διασφαλίζοντας ότι αυτά διατηρούνται ακόμα και εάν γίνει επανεκκίνηση στο container ή ακόμη και αν διαγραφεί και ξαναδημιουργηθεί.

6.2 Διαδικασία εγκατάστασης της εφαρμογής

Η εγκατάσταση της εφαρμογής έχει σχεδιαστεί ώστε να είναι όσο το δυνατόν αυτοματοποιημένη. Η διαδικασία ξεκινά με τη μεταφόρτωση του φακέλου του έργου στο περιβάλλον φιλοξενίας. Απαραίτητη προϋπόθεση είναι η δημιουργία του αρχείου μεταβλητών περιβαλλοντικών (`.env`), στο οποίο ορίζονται οι κωδικοί πρόσβασης της βάσης δεδομένων, διασφαλίζοντας ότι τα στοιχεία αυτά δεν εκτίθενται στον κώδικα και μπορούν να συντηρούνται αυτόνομα.

6.3 Εκτέλεση και πρόσβαση στην εφαρμογή

Η εκτέλεση της εφαρμογής βασίζεται στο εργαλείο Docker Compose. Μέσω μιας εντολής τερματικού, το Docker αναλαμβάνει να «κατεβάσει» τις απαραίτητες εικόνες (Images) για την PHP και την MySQL, να δημιουργήσει το ιδεατό δίκτυο επικοινωνίας μεταξύ τους και να εκκινήσει τις υπηρεσίες. Μετά την επιτυχή εκκίνηση, η εφαρμογή είναι προσβάσιμη μέσω οποιουδήποτε φυλλομετρητή στη διεύθυνση localhost (ή στην IP του server), στη θύρα που έχει οριστεί.

6.4 Προετοιμασία βάσης δεδομένων και αρχικών δεδομένων

Ένα από τα πλεονεκτήματα της αρχιτεκτονικής που επιλέχθηκε είναι η αυτόματη προετοιμασία της βάσης δεδομένων. Κατά την πρώτη εκκίνηση του container της MySQL, εκτελείται αυτόματα το αρχείο `schema-data.sql`. Αυτό το αρχείο δημιουργεί τη δομή των πινάκων (tables) και των συσχετίσεων (foreign keys).

Εισάγει τα απαραίτητα αρχικά δεδομένα, όπως τους ρόλους χρηστών (roles) και τον λογαριασμό του Διαχειριστή (`admin`). Με αυτόν τον τρόπο, το σύστημα είναι έτοιμο προς χρήση αμέσως μετά την εγκατάσταση, χωρίς να απαιτείται χειροκίνητη παρέμβαση στη βάση.

6.5 Διαχείριση μέσω GitHub και ενημερώσεις

Ο πηγαίος κώδικας της εφαρμογής φιλοξενείται σε αποθετήριο (repository) στο GitHub. Αυτό επιτρέπει τη συνεχή ενσωμάτωση νέων λειτουργιών. Η ενημέρωση της εγκατάστασης σε νεότερη έκδοση γίνεται απλά με τη λήψη των αλλαγών (`git pull`) και την επανεκκίνηση των containers, ώστε να ενσωματωθούν οι αλλαγές στον κώδικα ή στη βάση δεδομένων.

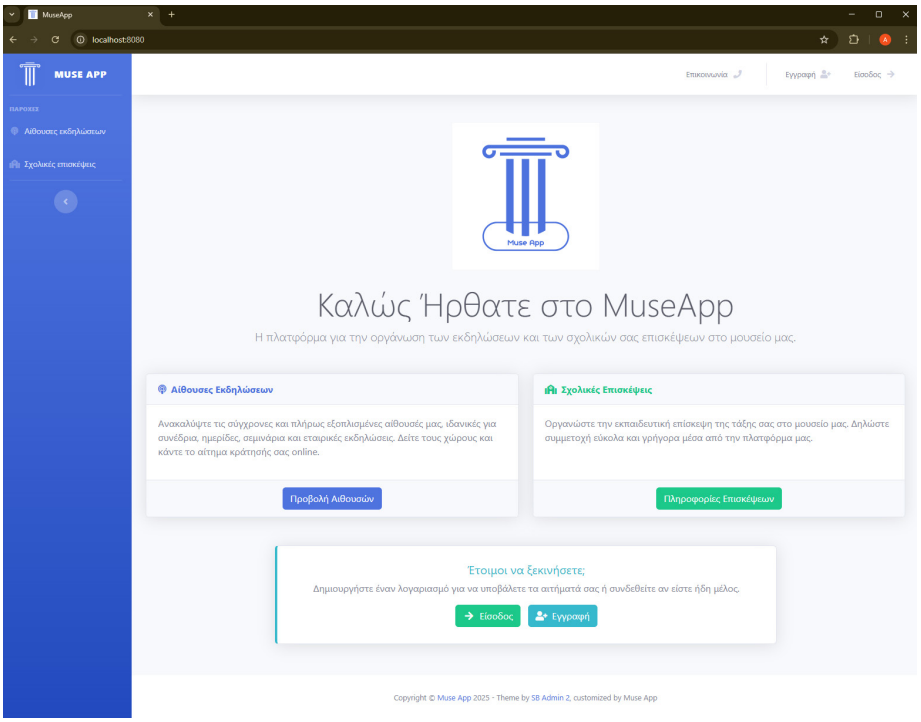
7 Παρουσίαση Λειτουργιών του Συστήματος

Στο κεφάλαιο αυτό παρουσιάζεται το τελικό αποτέλεσμα της εργασίας μέσω της οπτικής διεπαφής της εφαρμογής. Μέσα από στιγμιότυπα οθόνης και περιγραφές ροών εργασίας, αναδεικνύονται οι δυνατότητες του συστήματος τόσο για τον απλό επισκέπτη όσο και για τον διαχειριστή του πολιτιστικού οργανισμού.

7.1 Επισκόπηση διεπαφής χρήστη

7.1.1 Αρχική Σελίδα (Landing Page)

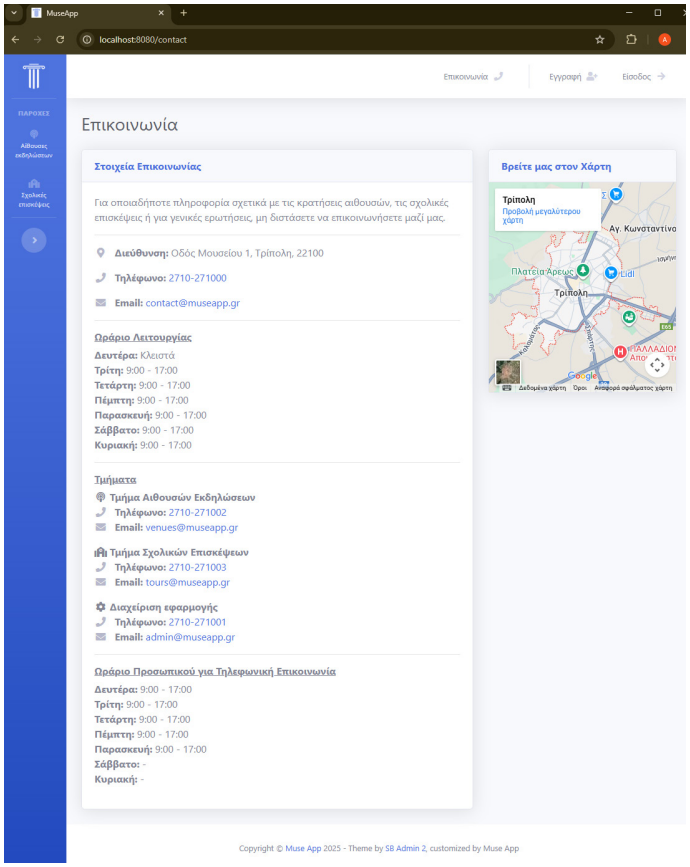
Η αρχική σελίδα (`home.php`) αποτελεί το σημείο υποδοχής των επισκεπτών. Σχεδιάστηκε με έμφαση στην καθαρότητα και την άμεση καθοδήγηση του χρήστη («Call to Action»). Περιλαμβάνει έναν κεντρικό τομέα για το λογότυπο και σύντομη περιγραφή. Επιπλέον υπάρχουν δύο διακριτές κάρτες που κατευθύνουν τον χρήστη στις κύριες υπηρεσίες «Αίθουσες Εκδηλώσεων» και «Σχολικές Επισκέψεις». Τέλος έχει μια ενότητα προτροπής για Είσοδο/Εγγραφή, η οποία εμφανίζεται δυναμικά μόνο σε μη συνδεδεμένους χρήστες.



Εικόνα 37. Αρχική σελίδα

7.1.2 Σελίδα Επικοινωνίας και Πληροφοριών

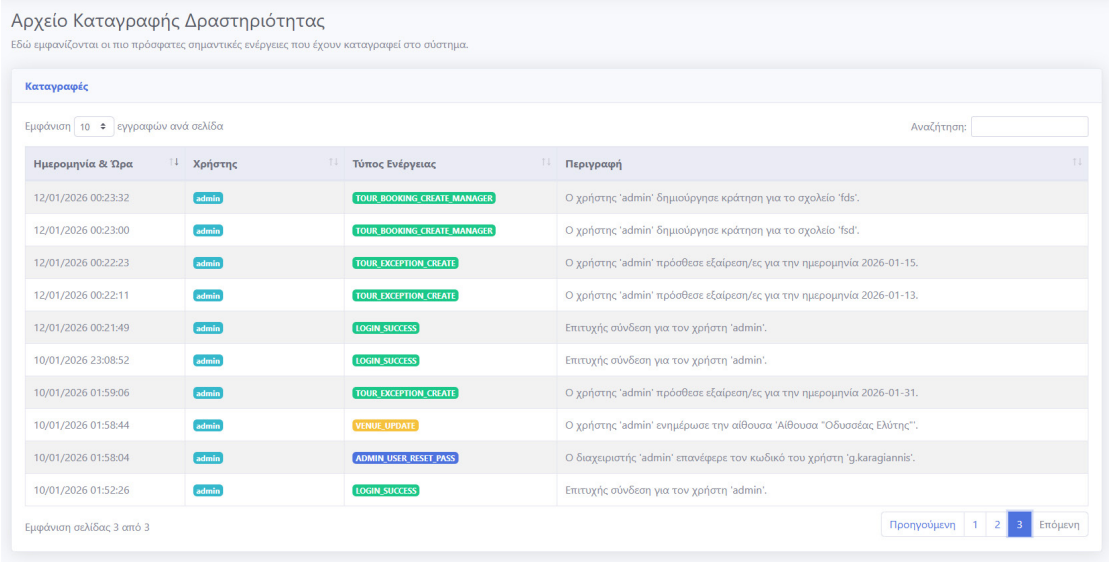
Η σελίδα επικοινωνίας είναι πλήρως δυναμική. Όλα τα στοιχεία που εμφανίζονται (Διεύθυνση, Τηλέφωνα ανά τμήμα, Email, Ωράρια Λειτουργίας Κοινού και Προσωπικού, καθώς και ο διαδραστικός χάρτης Google Maps) ελέγχονται από τον Διαχειριστή μέσω της φόρμας «Ρυθμίσεις Ιστοσελίδας». Αυτό επιτρέπει στον οργανισμό να ενημερώνει άμεσα το κοινό για αλλαγές στο ωράριο (π.χ. θερινό - χειμερινό) ή στα στοιχεία επικοινωνίας, διασφαλίζοντας την εγκυρότητα της πληροφορίας.



Εικόνα 38. Σελίδα επικοινωνίας

7.1.3 Παρακολούθηση Δραστηριότητας (Logs)

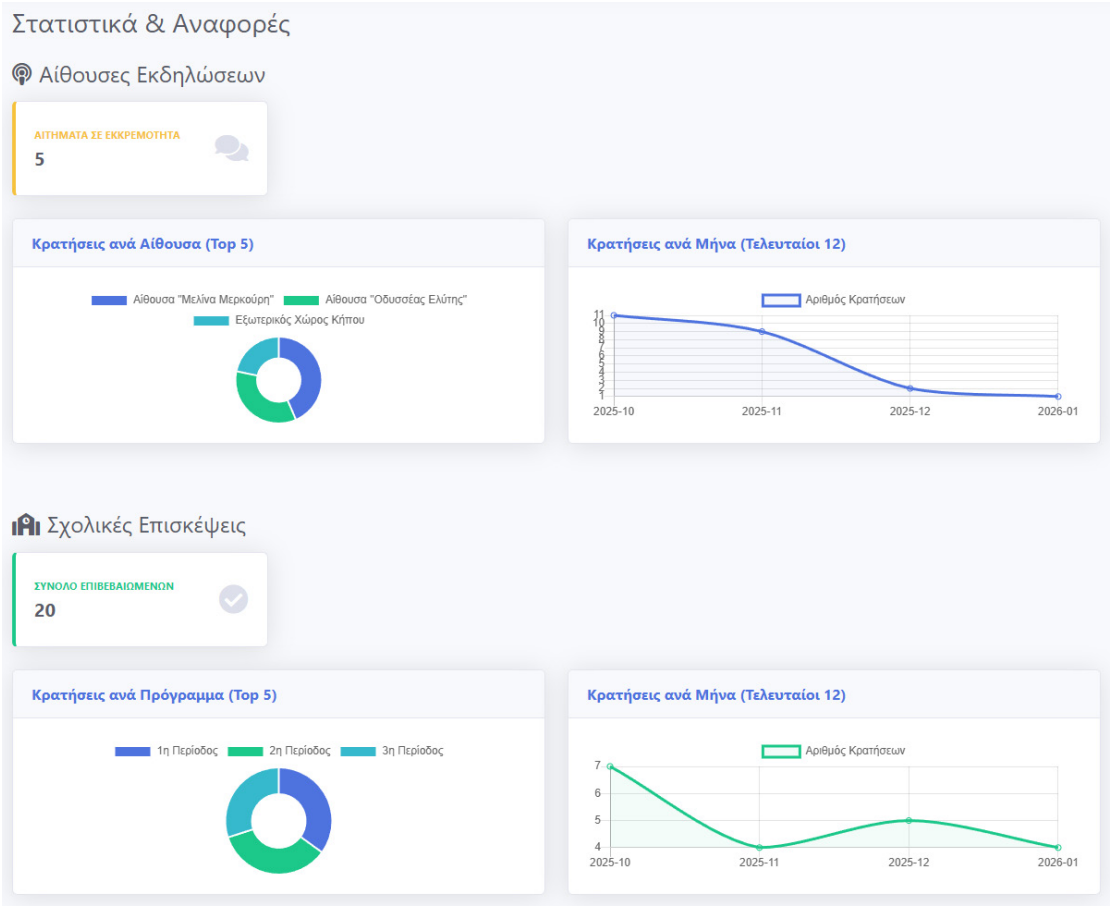
Η σελίδα «Αρχείο Καταγραφής Δραστηριότητας» είναι προσβάσιμη αποκλειστικά από τον Διαχειριστή του συστήματος. Παρέχει μια χρονολογική λίστα όλων των κρίσιμων ενεργειών που έχουν συμβεί στην πλατφόρμα. Μέσω του ενσωματωμένου μηχανισμού αναζήτησης (`DataTables`), ο διαχειριστής μπορεί να φιλτράρει τα αποτελέσματα για να βρει ενέργειες συγκεκριμένου χρήστη ή συγκεκριμένης ημερομηνίας, διευκολύνοντας τον έλεγχο σε περίπτωση λανθασμένων ενεργειών.



Εικόνα 39. Σελίδα καταγραφής δραστηριότητας (Logs)

7.1.4 Πίνακας Ελέγχου και Στατιστικά (Dashboard)

Η σελίδα Στατιστικών αποτελεί το κέντρο ελέγχου για τους διαχειριστές. Ανάλογα με τον ρόλο του χρήστη (Venue Manager, Tour Manager ή Super Admin), το σύστημα προσαρμόζει δυναμικά το περιεχόμενο. Ο διαχειριστής ενημερώνεται άμεσα με κάρτες για βασικούς δείκτες απόδοσης, όπως το πλήθος των εκκρεμών αιτημάτων και το σύνολο των επιβεβαιωμένων κρατήσεων. Επιπλέον παρέχεται η οπτική απεικόνιση της απόδοσης των χώρων και των προγραμμάτων, βοηθώντας στον εντοπισμό των πιο δημοφιλών υπηρεσιών.



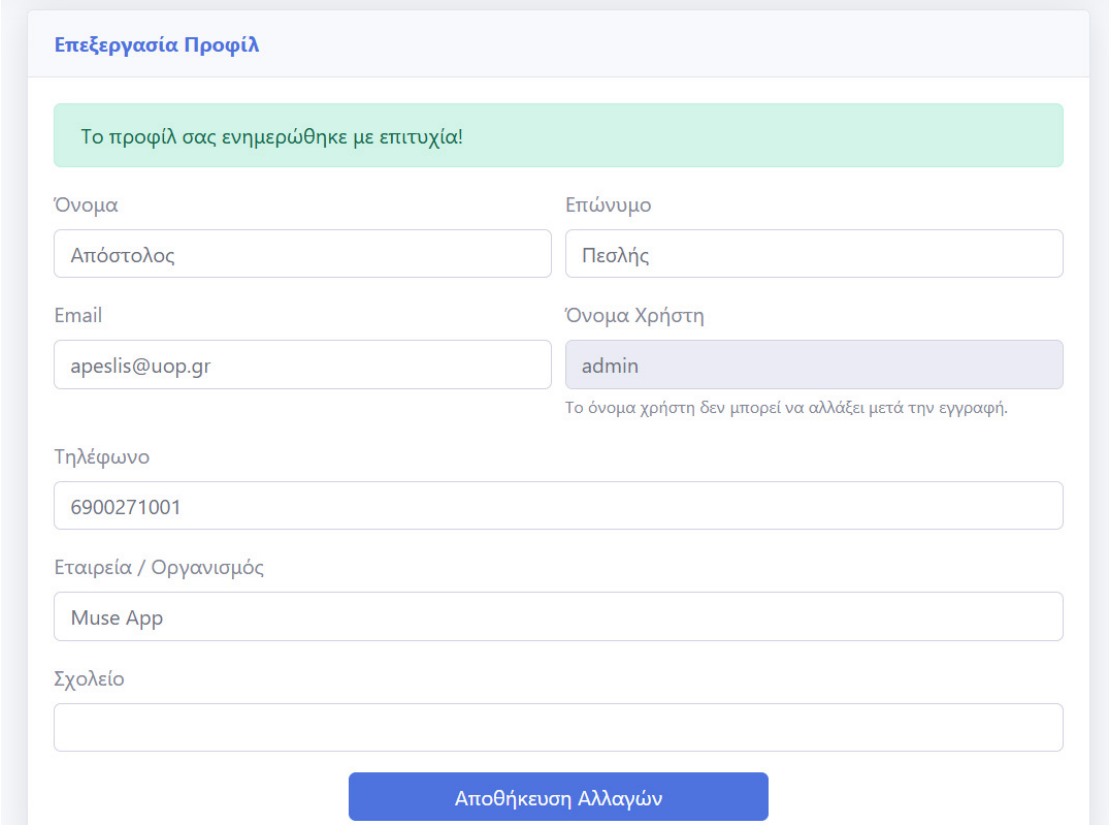
Εικόνα 40. Σελίδα στατιστικών

7.1.5 Πίνακας Ελέγχου Χρήστη (User Dashboard)

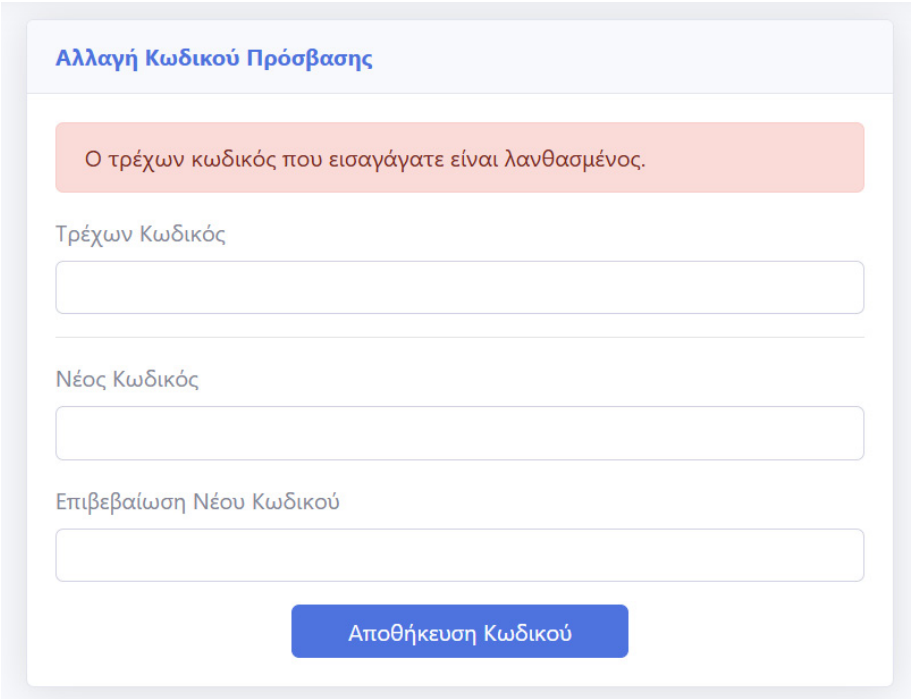
Κάθε εγγεγραμμένος χρήστης έχει πρόσβαση σε ένα προσωπικό περιβάλλον διαχείρισης. Από εκεί μπορεί:

- Να δει και να επεξεργαστεί τα προσωπικά του στοιχεία.
- Να αλλάξει τον κωδικό πρόσβασής του.
- Να δηλώσει τον οργανισμό (Εταιρεία ή Σχολείο) που εκπροσωπεί, στοιχείο απαραίτητο για την πραγματοποίηση κρατήσεων.

Η διεπαφή παρέχει άμεση ανατροφοδότηση (Flash Messages) για την επιτυχία ή αποτυχία των ενεργειών, βελτιώνοντας την αλληλεπίδραση.



Εικόνα 41. Σελίδα επεξεργασίας προφίλ



Εικόνα 42. Σελίδα αλλαγής κωδικού πρόσβασης

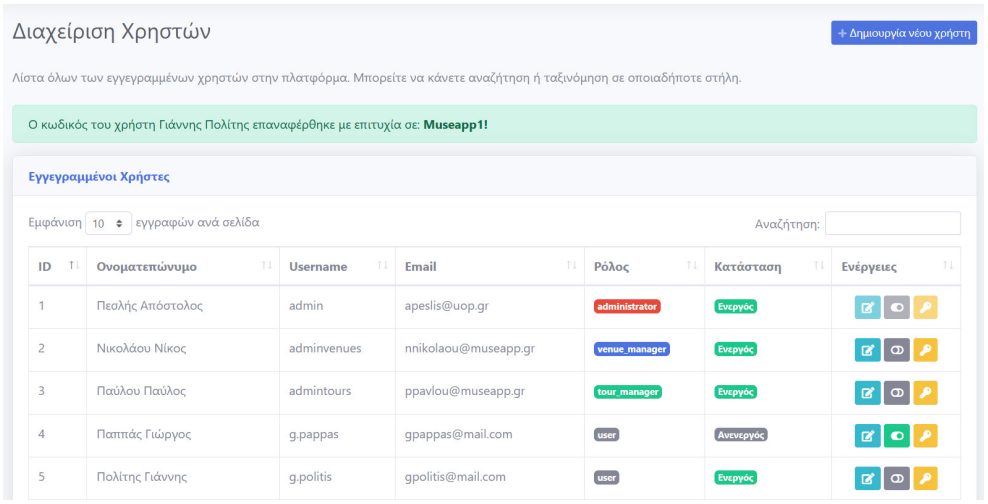
7.2 Διαχείριση χρηστών και ρόλων

Η ενότητα της διαχείρισης χρηστών είναι προσβάσιμη μόνο στους χρήστες που έχουν τον ρόλο του «Administrator» (Super Admin). Μέσω αυτής της διεπαφής, ο διαχειριστής έχει την πλήρη εποπτεία όλων των λογαριασμών που έχουν πρόσβαση στο σύστημα, είτε πρόκειται για μέλη του προσωπικού είτε για εξωτερικούς χρήστες.

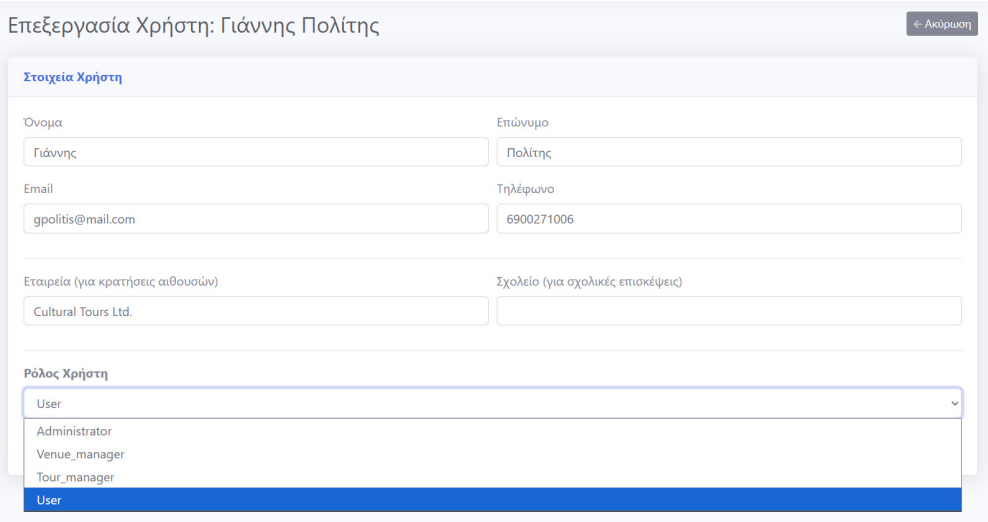
Κατά την είσοδο στη σελίδα «Χρήστες», εμφανίζεται ένας δυναμικός πίνακας (Data Table) που περιλαμβάνει όλους τους εγγεγραμμένους χρήστες. Για κάθε εγγραφή εμφανίζονται βασικά στοιχεία όπως το ονοματεπώνυμο, το email και το όνομα χρήστη. Επιπλέον, ιδιαίτερη έμφαση δίνεται στην οπτική αποτύπωση της κατάστασης του χρήστη. Αρχικά ο ρόλο εμφανίζεται με ευδιάκριτη ετικέτα (badge), επιτρέποντας στον διαχειριστή να ξεχωρίζει γρήγορα τους Venue Managers, Tour Mangers και τους απλούς Users. Επίσης οι ενεργοί χρήστες επισημαίνονται με πράσινο χρώμα, ενώ οι απενεργοποιημένοι με γκρι, διευκολύνοντας τον εντοπισμό λογαριασμών που έχουν αποκλειστεί.

Επιπλέον των παραπάνω δίνονται στον διαχειριστή τρεις (3) βασικές λειτουργίες που μπορεί να εκτελέσει για κάθε υφιστάμενο χρήστη.

- *Επεξεργασία*: Αλλαγή προσωπικών στοιχείων ή ρόλου.
- *Απενεργοποίηση* (Ban): Με το πάτημα ενός διακόπτη (Toggle switch), ο διαχειριστής μπορεί να απενεργοποιήσει άμεσα την πρόσβαση ενός χρήστη στο σύστημα, χωρίς να διαγράψει τα ιστορικά δεδομένα των κρατήσεών του.
- *Επαναφορά Κωδικού* (Password Reset): Σε περίπτωση που ένας χρήστης ξεχάσει τον κωδικό του, ο διαχειριστής μπορεί να ορίσει έναν νέο, προσωρινό κωδικό πρόσβασης.



Εικόνα 43. Σελίδα διαχείρισης χρηστών

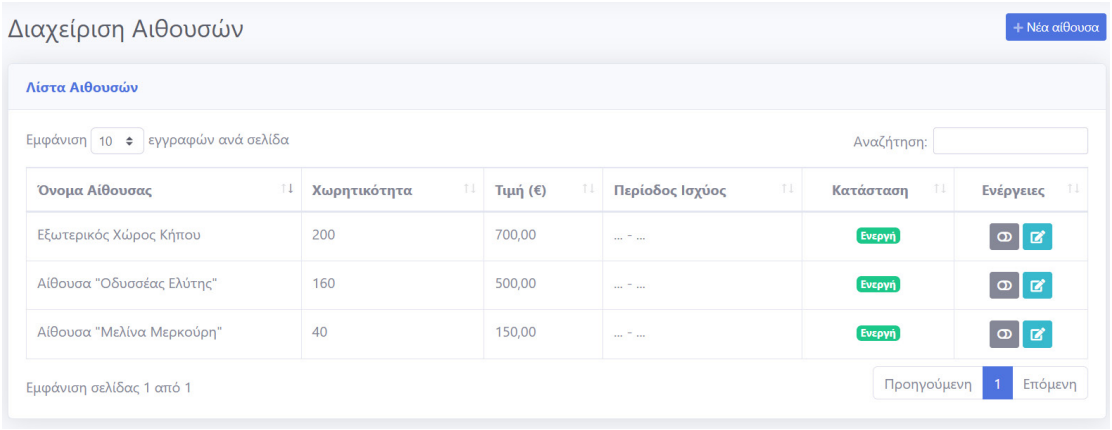


Εικόνα 44. Σελίδα επεξεργασίας χρήστη

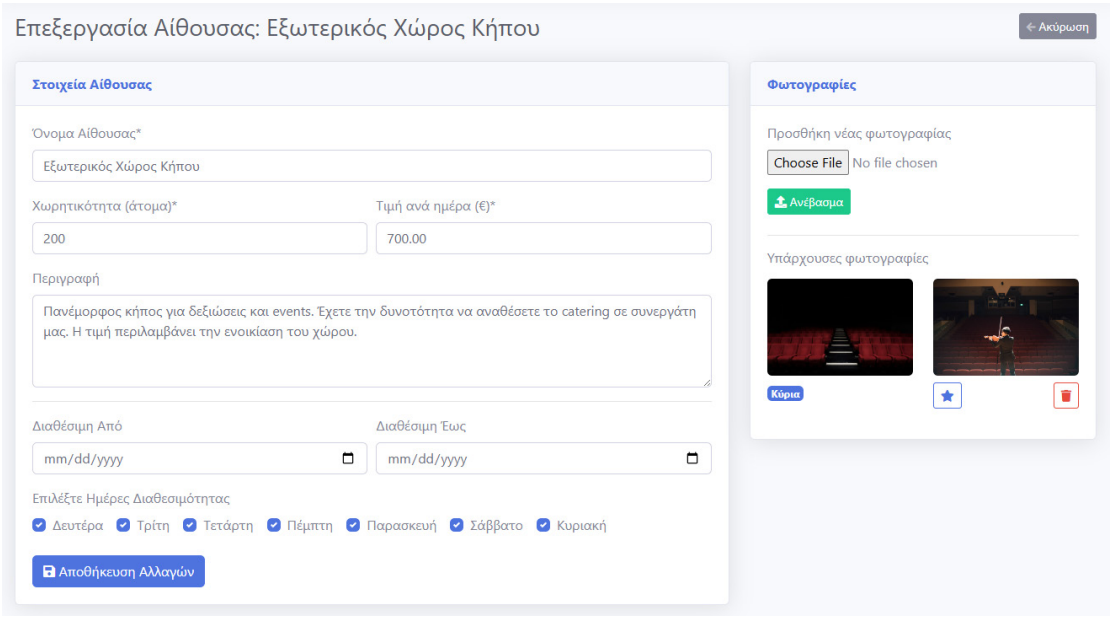
7.3 Διαχείριση αιθουσών και σχολικών επισκέψεων

Α. Προβολή διαχείρισης αιθουσών εκδηλώσεων

Στο περιβάλλον διαχείρισης, ο διαχειριστής αιθουσών εκδηλώσεων έχει πρόσβαση σε έναν πίνακα ελέγχου όπου μπορεί να δει την κατάσταση κάθε αίθουσας. Παρέχονται δυνατότητες για την άμεση ενεργοποίηση/απενεργοποίηση με το πάτημα ενός κουμπιού ή την επεξεργασία μίας αίθουσας.



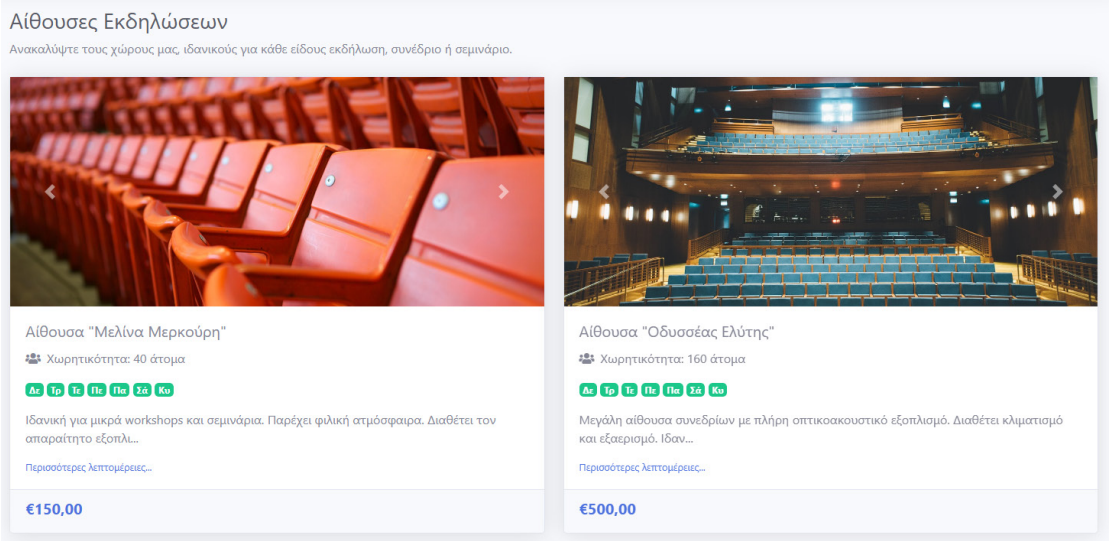
Εικόνα 45. Σελίδα διαχείρισης αιθουσών εκδηλώσεων



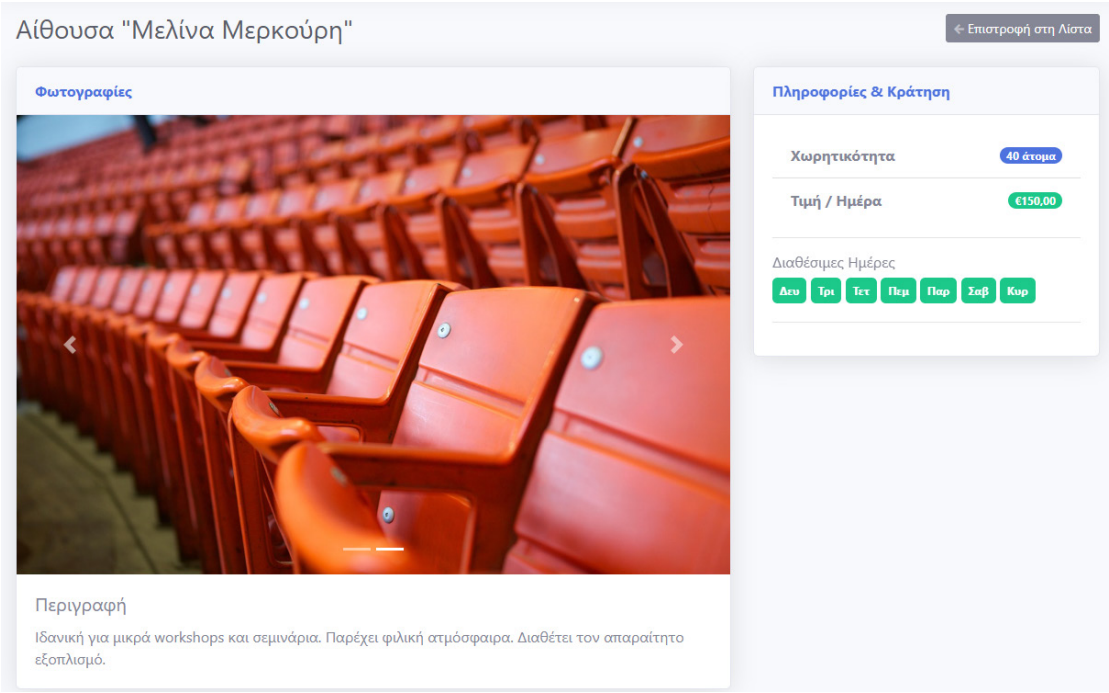
Εικόνα 46. Σελίδα επεξεργασίας αίθουσας εκδηλώσεων

Β. Δημόσια προβολή αιθουσών εκδηλώσεων

Για τον επισκέπτη, οι αίθουσες παρουσιάζονται σε μορφή καρτών. Κάθε κάρτα περιλαμβάνει δυναμικό Slider φωτογραφιών με τη μορφή Carousel, εικονίδια για τη χωρητικότητα και badges που δείχνουν με μια ματιά τις ημέρες λειτουργίας της αίθουσας. Αν ο χρήστης είναι συνδεδεμένος, γίνεται προτροπή να προχωρήσει στο Ημερολόγιο για κράτηση.



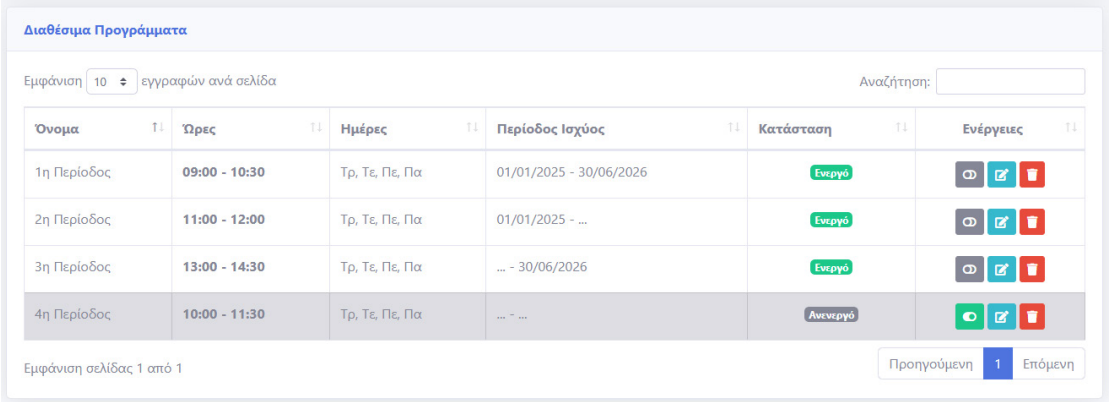
Εικόνα 47. Σελίδα προβολής αιθουσών εκδηλώσεων



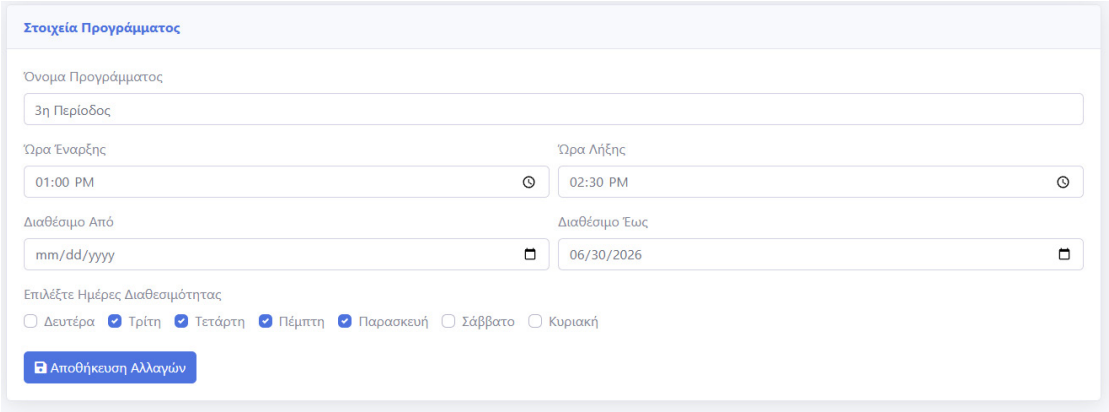
Εικόνα 48. Σελίδα προβολής λεπτομερειών αίθουσας εκδηλώσεων

Γ. Προβολή διαχείρισης επισκέψεων

Ο διαχειριστής επισκέψεων έχει στη διάθεσή του εργαλεία για τον καθορισμό των προγραμμάτων κρατήσεων για ομαδικές επισκέψεις. Η διεπαφή, πέρα από την προβολή βασικών στοιχείων (όνομα προγράμματος, ώρες και ημέρες που πραγματοποιείται και περίοδος ισχύος) επιτρέπει τη γρήγορη επισκόπηση των ενεργών προγραμμάτων με χρωματική κωδικοποίηση. Επιπλέον παρέχει βασικές λειτουργίες όπως την εύκολη απενεργοποίηση ενός προγράμματος, την επεξεργασία και την διαγραφή.



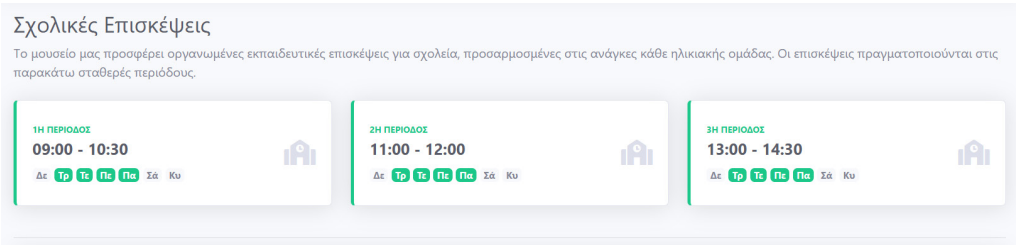
Εικόνα 49. Σελίδα διαχείρισης προγραμμάτων σχολικών επισκέψεων



Εικόνα 50. Σελίδα επεξεργασίας προγράμματος σχολικών επισκέψεων

Δ. Δημόσια προβολή επισκέψεων

Η σελίδα «Σχολικές Επισκέψεις» απευθύνεται σε εκπαιδευτικούς ή εν γένει σε οργανωτές ομαδικών επισκέψεων. Τα διαθέσιμα προγράμματα εμφανίζονται ως κάρτες πληροφοριών που αναγράφουν με σαφήνεια τις ώρες έναρξης και λήξης, καθώς και τις ημέρες που πραγματοποιούνται.

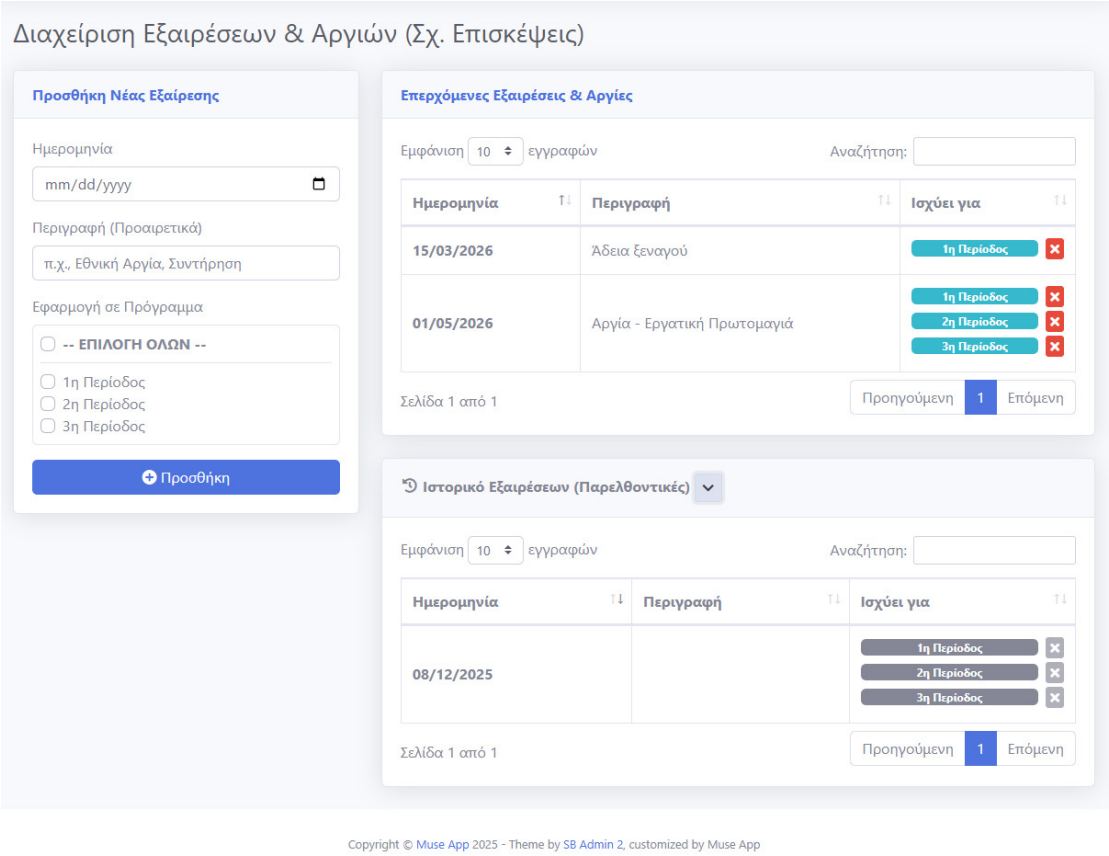


Εικόνα 51. Σελίδα προβολής επισκέψεων

Αν ο χρήστης δεν είναι συνδεδεμένος, το σύστημα τον προτρέπει να πραγματοποιήσει Είσοδο/Εγγραφή για να προχωρήσει σε κράτηση, αποκρύπτοντας το κουμπί του ημερολογίου για την αποφυγή άσκοπων ενεργειών.

7.4 Κανόνες διαθεσιμότητας και εξαιρέσεις

Η οθόνη διαχείρισης εξαιρέσεων παρέχει στον διαχειριστή απόλυτο έλεγχο στο ημερολόγιο του οργανισμού. Μέσω μιας απλής φόρμας, ο διαχειριστής μπορεί να επιλέξει μια ημερομηνία και να την ορίσει ως «Μη Διαθέσιμη» για συγκεκριμένες αίθουσες ή προγράμματα. Η διεπαφή παρέχει τη δυνατότητα μαζικής επιλογής, επιτρέποντας, για παράδειγμα, το κλείσιμο όλων των αιθουσών κατά τις επίσημες αργίες με μία μόνο κίνηση. Επιπλέον αυτής της φόρμας, παρουσιάζονται δύο πίνακες (Επερχόμενες και Παρελθοντικές Εξαιρέσεις) για την εύκολη επισκόπηση και διαγραφή των κανόνων.

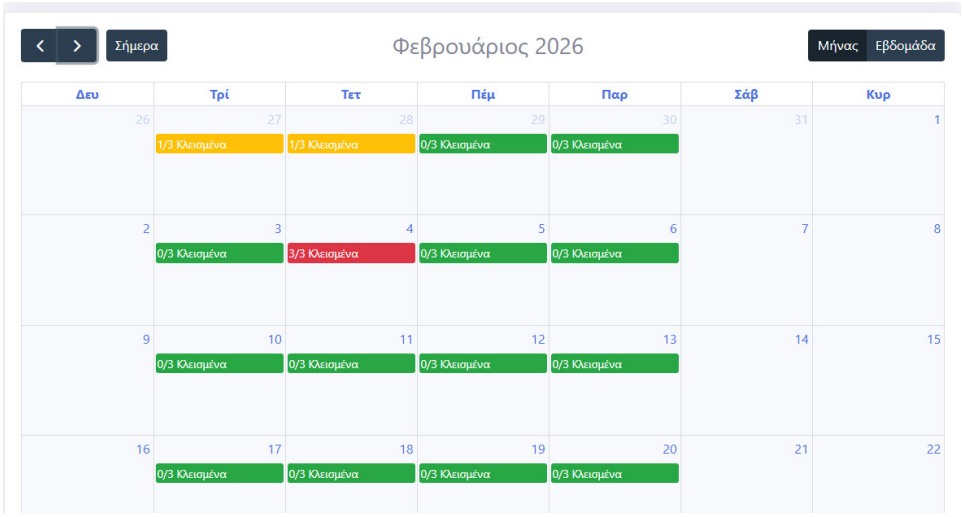


Εικόνα 52. Σελίδα διαχείρισης εξαιρέσεων σχολικών επισκέψεων

7.5 Ημερολόγιο κρατήσεων

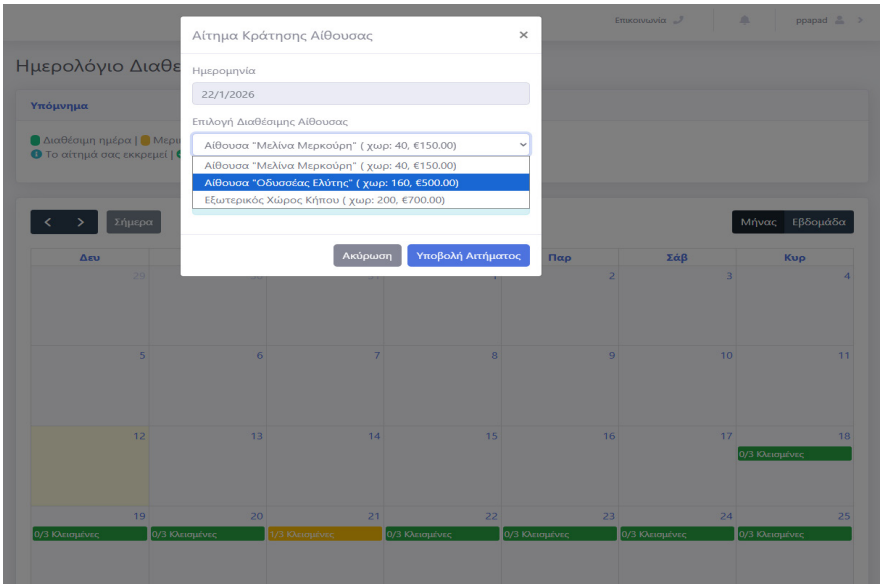
Το ημερολόγιο αποτελεί το κεντρικό σημείο αλληλεπίδρασης του χρήστη με την εφαρμογή. Ο επισκέπτης βλέπει μια χρωματικά κωδικοποιημένη επισκόπηση του μήνα. Κάνοντας κλικ σε μια "πράσινη" ή "πορτοκαλί" ημέρα, ανοίγει ένα αναδυόμενο

παράθυρο (Modal). Το Modal φορτώνει δυναμικά τη λίστα με τις συγκεκριμένες διαθέσιμες αίθουσες ή προγράμματα επισκέψεων για εκείνη την ημέρα.

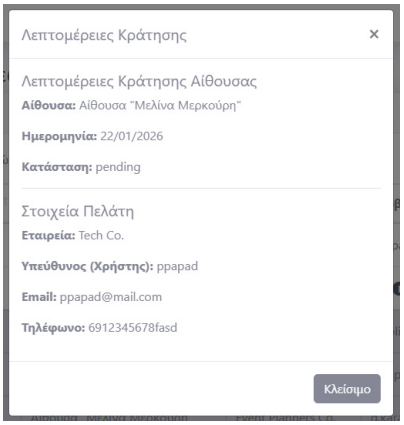


Εικόνα 53. Σελίδα ημερολογίου σχολικών επισκέψεων

Επιπλέον η φόρμα συμπληρώνεται αυτόματα με τα στοιχεία του χρήστη, ελαχιστοποιώντας τον χρόνο καταχώρησης.



Εικόνα 54. Υποβολή αιτήματος κράτησης αίθουσας εκδηλώσεων



Εικόνα 55. Αποτελέσματα αιτήματος κράτησης αίθουσας εκδηλώσεων

Ο διαχειριστής διαθέτει την πλήρη εικόνα. Κάνοντας κλικ σε οποιαδήποτε ημέρα ακόμα και παρελθοντική, βλέπει μια αναλυτική λίστα με το ποιοι έχουν κάνει κράτηση, την κατάσταση της κράτησης και τα στοιχεία επικοινωνίας τους, επιτρέποντας την άμεση διαχείριση.

8 ΕΠΙΛΟΓΟΣ

Στο πλαίσιο της παρούσας διπλωματικής εργασίας δημιουργήθηκε ένα πλήρες και λειτουργικό πληροφοριακό σύστημα, ικανό να καλύψει τις σύγχρονες ανάγκες διαχείρισης ενός πολιτιστικού οργανισμού. Στο κεφάλαιο αυτό παρουσιάζονται τα συμπεράσματα που προέκυψαν από την υλοποίηση, καθώς και προτάσεις για τη μελλοντική επέκταση της εφαρμογής.

8.1 Συμπεράσματα

Ο πρωταρχικός στόχος της εργασίας, ήταν ο σχεδιασμός και η υλοποίηση της διαδικτυακής εφαρμογής «MuseApp» για την αυτοματοποίηση των κρατήσεων. Μέσα από την ανάλυση, τον σχεδιασμό και την προγραμματιστική υλοποίηση, προέκυψαν τα παρακάτω συμπεράσματα.

Αρχικά, το σύστημα επιλύει υπαρκτά προβλήματα των μουσείων/πολιτιστικών οργανισμών. Η εφαρμογή αντιμετωπίζει αποτελεσματικά τα προβλήματα της χειρόγραφης διαχείρισης κρατήσεων/προγραμματισμού επισκέψεων. Ο μηχανισμός ελέγχου διαθεσιμότητας σε πραγματικό χρόνο εξάλειψε τον κίνδυνο διπλοεγγραφών, ενώ η ψηφιοποίηση της διαδικασίας μείωσε δραστικά τον διοικητικό φόρτο του προσωπικού.

Επίσης σημαντική επιλογή ήταν η χρήση της Αρχιτεκτονικής MVC. Η υιοθέτηση του προτύπου Model - View - Controller, χωρίς τη χρήση έτοιμου framework, αποδείχθηκε χρήσιμη για την εμπέδωση καλών πρακτικών σχεδίου λογισμικού και τεχνικά αποδοτική. Η επιλογή αυτή οδήγησε σε καθαρό, δομημένο και συντηρήσιμο κώδικα, όπου η λογική της εφαρμογής είναι πλήρως διαχωρισμένη από τη διεπαφή χρήστη.

Αναδείχθηκε η σημασία της Εμπειρίας Χρήστη. Η ενσωμάτωση εργαλείων όπως το `FullCalendar.js` και η χρήση AJAX για ασύγχρονη επικοινωνία αναβάθμισαν σημαντικά την εμπειρία χρήστη, καθιστώντας την εφαρμογή γρήγορη και φιλική, τόσο για τους διαχειριστές όσο και για το κοινό.

Τέλος χαρακτηριστική είναι η ευελιξία που παρέχεται μέσω του Docker. Η χρήση τεχνολογιών containerization εξασφάλισε ότι η εφαρμογή μπορεί να μεταφερθεί και να

εγκατασταθεί σε οποιοδήποτε περιβάλλον με ελάχιστες ρυθμίσεις, λύνοντας το πρόβλημα της συμβατότητας εκδόσεων λογισμικού.

Συνολικά, το σύστημα αποτελεί μια στιβαρή βάση που εκσυγχρονίζει τη λειτουργία του οργανισμού και βελτιώνει την εξυπηρέτηση των επισκεπτών.

8.2 Δυνατότητες Μελλοντικής Επέκτασης

Λαμβάνοντας υπόψη τη δυναμική φύση της τεχνολογίας και τις αυξανόμενες απαιτήσεις των πολιτιστικών φορέων, προτείνονται οι ακόλουθες επεκτάσεις που θα μπορούσαν να ενσωματωθούν σε μελλοντικές εκδόσεις της εφαρμογής.

- *Online Πληρωμές*: Διασύνδεση με τραπεζικά συστήματα ή πύλες πληρωμών για την εξόφληση του αντιτίμου της κράτησης, μετατρέποντας την εφαρμογή σε πλήρες σύστημα.
- *Ειδοποιήσεις μέσω Email*: Επέκταση του υπάρχοντος συστήματος ειδοποιήσεων ώστε να αποστέλλονται αυτόματα emails στους χρήστες για την επιβεβαίωση της κράτησης ή για υπενθυμίσεις πριν από την ημερομηνία της επίσκεψης.
- *Βελτίωση Μηχανισμού Ασφαλείας Κωδικών*: Η τρέχουσα υλοποίηση κατά τη δημιουργία χρήστη ή επαναφορά κωδικού από τον διαχειριστή χρησιμοποιεί έναν προκαθορισμένο αρχικό κωδικό. Για την ενίσχυση της ασφάλειας, σε μελλοντική έκδοση κρίνεται σκόπιμη η ανάπτυξη ενός μηχανισμού δημιουργίας τυχαίων κωδικών ή η ενσωμάτωση υπηρεσίας αποστολής email. Μέσω αυτής, ο χρήστης θα λαμβάνει έναν μοναδικό, κρυπτογραφημένο σύνδεσμο (reset link) μιας χρήσης, ώστε να ορίζει αποκλειστικά ο ίδιος τον προσωπικό του κωδικό πρόσβασης.
- *Διαχείριση Ανταγωνιστικών Αιτημάτων*: Αλλαγή της λογικής διαθεσιμότητας ώστε οι «εκκρεμείς» (pending) κρατήσεις να μην δεσμεύουν την αίθουσα. Πολλοί χρήστες θα μπορούν να αιτηθούν την ίδια ημερομηνία και η έγκριση μιας αίτησης από τον διαχειριστή θα απορρίπτει αυτόματα (auto-reject) τις υπόλοιπες ανταγωνιστικές αιτήσεις, ενημερώνοντας τους χρήστες με αντίστοιχο αιτιολογικό απόρριψης.

- *Κρατήσεις βάσει Χρονοθυρίδων*: Αντί για ολοήμερη δέσμευση μίας αίθουσας, το σύστημα θα επιτρέπει κρατήσεις μικρότερης διάρκειας. Αυτό θα αυξήσει τη διαθεσιμότητα και την αποδοτικότητα χρήσης των πόρων του οργανισμού.
- *Βελτιστοποίηση Αρχιτεκτονικής και Αποφυγή Ταυτόχρονων Εγγραφών*: Στην παρούσα έκδοση, η αποτροπή διπλοεγγραφών γίνεται μέσω απλών ερωτημάτων ελέγχου διαθεσιμότητας. Ωστόσο, αναγνωρίζεται η πιθανότητα εμφάνισης Ταυτοχρόνων Εγγραφών (Race Conditions) σε συνθήκες υψηλού φόρτου. Σε μελλοντική έκδοση, προβλέπεται η αναδιαμόρφωση του κώδικα των Controllers με χρήση αυστηρών τύπων δεδομένων και η ενσωμάτωση όλης της λογικής ελέγχου/εισαγωγής εντός Δοσοληψιών Βάσης Δεδομένων (Database Transactions).
- *Υποστήριξη πολυγλωσσικού περιβάλλοντος*: Προσθήκη μηχανισμού διεθνοποίησης ώστε η εφαρμογή να είναι προσβάσιμη σε τουρίστες και ξενόγλωσσους επισκέπτες ή να μπορεί να χρησιμοποιηθεί σε πολιτιστικούς οργανισμούς άλλων χωρών.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] «https://owasp.org/,» Open Worldwide Application Security Project, [Ηλεκτρονικό]. Available: https://owasp.org/www-project-top-ten/. [Πρόσβαση 28 Απρ 2026].

[2] «https://db-engines.com,» DB-Engines, [Ηλεκτρονικό]. Available: https://db-engines.com/en/ranking. [Πρόσβαση 28 Απρ 2026].

[3] MySQL, «mysql.com,» MySQL, [Ηλεκτρονικό]. Available: https://dev.mysql.com/doc/refman/8.0/en/. [Πρόσβαση 4 Ιαν 2026].

[4] R. Elmasri και S. Navathe, Fundamentals of Database Systems, Boston: Pearson, 2016.

[5] L. Welling και L. Thomson, PHP and MySQL Web Development, Hoboken: Addison-Wesley, 2016.

[6] «php.net,» PHP, [Ηλεκτρονικό]. Available: https://www.php.net/manual/en/history.php.php. [Πρόσβαση 10 Ιαν 2026].

[7] T. Butler, PHP & MySQL Novice to Ninja, SitePoint, 2022.

[8] «php.net,» PHP, [Ηλεκτρονικό]. Available: https://www.php.net/manual/en/book.mysqlnd.php. [Πρόσβαση 29 Απρ 2026].

[9] PHP, «php.net,» PHP, [Ηλεκτρονικό]. Available: https://www.php.net/manual/en/book.pdo.php. [Πρόσβαση 12 Ιαν 2026].

[10] B. McLaughlin, PHP & MySQL: The Missing Manual, O'Reilly, 2015.

[11] A. Freeman, Pro ASP.NET Core MVC 2, Apress, 2017.

[12] Bootstrap, «getbootstrap.com,» Bootstrap, [Ηλεκτρονικό]. Available: https://getbootstrap.com/docs/4.0/getting-started/introduction/. [Πρόσβαση 15 Ιαν 2025].

[13] FullCalendar, «fullcalendar.io,» FullCalendar, [Ηλεκτρονικό]. Available: https://fullcalendar.io/docs. [Πρόσβαση 15 Ιαν 2026].

[14] Docker, «docker.com,» Docker, [Ηλεκτρονικό]. Available: https://www.docker.com/why-docker/. [Πρόσβαση 15 Δεκ 2025].

[15] S. Johnston, «docker.com,» 21 Μαρ 2024. [Ηλεκτρονικό]. Available: https://www.docker.com/blog/docker-11-year-anniversary/. [Πρόσβαση 15 Δεκ 2025].

[16] Docker, «docker.com,» Docker, [Ηλεκτρονικό]. Available: https://www.docker.com/resources/what-container/. [Πρόσβαση 15 Δεκ 2025].

[17] D. Jones, «netapp.com,» NetApp, 16 Μαρ 2018. [Ηλεκτρονικό]. Available: https://www.netapp.com/blog/containers-vs-vms/. [Πρόσβαση 28 Απρ 2026].

[18] Docker, «docker.com,» Docker, [Ηλεκτρονικό]. Available: https://docs.docker.com/get-started/docker-overview/. [Πρόσβαση 15 Δεκ 2025].

[19] D. Merkel, «Docker: lightweight Linux containers for consistent development and deployment,» *Linux Journal*, p. 2, 1 Μαρ 2014.

[20] «linuxcontainers.org,» [Ηλεκτρονικό]. Available: https://linuxcontainers.org/. [Πρόσβαση 29 Απρ 2026].

[21] «containerd.io/,» [Ηλεκτρονικό]. Available: https://containerd.io/. [Πρόσβαση 29 Απρ 2026].

[22] S. Hykes, «docker.com,» Docker, 22 Ιουν 2015. [Ηλεκτρονικό]. Available: https://www.docker.com/blog/runc/. [Πρόσβαση 29 Απρ 2026].

[23] W. Wang, «Application of Docker Container Technology in University Information Center,» σε *International Conference on Network and Information Systems for Computers (ICNISC)* , Guiyang, China, 2021.

[24] A. Peslis, «MuseApp,» 2025. [Ηλεκτρονικό]. Available: https://github.com/apospes/museapp. [Πρόσβαση 2026].

[25] The Power MBA, «thepowermba.com,» The Power MBA, 14 Απρ 2025. [Ηλεκτρονικό]. Available: https://www.thepowermba.com/en/blog/what-is-the-model-view-controller-pattern-and-how-does-it-work. [Πρόσβαση 9 Ιαν 2026].

ΠΑΡΑΡΤΗΜΑ Α’: Docker Configuration Files

Το Παράρτημα αυτό περιλαμβάνει τα βασικά αρχεία ρύθμισης που χρησιμοποιούνται για την κατασκευή και την εκτέλεση της εφαρμογής μέσω Docker.

Α.1 Αρχείο Ρύθμισης Υπηρεσιών (compose.yml)

```
version: '3.8'
services:
  web:
    build: ./php
    container_name: museapp_web
    ports:
      - "8080:80"
    volumes:
      - ./php:/var/www/html
      - ./php/apache.conf:/etc/apache2/sites-available/000-default.conf
    depends_on:
      - db
    environment:
      MYSQL_DATABASE: ${MYSQL_DATABASE}
      MYSQL_USER: ${MYSQL_USER}
      MYSQL_PASSWORD: ${MYSQL_PASSWORD}

  db:
    image: mysql:8.0
    container_name: museapp_db
    restart: unless-stopped
    ports:
      - "3306:3306"
    environment:
      MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
      MYSQL_DATABASE: ${MYSQL_DATABASE}
      MYSQL_USER: ${MYSQL_USER}
      MYSQL_PASSWORD: ${MYSQL_PASSWORD}
      TZ: Europe/Athens
      MYSQL_INITDB_SKIP_TZINFO: "true"
    command: --character-set-server=utf8mb4 --collation-
server=utf8mb4_unicode_ci --skip-character-set-client-handshake
    volumes:
      - db_data:/var/lib/mysql
      - ./mysql/schema-data.sql:/docker-entrypoint-initdb.d/init.sql

volumes:
  db_data:
```

A.2 Αρχείο Δημιουργίας Εικόνας Web Server (Dockerfile)

```
FROM php:8.2-apache

# Ενεργοποιούμε τις απαραίτητες PHP επεκτάσεις
RUN docker-php-ext-install mysqli pdo pdo_mysql

# Ενεργοποιούμε το module του Apache για τα "καθαρά" URLs
RUN a2enmod rewrite
```

A.3 Αρχείο Ρύθμισης Apache (apache.conf)

```
<VirtualHost *:80>
    DocumentRoot /var/www/html/app/public
    <Directory /var/www/html/app/public>
        Options Indexes FollowSymLinks
        AllowOverride All
        Require all granted
    </Directory>
    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

ΠΑΡΑΡΤΗΜΑ Β': Σχεδίαση Βάσης Δεδομένων

Ακολουθεί ο κώδικας SQL για τη δημιουργία του σχήματος της βάσης δεδομένων (`schema-data.sql`).

```sql
DROP DATABASE IF EXISTS musedb;
CREATE DATABASE musedb;
USE musedb;

-- -----------------
-- TABLES
-- -----------------

-- Roles
CREATE TABLE roles (
    idRole INT AUTO_INCREMENT PRIMARY KEY,
    roleName VARCHAR(50) NOT NULL
);

-- Users
CREATE TABLE users (
    idUser INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(100) NOT NULL UNIQUE,
    firstname VARCHAR(100) NOT NULL,
    lastname VARCHAR(100) NOT NULL,
    password VARCHAR(255) NOT NULL,
    phone VARCHAR(15) NOT NULL,
    mail VARCHAR(200) NOT NULL UNIQUE,
    company VARCHAR(255),
    school VARCHAR(255),
    idRole INT NOT NULL DEFAULT 4,
    active INT NOT NULL DEFAULT 1,
    FOREIGN KEY (idRole) REFERENCES roles(idRole)
);

-- Settings
CREATE TABLE settings (
    setting_key VARCHAR(50) PRIMARY KEY,
    setting_value VARCHAR(1000)
);

-- Activity_logs
CREATE TABLE activity_logs (
    idLog INT AUTO_INCREMENT PRIMARY KEY,
    idUser INT,
    action_type VARCHAR(255) NOT NULL,
    action_description VARCHAR(1000),
```

```
    timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (idUser) REFERENCES users(idUser)
);

-- Venues
CREATE TABLE venues (
    idVenue INT AUTO_INCREMENT PRIMARY KEY,
    venueName VARCHAR(255) NOT NULL UNIQUE,
    capacity INT NOT NULL,
    description VARCHAR(1000) NOT NULL,
    price DECIMAL(10,2) NOT NULL,
    monday INT NOT NULL DEFAULT 1,
    tuesday INT NOT NULL DEFAULT 1,
    wednesday INT NOT NULL DEFAULT 1,
    thursday INT NOT NULL DEFAULT 1,
    friday INT NOT NULL DEFAULT 1,
    saturday INT NOT NULL DEFAULT 1,
    sunday INT NOT NULL DEFAULT 1,
    active INT NOT NULL DEFAULT 1,
    valid_from DATE NULL,
    valid_until DATE NULL
);

CREATE TABLE venue_exceptions (
    idException INT AUTO_INCREMENT PRIMARY KEY,
    exception_date DATE NOT NULL,
    description VARCHAR(255) NULL,
    idVenue INT NULL,
    FOREIGN KEY (idVenue) REFERENCES venues(idVenue) ON DELETE CASCADE
);

-- Photo
CREATE TABLE photo (
    idPhoto INT AUTO_INCREMENT PRIMARY KEY,
    photoPath VARCHAR(255) NOT NULL,
    idVenue INT,
    is_primary BOOLEAN NOT NULL DEFAULT FALSE,
    FOREIGN KEY (idVenue) REFERENCES venues(idVenue)
);

-- Tours
CREATE TABLE tours (
    idTour INT AUTO_INCREMENT PRIMARY KEY,
    tourName VARCHAR(50) NOT NULL UNIQUE,
    tourTimeStart TIME,
    tourTimeFinish TIME,
    monday INT,
    tuesday INT,
```

```
    wednesday INT,
    thursday INT,
    friday INT,
    saturday INT,
    sunday INT,
    active INT NOT NULL DEFAULT 1,
    valid_from DATE NULL,
    valid_until DATE NULL
);

CREATE TABLE tour_exceptions (
    idException INT AUTO_INCREMENT PRIMARY KEY,
    exception_date DATE NOT NULL,
    description VARCHAR(255) NULL,
    idTour INT NOT NULL,
    FOREIGN KEY (idTour) REFERENCES tours(idTour) ON DELETE CASCADE
);

-- Statuses
CREATE TABLE statuses (
    idStatus INT PRIMARY KEY,
    statusName VARCHAR(50) NOT NULL
);

-- BookVenues
CREATE TABLE bookVenues (
    idBookVenue INT AUTO_INCREMENT PRIMARY KEY,
    idVenue INT NOT NULL,
    idStatus INT NOT NULL,
    date DATE NOT NULL,
    idUser INT NULL,
    created_by_idUser INT NULL,
    on_behalf_of_name VARCHAR(255) NULL,
    on_behalf_of_email VARCHAR(255) NULL,
    on_behalf_of_phone VARCHAR(50) NULL,
    on_behalf_of_company VARCHAR(255) NULL,
    FOREIGN KEY (idVenue) REFERENCES venues(idVenue),
    FOREIGN KEY (idStatus) REFERENCES statuses(idStatus),
    FOREIGN KEY (idUser) REFERENCES users(idUser),
    FOREIGN KEY (created_by_idUser) REFERENCES users(idUser)
);

-- BookTours
CREATE TABLE bookTours (
    idBookTour INT AUTO_INCREMENT PRIMARY KEY,
    idTour INT NOT NULL,
    idStatus INT NOT NULL,
    date DATE NOT NULL,
```

```
    children INT NOT NULL,
    class VARCHAR(255) NOT NULL,
    idUser INT NULL,
    created_by_idUser INT NULL,
    on_behalf_of_name VARCHAR(255) NULL,
    on_behalf_of_email VARCHAR(255) NULL,
    on_behalf_of_phone VARCHAR(50) NULL,
    on_behalf_of_school VARCHAR(255) NULL,
    FOREIGN KEY (idTour) REFERENCES tours(idTour),
    FOREIGN KEY (idStatus) REFERENCES statuses(idStatus),
    FOREIGN KEY (idUser) REFERENCES users(idUser),
    FOREIGN KEY (created_by_idUser) REFERENCES users(idUser)
);

-- BookAlerts
CREATE TABLE bookAlerts (
    idBookAlert INT AUTO_INCREMENT PRIMARY KEY,
    read_status BOOLEAN NOT NULL,
    dateAlert DATE NOT NULL,
    message VARCHAR(1000) NOT NULL,
    target_userId INT NULL,
    idBookVenue INT,
    idBookTour INT,
    FOREIGN KEY (idBookVenue) REFERENCES bookVenues(idBookVenue),
    FOREIGN KEY (idBookTour) REFERENCES bookTours(idBookTour)
);


-- ------------------
-- INITIAL DATA
-- ------------------

-- Roles
INSERT INTO roles (roleName) VALUES
('administrator'),
('venue_manager'),
('tour_manager'),
('user');

-- Statuses
INSERT INTO statuses (idStatus, statusName) VALUES
(1, 'pending'),
(2, 'accepted'),
(3, 'rejected'),
(4, 'cancelled');

-- Settings
INSERT INTO settings (setting_key, setting_value) VALUES
('contact_phone', '2710-271000'),
```

```
('contact_email', 'contact@museapp.gr'),
('address_street', 'Οδός Μουσείου 1'),
('address_city', 'Τρίπολη'),
('address_postal', '22100'),
('google_maps_url',
'https://www.google.com/maps/embed?pb=!1m18!1m12!1m3!1d25318.8824884791
4!2d22.355199178256104!3d37.511212653693775!2m3!1f0!2f0!3f0!3m2!1i1024!
2i768!4f13.1!3m3!1m2!1s0x136016f96cfcca21%3A0xdd407121c8634fca!2zzqTPgc
6vz4DOv867zrcgMjIxIDAw!5e0!3m2!1sel!2sgr!4v1751805021246!5m2!1sel!2sgr'
),
('monday', 'Κλειστά'),
('tuesday', '9:00 - 17:00'),
('wednesday', '9:00 - 17:00'),
('thursday', '9:00 - 17:00'),
('friday', '9:00 - 17:00'),
('saturday', '9:00 - 17:00'),
('sunday', '9:00 - 17:00'),
('admin_phone', '2710-271001'),
('admin_email', 'admin@museapp.gr'),
('venue_manager_phone', '2710-271002'),
('venue_manager_email', 'venues@museapp.gr'),
('tour_manager_phone', '2710-271003'),
('tour_manager_email', 'tours@museapp.gr'),
('staff_monday', '9:00 - 17:00'),
('staff_tuesday', '9:00 - 17:00'),
('staff_wednesday', '9:00 - 17:00'),
('staff_thursday', '9:00 - 17:00'),
('staff_friday', '9:00 - 17:00'),
('staff_saturday', '-'),
('staff_sunday', '-');

INSERT INTO users (username, firstname, lastname, password, phone,
mail, idRole) VALUES
('admin', 'Απόστολος', 'Πεσλής',
'$2y$10$CS1Zq7jM8GLmwLyDl5cFl.XMLfOlevooOim0cJL1VNDzwK.ETHWMW',
'6900271001', 'apeslis@uop.gr', 1);
-- pass: Museadmin1!
```

ΠΑΡΑΡΤΗΜΑ Γ': Κώδικας της Εφαρμογής

Γ.1 Κεντρική Υποδομή και Routing

Entry Point (Σημείο Εισόδου)

`public/index.php`

```php
<?php

session_start();

// 500 Error Handling
set_exception_handler(function($exception) {

    error_log($exception->getMessage());

    http_response_code(500);

    require_once dirname(__DIR__) . '/views/errors/500.php';

    exit();
});

date_default_timezone_set('Europe/Athens');

define('BASE_PATH', dirname(__DIR__));

// --- Φόρτωση Βασικών Αρχείων ---
require_once BASE_PATH . '/helpers.php';

require_once BASE_PATH . '/config/db.php';

require_once BASE_PATH . '/models/User.php';
require_once BASE_PATH . '/models/Venue.php';
require_once BASE_PATH . '/models/VenueException.php';
require_once BASE_PATH . '/models/Photo.php';
require_once BASE_PATH . '/models/Tour.php';
require_once BASE_PATH . '/models/TourException.php';
require_once BASE_PATH . '/models/Settings.php';
require_once BASE_PATH . '/models/ActivityLog.php';
require_once BASE_PATH . '/models/BookVenue.php';
require_once BASE_PATH . '/models/BookTour.php';
require_once BASE_PATH . '/models/BookAlert.php';

require_once BASE_PATH . '/controllers/PageController.php';
require_once BASE_PATH . '/controllers/AuthController.php';
require_once BASE_PATH . '/controllers/UserController.php';
```

```php
require_once BASE_PATH . '/controllers/ProfileController.php';
require_once BASE_PATH . '/controllers/SettingsController.php';
require_once BASE_PATH . '/controllers/ActivityLogController.php';
require_once BASE_PATH . '/controllers/VenueController.php';
require_once BASE_PATH . '/controllers/VenueExceptionController.php';
require_once BASE_PATH . '/controllers/TourController.php';
require_once BASE_PATH . '/controllers/TourExceptionController.php';
require_once BASE_PATH . '/controllers/ApiController.php';
require_once BASE_PATH . '/controllers/StatsController.php';
require_once BASE_PATH . '/controllers/BookingController.php';




// --- Δημιουργία Αντικειμένων ---
try {
    $pdo = (new Database())->connect();

    // Models
    $logModel           = new ActivityLog($pdo);
    $bookTourModel      = new BookTour($pdo);
    $bookVenueModel     = new BookVenue($pdo);
    $photoModel         = new Photo($pdo);
    $settingsModel      = new Settings($pdo);
    $tourModel          = new Tour($pdo);
    $tourExceptionModel = new TourException($pdo);
    $userModel          = new User($pdo);
    $venueModel         = new Venue($pdo);
    $venueExceptionModel= new VenueException($pdo);
    $bookAlertModel = new BookAlert($pdo);

    // Controllers
    $logController      = new ActivityLogController($logModel);
    $apiController      = new ApiController($bookVenueModel,
$bookTourModel, $tourModel, $tourExceptionModel, $bookAlertModel,
$venueModel, $venueExceptionModel, $userModel);
    $authController     = new AuthController($userModel, $logModel,
$settingsModel);
    $pageController     = new PageController($tourModel, $venueModel,
$photoModel, $settingsModel);
    $profileController  = new ProfileController($userModel, $logModel);
    $settingsController = new SettingsController($settingsModel,
$logModel);
    $tourController     = new TourController($tourModel, $logModel);
    $tourExceptionController = new
TourExceptionController($tourExceptionModel, $tourModel, $logModel);
    $userController     = new UserController($userModel, $logModel);
    $venueController    = new VenueController($venueModel, $photoModel,
$logModel);
```

```
    $venueExceptionController = new
VenueExceptionController($venueExceptionModel, $venueModel, $logModel);
    $statsController     = new StatsController($bookTourModel,
$bookVenueModel);
    $bookingController = new BookingController($bookTourModel,
$userModel, $logModel, $bookAlertModel, $tourModel, $venueModel,
$bookVenueModel);

} catch (PDOException $e) {
    http_response_code(500);
    error_log("Database connection failed: " . $e->getMessage());
    die("Σφάλμα σύνδεσης στη βάση. Παρακαλώ δοκιμάστε ξανά αργότερα.");
}

// --- Ορισμός Διαδρομών (Routes) ---
$routes = [
    // --- Public Routes ---
    'GET /' => [function() use ($pageController) { $pageController-
>home(); }],
    'GET /contact' => [function() use ($pageController) {
$pageController->contact(); }],
    'GET /school-tours' => [function() use ($pageController) {
$pageController->schoolTours(); }],
    'GET /venues-list' => [function() use ($pageController) {
$pageController->venues(); }],
    'GET /venue/{id}' => [function(int $id) use ($pageController) {
$pageController->venueDetail($id); }],

    // --- Authentication Routes ---
    'GET /login' => [function() use ($authController) {
$authController->showLoginForm(); }],
    'POST /login' => [function() use ($authController) {
$authController->login(); }],
    'GET /logout' => [function() use ($authController) {
$authController->logout(); }],
    'GET /register' => [function() use ($authController) {
$authController->showRegisterForm(); }],
    'POST /register' => [function() use ($authController) {
$authController->register(); }],
    'GET /forgot-password' => [function() use ($authController) {
$authController->showForgotPasswordForm(); }],

    // --- Logged-in User Routes ---
    'GET /profile/edit' => [function() use ($profileController) { if
(!is_logged_in()) { header('Location: /login'); exit(); }
$profileController->showProfileForm(); }],
```

```
    'POST /profile/edit' => [function() use ($profileController) { if
(!is_logged_in()) { header('Location: /login'); exit(); }
$profileController->updateProfile(); }],
    'GET /profile/change-password' => [function() use
($profileController) { if (!is_logged_in()) { header('Location:
/login'); exit(); } $profileController->showChangePasswordForm(); }],
    'POST /profile/change-password' => [function() use
($profileController) { if (!is_logged_in()) { header('Location:
/login'); exit(); } $profileController->updatePassword(); }],

    'POST /tours/book' => [function() use ($bookingController) {
$bookingController->storeTourBooking(); }],
    'GET /tours-calendar' => [function() use ($pageController) {
$pageController->toursCalendar(); }],
    'GET /venues-calendar' => [function() use ($pageController) {
$pageController->venuesCalendar(); }],
    'GET /my-bookings' => [function() use ($bookingController) { if
(is_logged_in()) { $bookingController->listUserBookings(); } else {
header('Location: /login'); } }],

    'GET /stats' => [function() use ($statsController) { if
(is_super_admin() || is_venue_manager() || is_tour_manager()) {
$statsController->index(); } else { require_once BASE_PATH .
'/views/errors/403.php'; } }],

    // --- Venue Management Routes ---
    'GET /venues' => [function() use ($venueController) { if
(is_super_admin() || is_venue_manager()) { $venueController->index(); }
else { require_once BASE_PATH . '/views/errors/403.php'; } }],
    'GET /venues/create' => [function() use ($venueController) { if
(is_super_admin() || is_venue_manager()) { $venueController->create();
} else { require_once BASE_PATH . '/views/errors/403.php'; } }],
    'POST /venues' => [function() use ($venueController) { if
(is_super_admin() || is_venue_manager()) { $venueController->store(); }
else { require_once BASE_PATH . '/views/errors/403.php'; } }],
    'GET /venues/edit/{id}' => [function(int $id) use
($venueController) { if (is_super_admin() || is_venue_manager()) {
$venueController->edit($id); } else { require_once BASE_PATH .
'/views/errors/403.php'; } }],
    'POST /venues/update/{id}' => [function(int $id) use
($venueController) { if (is_super_admin() || is_venue_manager()) {
$venueController->update($id); } else { require_once BASE_PATH .
'/views/errors/403.php'; } }],
    'POST /venues/toggle/{id}' => [function(int $id) use
($venueController) { if (is_super_admin() || is_venue_manager()) {
$venueController->toggleStatus($id); } else { require_once BASE_PATH .
'/views/errors/403.php'; } }],
```

```
    'POST /venues/delete-unused/{id}' => [function(int $id) use
($venueController) { if (is_super_admin() || is_venue_manager()) {
$venueController->destroyUnused($id); } else { require_once BASE_PATH .
'/views/errors/403.php'; } }],
    'GET /venue-exceptions' => [function() use
($venueExceptionController) { if (is_super_admin() ||
is_venue_manager()) { $venueExceptionController->index(); } else {
require_once BASE_PATH . '/views/errors/403.php'; } }],
    'POST /venue-exceptions' => [function() use
($venueExceptionController) { if (is_super_admin() ||
is_venue_manager()) { $venueExceptionController->store(); } else {
require_once BASE_PATH . '/views/errors/403.php'; } }],
    'POST /venue-exceptions/delete/{id}' => [function(int $id) use
($venueExceptionController) { if (is_super_admin() ||
is_venue_manager()) { $venueExceptionController->destroy($id); } else {
require_once BASE_PATH . '/views/errors/403.php'; } }],
    'POST /venues/edit/{id}/upload-photo' => [function(int $id) use
($venueController) {if (is_super_admin() || is_venue_manager()) {
$venueController->uploadPhoto($id); } else { require_once BASE_PATH .
'/views/errors/403.php'; } }],
    'POST /venues/edit/{venueId}/set-primary/{photoId}' =>
[function(int $venueId, int $photoId) use ($venueController) { if
(is_super_admin() || is_venue_manager()) { $venueController-
>setPrimaryPhoto($venueId, $photoId); } else { require_once BASE_PATH .
'/views/errors/403.php'; }}],
    'POST /venues/edit/{venueId}/delete-photo/{photoId}' =>
[function(int $venueId, int $photoId) use ($venueController) { if
(is_super_admin() || is_venue_manager()) { $venueController-
>deletePhoto($venueId, $photoId); } else { require_once BASE_PATH .
'/views/errors/403.php'; } }],

    'POST /venues/book' => [function() use ($bookingController) {
$bookingController->storeVenueBooking(); }],
    'GET /venues/requests' => [function() use ($bookingController) { if
(is_super_admin() || is_venue_manager()) { $bookingController-
>listPendingVenueBookings(); } else { require_once BASE_PATH .
'/views/errors/403.php'; } }],
    'GET /venues/requests' => [function() use ($bookingController) { if
(is_super_admin() || is_venue_manager()) { $bookingController-
>listPendingVenueBookings(); } else { require_once BASE_PATH .
'/views/errors/403.php'; } }],
    'GET /venues/bookings' => [function() use ($bookingController) { if
(is_super_admin() || is_venue_manager()) { $bookingController-
>listAllVenueBookings(); } else { require_once BASE_PATH .
'/views/errors/403.php'; } }],
    'POST /venues/bookings/approve/{id}' => [function(int $id) use
($bookingController) { if (is_super_admin() || is_venue_manager()) {
```

```
$bookingController->approveVenueBooking($id); } else { require_once
BASE_PATH . '/views/errors/403.php'; } }],
    'POST /venues/bookings/reject/{id}' => [function(int $id) use
($bookingController) { if (is_super_admin() || is_venue_manager()) {
$bookingController->rejectVenueBooking($id); } else { require_once
BASE_PATH . '/views/errors/403.php'; } }],
    'POST /venues/bookings/cancel-by-user/{id}' => [function(int $id)
use ($bookingController) { if (is_simple_user()) { $bookingController-
>cancelVenueBookingByUser($id); } else { http_response_code(403);
require_once BASE_PATH . '/views/errors/403.php'; } }],
    'GET /venues/create-booking' => [function() use
($bookingController) { if (is_super_admin() || is_venue_manager()) {
$bookingController->showCreateVenueBookingFormForManager(); } else {
require_once BASE_PATH . '/views/errors/403.php'; }  }],
    'POST /venues/create-booking' => [function() use
($bookingController) { if (is_super_admin() || is_venue_manager()) {
$bookingController->storeVenueBookingByManager(); } else { require_once
BASE_PATH . '/views/errors/403.php'; } }],
    'POST /venues/bookings/cancel/{id}' => [function(int $id) use
($bookingController) { if (is_super_admin() || is_venue_manager()) {
$bookingController->cancelVenueBookingByManager($id); } else {
require_once BASE_PATH . '/views/errors/403.php'; } }],


    // --- Tour Management Routes ---
    'GET /tours' => [function() use ($tourController) { if
(is_super_admin() || is_tour_manager()) { $tourController->index(); }
else { require_once BASE_PATH . '/views/errors/403.php'; } }],
    'GET /tours/create' => [function() use ($tourController) { if
(is_super_admin() || is_tour_manager()) { $tourController->create(); }
else { require_once BASE_PATH . '/views/errors/403.php'; } }],
    'POST /tours' => [function() use ($tourController) { if
(is_super_admin() || is_tour_manager()) { $tourController->store(); }
else { require_once BASE_PATH . '/views/errors/403.php'; } }],
    'GET /tours/edit/{id}' => [function(int $id) use ($tourController)
{ if (is_super_admin() || is_tour_manager()) { $tourController-
>edit($id); } else { require_once BASE_PATH . '/views/errors/403.php';
} }],
    'POST /tours/update/{id}' => [function(int $id) use
($tourController) { if (is_super_admin() || is_tour_manager()) {
$tourController->update($id); } else { require_once BASE_PATH .
'/views/errors/403.php'; } }],
    'POST /tours/toggle/{id}' => [function(int $id) use
($tourController) { if (is_super_admin() || is_tour_manager()) {
$tourController->toggleStatus($id); } else { require_once BASE_PATH .
'/views/errors/403.php'; } }],
    'POST /tours/delete-unused/{id}' => [function(int $id) use
($tourController) { if (is_super_admin() || is_tour_manager()) {
```

```
$tourController->destroyUnused($id); } else { require_once BASE_PATH .
'/views/errors/403.php'; } }],
    'GET /tour-exceptions' => [function() use
($tourExceptionController) { if (is_super_admin() || is_tour_manager())
{ $tourExceptionController->index(); } else { require_once BASE_PATH .
'/views/errors/403.php'; } }],
    'POST /tour-exceptions' => [function() use
($tourExceptionController) { if (is_super_admin() || is_tour_manager())
{ $tourExceptionController->store(); } else { require_once BASE_PATH .
'/views/errors/403.php'; } }],
    'POST /tour-exceptions/delete/{id}' => [function(int $id) use
($tourExceptionController) { if (is_super_admin() || is_tour_manager())
{ $tourExceptionController->destroy($id); } else { require_once
BASE_PATH . '/views/errors/403.php'; } }],

    'GET /tours/bookings' => [function() use ($bookingController) { if
(is_super_admin() || is_tour_manager()) { $bookingController-
>listTourBookings(); } else { require_once BASE_PATH .
'/views/errors/403.php'; } }],
    'POST /tours/bookings/cancel/{id}' => [function(int $id) use
($bookingController) { if (is_super_admin() || is_tour_manager()) {
$bookingController->cancelTourBookingByManager($id); } else {
require_once BASE_PATH . '/views/errors/403.php'; } }],
    'POST /tours/book' => [function() use ($bookingController) {
$bookingController->storeTourBooking(); }],
    'POST /tours/bookings/cancel-by-user/{id}' => [function(int $id)
use ($bookingController) { if (is_simple_user()) { $bookingController-
>cancelTourBookingByUser($id); } else { http_response_code(403);
require_once BASE_PATH . '/views/errors/403.php'; } }],
    'GET /tours/create-booking' => [function() use ($bookingController)
{ if (is_super_admin() || is_tour_manager()) { $bookingController-
>showCreateTourBookingFormForManager(); } else { require_once BASE_PATH
. '/views/errors/403.php'; } }],
    'POST /tours/create-booking' => [function() use
($bookingController) { if (is_super_admin() || is_tour_manager()) {
$bookingController->storeTourBookingByManager(); } else { require_once
BASE_PATH . '/views/errors/403.php'; }}],



    // --- Super Admin Routes ---
    'GET /users' => [function() use ($userController) { if
(is_super_admin()) { $userController->index(); } else { require_once
BASE_PATH . '/views/errors/403.php'; } }],
    'GET /users/create' => [function() use ($userController) { if
(is_super_admin()) { $userController->create(); } else { require_once
BASE_PATH . '/views/errors/403.php'; } }],
```

```
    'POST /users' => [function() use ($userController) { if
(is_super_admin()) { $userController->store(); } else { require_once
BASE_PATH . '/views/errors/403.php'; } }],
    'GET /users/edit/{id}' => [function(int $id) use ($userController)
{ if (is_super_admin()) { $userController->edit($id); } else {
require_once BASE_PATH . '/views/errors/403.php'; } }],
    'POST /users/update/{id}' => [function(int $id) use
($userController) { if (is_super_admin()) { $userController-
>update($id); } else { require_once BASE_PATH .
'/views/errors/403.php'; } }],
    'POST /users/toggle/{id}' => [function(int $id) use
($userController) { if (is_super_admin()) { $userController-
>toggleStatus($id); } else { require_once BASE_PATH .
'/views/errors/403.php'; } }],
    'POST /users/reset/{id}' => [function(int $id) use
($userController) { if (is_super_admin()) { $userController-
>resetPassword($id); } else { require_once BASE_PATH .
'/views/errors/403.php'; } }],
    'GET /settings' => [function() use ($settingsController) { if
(is_super_admin()) { $settingsController->index(); } else {
require_once BASE_PATH . '/views/errors/403.php'; } }],
    'POST /settings' => [function() use ($settingsController) { if
(is_super_admin()) { $settingsController->update(); } else {
require_once BASE_PATH . '/views/errors/403.php'; } }],
    'GET /logs' => [function() use ($logController) { if
(is_super_admin()) { $logController->index(); } else { require_once
BASE_PATH . '/views/errors/403.php'; } }],

    // --- API Routes ---
    'GET /api/tour-events' => [function() use ($apiController) { if
(is_logged_in()) { $apiController->getTourEvents(); } else {
http_response_code(403); exit('Access Denied'); } }],
    'GET /api/available-tours' => [function() use ($apiController) { if
(is_logged_in()) { $apiController->getAvailableToursForDate(); } else {
http_response_code(403); exit('Access Denied'); } }],

    'GET /api/venue-events' => [function() use ($apiController) { if
(is_logged_in()) { $apiController->getVenueEvents(); } else {
http_response_code(403); exit('Access Denied'); } }],
    'GET /api/available-venues' => [function() use ($apiController) {
if (is_logged_in()) { $apiController->getAvailableVenuesForDate(); }
else { http_response_code(403); exit('Access Denied'); } }],

    'GET /api/alerts' => [function() use ($apiController) { if
(is_logged_in()) { $apiController->getAlerts(); } else {
http_response_code(403); exit('Access Denied'); } }],
```

```
    'POST /api/alerts/mark-as-read' => [function() use ($apiController)
{ if (is_logged_in()) { $apiController->markAlertAsRead(); } else {
http_response_code(403); exit('Access Denied'); } }],

    'GET /api/booking-details' => [function() use ($apiController) { if
(is_logged_in()) { $apiController->getBookingDetails(); } else {
http_response_code(403); exit('Access Denied'); } }],
];



// --- ΛΟΓΙΚΗ ΤΟΥ ROUTER ---
$request_key = "{$_SERVER['REQUEST_METHOD']}
{$_SERVER['REQUEST_URI']}";

foreach ($routes as $route_pattern => $handler) {
    // Μετατρέπουμε το pattern του πίνακα (π.χ., 'GET
/venues/edit/{id}') σε regular expression
    $regex_pattern = preg_replace('/\{([a-zA-Z]+)\}/', '([0-9]+)',
$route_pattern);
    $regex_pattern = "#^{$regex_pattern}$#";

    // Παίρνουμε μόνο το path από το URL για να ταιριάξουμε, χωρίς
query παραμέτρους
    $request_path = parse_url($_SERVER['REQUEST_URI'], PHP_URL_PATH);
    $string_to_match = "{$_SERVER['REQUEST_METHOD']} {$request_path}";

    if (preg_match($regex_pattern, $string_to_match, $matches)) {
        // Βρέθηκε

        // Το πρώτο στοιχείο στο $matches είναι ολόκληρο το string που
ταίριαξε. Το αφαιρούμε.
        array_shift($matches);

        // Τα υπόλοιπα στοιχεία είναι οι παράμετροι (π.χ., το id).
        $params = $matches;

        // Καλούμε τη συνάρτηση του controller, περνώντας τις
παραμέτρους.
        call_user_func_array($handler[0], $params);

        // Αφού βρήκαμε και εκτελέσαμε τη διαδρομή, σταματάμε εντελώς
την εκτέλεση του script.
        exit();
    }
}

// Αν το loop τελειώσει και δεν έχουμε κάνει exit(), σημαίνει ότι δεν
βρέθηκε καμία διαδρομή.
```

```php
http_response_code(404);
require_once BASE_PATH . '/views/errors/404.php';
exit();
```

Helper Functions (Βοηθητικές Συναρτήσεις)

`helpers.php`

```php
<?php

// ====================================================================
// HELPERS ΓΙΑ ΔΙΑΔΡΟΜΕΣ (URL & ASSET HELPERS)
// ====================================================================

if (!function_exists('asset')) {
    /**
     * Δημιουργεί ένα απόλυτο URL για ένα αρχείο asset (CSS, JS,
εικόνα).
     * @param string $path Η σχετική διαδρομή προς το asset.
     * @return string Το πλήρες, απόλυτο URL.
     */
    function asset($path): string
    {
        $path = ltrim($path, '/');
        return "/{$path}";
    }
}

if (!function_exists('url')) {
    /**
     * Δημιουργεί ένα απόλυτο URL για μια διαδρομή της εφαρμογής.
     * @param string $path Η σχετική διαδρομή.
     * @return string Το πλήρες, απόλυτο URL.
     */
    function url($path): string
    {
        $path = ltrim($path, '/');
        return "/{$path}";
    }
}


// ====================================================================
// HELPERS ΓΙΑ AUTHENTICATION & ROLES
// ====================================================================

if (!function_exists('is_logged_in')) {
    /**
     * Ελέγχει αν ο χρήστης είναι συνδεδεμένος.
```

```
     * @return bool
     */
    function is_logged_in(): bool
    {
        return isset($_SESSION['user_id']);
    }
}

if (!function_exists('has_role')) {
    /**
     * (Internal Helper) Ελέγχει αν ο συνδεδεμένος χρήστης έχει έναν
συγκεκριμένο ρόλο.
     * @param string $roleName Το όνομα του ρόλου όπως είναι στη βάση
δεδομένων.
     * @return bool
     */
    function has_role(string $roleName): bool
    {
        return isset($_SESSION['user_role']) && $_SESSION['user_role']
=== $roleName;
    }
}

if (!function_exists('is_super_admin')) {
    /**
     * Ελέγχει αν ο χρήστης είναι Super Administrator.
     * @return bool
     */
    function is_super_admin(): bool
    {
        return has_role('administrator');
    }
}

if (!function_exists('is_venue_manager')) {
    /**
     * Ελέγχει αν ο χρήστης είναι Venue Manager.
     * @return bool
     */
    function is_venue_manager(): bool
    {
        return has_role('venue_manager');
    }
}

if (!function_exists('is_tour_manager')) {
    /**
     * Ελέγχει αν ο χρήστης είναι Tour Manager.
```

```
     * @return bool
     */
    function is_tour_manager(): bool
    {
        return has_role('tour_manager');
    }
}

if (!function_exists('is_simple_user')) {
    /**
     * Ελέγχει αν ο χρήστης είναι απλός User.
     * @return bool
     */
    function is_simple_user(): bool
    {
        return has_role('user');
    }
}


// ================================================================
// HELPERS ΓΙΑ ΜΗΝΥΜΑΤΑ (FLASH MESSAGES)
// ================================================================

if (!function_exists('flash')) {
    /**
     * Ορίζει, ανακτά και εμφανίζει flash messages.
     * - ΟΡΙΣΜΟΣ: flash('success', 'Το μήνυμά σου εδώ.');
     * - ΕΜΦΑΝΙΣΗ: flash();
     * @param string|null $type 'success', 'danger', 'info'
     * @param string|null $message Το μήνυμα προς εμφάνιση.
     * @param bool $allow_html Αν θα επιτρέπεται η εμφάνιση HTML tags
στο μήνυμα.
     */
    function flash($type = null, $message = null, $allow_html = false):
void
    {
        // Ορισμός μηνύματος
        if ($type && $message) {
            $_SESSION['flash_messages'][] = [
                'type' => $type,
                'message' => $message,
                'allow_html' => $allow_html
            ];
            return;
        }

        // Εμφάνιση μηνυμάτων
```

```
        if (isset($_SESSION['flash_messages']) &&
is_array($_SESSION['flash_messages'])) {
            foreach ($_SESSION['flash_messages'] as $flash) {
                $display_message = $flash['allow_html'] ?
$flash['message'] : htmlspecialchars($flash['message']);
                echo '<div class="alert alert-' .
htmlspecialchars($flash['type']) . '">' . $display_message . '</div>';
            }
            unset($_SESSION['flash_messages']);
        }
    }
}

?>
```

Γ.2 Αυθεντικοποίηση

Αρχείο Ελεγκτή

`app/controllers/AuthController.php`

```
<?php

class AuthController
{
    private $userModel;
    private $logModel;
    private $settingsModel;

    public function __construct(User $userModel, ActivityLog $logModel,
Settings $settingsModel)
    {
        $this->userModel = $userModel;
        $this->logModel = $logModel;
        $this->settingsModel = $settingsModel;
    }

    public function showLoginForm()
    {
        if (is_logged_in()) {
            header('Location: ' . url('/'));
            exit();
        }
        require_once BASE_PATH . '/views/auth/login.php';
    }

    public function showRegisterForm()
    {
```

```
        if (is_logged_in()) {
            header('Location: ' . url('/'));
            exit();
        }

        $old_post = $_SESSION['old_post'] ?? [];
        unset($_SESSION['old_post']);

        require_once BASE_PATH . '/views/auth/register.php';
    }

    public function showForgotPasswordForm()
    {
        $settings = $this->settingsModel->getAllAsArray();
        require_once BASE_PATH . '/views/auth/forgot-password.php';
    }

    public function login()
    {
        $username = $_POST['username'] ?? '';
        $password = $_POST['password'] ?? '';

        $user = $this->userModel->findByUsernameWithRole($username);

        // Έλεγχος 1: Ο χρήστης υπάρχει και ο κωδικός είναι σωστός;
        if ($user && password_verify($password, $user['password'])) {

            // Έλεγχος 2: Ο χρήστης είναι ενεργός;
            if ($user['active'] == 0) {
                $this->logModel->logAction('LOGIN_DISABLED',
"Προσπάθεια σύνδεσης σε απενεργοποιημένο λογαριασμό ('{$username}').",
$user['idUser']);
                flash('danger', 'Ο λογαριασμός σας έχει
απενεργοποιηθεί.');
                header('Location: ' . url('/login'));
                exit();
            }

            // Επιτυχής σύνδεση.
            $this->logModel->logAction('LOGIN_SUCCESS', "Επιτυχής
σύνδεση για τον χρήστη '{$username}'.", $user['idUser']);

            $_SESSION['user_id'] = $user['idUser'];
            $_SESSION['username'] = $user['username'];
            $_SESSION['user_role'] = $user['roleName'];

            header('Location: ' . url('/'));
            exit();
```

```
        } else {
            $this->logModel->logAction('LOGIN_FAILURE', "Αποτυχημένη
προσπάθεια σύνδεσης για τον χρήστη '{$username}'.");
            flash('danger', 'Λάθος όνομα χρήστη ή κωδικός.');
            header('Location: ' . url('/login'));
            exit();
        }
    }

    public function register()
    {
        // Αποθηκεύουμε προσωρινά τα δεδομένα σε περίπτωση σφάλματος
        $_SESSION['old_post'] = $_POST;

        // 1. Βασικός έλεγχος αν τα πεδία έχουν σταλεί
        if (empty($_POST['username']) || empty($_POST['password']) ||
empty($_POST['mail']) || empty($_POST['firstname']) ||
empty($_POST['lastname']) || empty($_POST['phone'])) {
            flash('danger', 'Παρακαλώ συμπληρώστε όλα τα πεδία με
αστερίσκο (*).');
            header('Location: ' . url('/register'));
            exit();
        }

        // 2. Έλεγχος αν οι κωδικοί ταιριάζουν
        if ($_POST['password'] !== $_POST['password_repeat']) {
            flash('danger', 'Οι κωδικοί που εισαγάγατε δεν
ταιριάζουν.');
            header('Location: ' . url('/register'));
            exit();
        }

        // 3. Έλεγχος πολιτικής κωδικού (Server-side)
        $password = $_POST['password'];
        $errors = [];
        if (strlen($password) < 8) $errors[] = 'Ο κωδικός πρέπει να
έχει τουλάχιστον 8 χαρακτήρες.';
        if (!preg_match('/[A-Z]/', $password)) $errors[] = 'Ο κωδικός
πρέπει να περιέχει τουλάχιστον ένα κεφαλαίο γράμμα.';
        if (!preg_match('/[a-z]/', $password)) $errors[] = 'Ο κωδικός
πρέπει να περιέχει τουλάχιστον ένα μικρό γράμμα.';
        if (!preg_match('/[0-9]/', $password)) $errors[] = 'Ο κωδικός
πρέπει να περιέχει τουλάχιστον έναν αριθμό.';
        if (!preg_match('/[!@#$%^&*]/', $password)) $errors[] = 'Ο
κωδικός πρέπει να περιέχει τουλάχιστον έναν ειδικό χαρακτήρα
(!@#$%^&*).';
```

```
        if (!empty($errors)) {
            $errorMessage = 'Ο κωδικός δεν πληροί τις προϋποθέσεις
ασφαλείας:<ul>';
            foreach ($errors as $error) { $errorMessage .= '<li>' .
$error . '</li>'; }
            $errorMessage .= '</ul>';
            flash('danger', $errorMessage, true);
            header('Location: ' . url('/register'));
            exit();
        }

        // 4. Προσπάθεια δημιουργίας χρήστη
        $success = $this->userModel->create($_POST);

        if ($success) {
            unset($_SESSION['old_post']); // Καθαρίζουμε τα δεδομένα
μόνο σε επιτυχία
            flash('success', 'Ο λογαριασμός σας δημιουργήθηκε με
επιτυχία! Μπορείτε τώρα να συνδεθείτε.');
            $this->logModel->logAction(
                'USER_REGISTER_SUCCESS',
                "Ένας νέος χρήστης εγγράφηκε με username
'{$_POST['username']}'."
                // Δεν περνάμε user id γιατί δεν το ξέρουμε ακόμα και
είναι ενέργεια συστήματος
            );
            header('Location: ' . url('/login'));
            exit();
        } else {
            // Η αποτυχία σημαίνει ότι το username/email υπάρχει ήδη.
            header('Location: ' . url('/register'));
            exit();
        }
    }

    public function logout()
    {
        if (isset($_SESSION['user_id'])) {
            $userId = $_SESSION['user_id'];
            $username = $_SESSION['username'];
            $this->logModel->logAction(
                'LOGOUT_SUCCESS',
                "Ο χρήστης '{$username}' αποσυνδέθηκε.",
                $userId
            );
        }
        session_unset();
        session_destroy();
```

```
        header('Location: ' . url('/'));
        exit();
    }
}
```

Αρχεία Προβολής

`app/views/auth/login.php`

```
<?php require_once BASE_PATH . '/views/includes/header.inc.php'; ?>

<div class="container">
    <div class="row justify-content-center">
        <div class="col-xl-6 col-lg-8 col-md-9">
            <div class="card o-hidden border-0 shadow-lg my-5">
                <div class="card-body p-0">
                    <div class="p-5">
                        <div class="text-center">
                            <h1 class="h4 text-gray-900 mb-4">Καλώς
Ήρθατε!</h1>
                        </div>

                        <?php flash(); ?>

                        <form class="user" action="<?= url('/login')
?>" method="POST">
                            <div class="form-group">
                                <input type="text" class="form-control
form-control-user" name="username" placeholder="Όνομα χρήστη..."
required>
                            </div>
                            <div class="form-group">
                                <input type="password" class="form-
control form-control-user" name="password" placeholder="Κωδικός"
required>
                            </div>
                            <button type="submit" class="btn btn-
primary btn-user btn-block col-5 justify-content-center mx-auto">
                                Είσοδος
                            </button>
                        </form>
                        <hr>
                        <div class="text-center">
                            <a class="small" href="<?= url('/forgot-
password') ?>">Ξεχάσατε τον κωδικό;</a>
                        </div>
                        <div class="text-center">
                            <a class="small" href="<?= url('/register')
?>">Δημιουργία Λογαριασμού!</a>
```

```
                    </div>
                </div>
            </div>
        </div>
    </div>
</div>

<?php require_once BASE_PATH . '/views/includes/footer.inc.php'; ?>
```

app/views/auth/register.php

```
<?php require_once BASE_PATH . '/views/includes/header.inc.php'; ?>

<div class="container">
    <div class="card o-hidden border-0 shadow-lg my-5">
        <div class="card-body p-0">
            <!-- Nested Row within Card Body -->
            <div class="row">
                <div class="col-lg-12">
                    <div class="p-5">

                        <div class="text-center">
                            <h1 class="h4 text-gray-900 mb-
4">Δημιουργία Λογαριασμού!</h1>
                        </div>

                        <?php flash(); ?>

                        <form class="user" action="<?= url('/register')
?>" method="POST" id="registerForm">
                            <div class="form-group row">
                                <div class="col-sm-6 mb-3 mb-sm-0">
                                    <input type="text" class="form-
control form-control-user" name="firstname" value="<?=
htmlspecialchars($old_post['firstname'] ?? '') ?>" placeholder="Όνομα*"
required>
                                </div>
                                <div class="col-sm-6">
                                    <input type="text" class="form-
control form-control-user" name="lastname" value="<?=
htmlspecialchars($old_post['lastname'] ?? '') ?>"
placeholder="Επώνυμο*" required>
                                </div>
                            </div>
                            <div class="form-group">
                                <input type="email" class="form-control
form-control-user" name="mail" value="<?=
```

```
htmlspecialchars($old_post['mail'] ?? '') ?>" placeholder="Διεύθυνση
Email*" required>
                            </div>
                            <div class="form-group">
                                <input type="text" class="form-control
form-control-user" name="username" value="<?=
htmlspecialchars($old_post['username'] ?? '') ?>" placeholder="Όνομα
Χρήστη*" required>
                            </div>
                            <div class="form-group">
                                <input type="text" class="form-control
form-control-user" name="phone" value="<?=
htmlspecialchars($old_post['phone'] ?? '') ?>" placeholder="Τηλέφωνο*"
required>
                            </div>
                             <div class="form-group">
                                <input type="text" class="form-control
form-control-user" name="company" value="<?=
htmlspecialchars($old_post['company'] ?? '') ?>" placeholder="Εταιρεία
/ Οργανισμός (για κρατήσεις αιθουσών εκδηλώσεων)">
                            </div>
                             <div class="form-group">
                                <input type="text" class="form-control
form-control-user" name="school" value="<?=
htmlspecialchars($old_post['school'] ?? '') ?>" placeholder="Σχολείο
(για κρατήσεις σχολικών επισκέψεων)">
                            </div>
                            <div class="form-group row">
                                <div class="col-sm-6 mb-3 mb-sm-0">
                                    <input type="password" class="form-
control form-control-user"
                                        id="password" name="password"
placeholder="Κωδικός*" required
                                        pattern="(?=.*\d)(?=.*[a-
z])(?=.*[A-Z])(?=.*[!@#$%^&*]).{8,}"
                                        title="Ο κωδικός πρέπει να
περιέχει τουλάχιστον 8 χαρακτήρες, ένα μικρό γράμμα, ένα κεφαλαίο
γράμμα, έναν αριθμό και έναν ειδικό χαρακτήρα (!@#$%^&*).">
                                </div>
                                <div class="col-sm-6">
                                    <input type="password" class="form-
control form-control-user"
                                        id="password_repeat"
name="password_repeat" placeholder="Επανάληψη Κωδικού*" required>
                                </div>
                            </div>
                            <div class="card bg-warning  mb-3 text-
gray-100 text-center">
```

```
                                Ο κωδικός πρέπει να περιέχει
τουλάχιστον 8 χαρακτήρες, 1 κεφαλαίο, 1 μικρό, 1 αριθμό & 1 ειδικό
χαρακτήρα (!@#$%^&*). </br>
                                Τα πεδία με το * είναι υποχρεωτικά.
                            </div>
                            <button type="submit" class="btn btn-
primary btn-user btn-block col-5 justify-content-center mx-auto">
                                Εγγραφή
                            </button>
                        </form>
                        <hr>
                        <div class="text-center">
                            <a class="small" href="<?= url('/login')
?>">Έχετε ήδη λογαριασμό; Συνδεθείτε!</a>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>
</div>

<script>
document.addEventListener('DOMContentLoaded', function() {
    // Παίρνουμε τα δύο input πεδία του κωδικού
    const password = document.getElementById('password');
    const confirmPassword = document.getElementById('password_repeat');

    // Μια συνάρτηση που θα κάνει τον έλεγχο
    function validatePassword() {
        if (password.value !== confirmPassword.value) {
            // Αν οι κωδικοί ΔΕΝ ταιριάζουν, θέτουμε ένα custom μήνυμα
σφάλματος στο πεδίο "Επανάληψη Κωδικού"
            confirmPassword.setCustomValidity("Οι κωδικοί δεν
ταιριάζουν.");
        } else {
            // Αν οι κωδικοί ταιριάζουν, καθαρίζουμε το μήνυμα
σφάλματος
            confirmPassword.setCustomValidity('');
        }
    }

    // Καλούμε τη συνάρτηση validatePassword() κάθε φορά που ο χρήστης
αλλάζει κάτι σε ένα από τα δύο πεδία
    password.addEventListener('change', validatePassword);
    confirmPassword.addEventListener('keyup', validatePassword);
});
</script>
```

```
<?php require_once BASE_PATH . '/views/includes/footer.inc.php'; ?>
```

app/views/auth/forgot-password.php

```
<?php require_once BASE_PATH . '/views/includes/header.inc.php'; ?>

<div class="container">

    <!-- Outer Row -->
    <div class="row justify-content-center">

        <div class="col-xl-8 col-lg-10 col-md-9">

            <div class="card o-hidden border-0 shadow-lg my-5">
                <div class="card-body p-0">
                    <!-- Nested Row within Card Body -->
                    <div class="row">
                        <div class="col-lg-12">
                            <div class="p-5">
                                <div class="text-center">
                                    <h1 class="h4 text-gray-900 mb-
2">Ξεχάσατε τον Κωδικό σας;</h1>
                                    <p class="mb-4">Για την ασφάλεια
του λογαριασμού σας, η επαναφορά του κωδικού γίνεται μέσω του
διαχειριστή του συστήματος.</p>
                                </div>

                                <div class="alert alert-info"
role="alert">
                                  <h4 class="alert-heading">Οδηγίες
Επαναφοράς</h4>
                                  <p>Παρακαλούμε επικοινωνήστε με τον
διαχειριστή στα παρακάτω στοιχεία για να ζητήσετε την επαναφορά του
κωδικού πρόσβασής σας. Θα σας δοθεί ένας προσωρινός κωδικός για να
συνδεθείτε.</p>
                                  <hr>
                                  <p class="mb-0">
                                    <i class="fas fa-phone fa-fw mr-2
text-gray-500"></i>
                                        <strong>Τηλέφωνο:</strong> <a
href="tel:<?= htmlspecialchars($settings['admin_phone'] ?? '')?>"><?=
htmlspecialchars($settings['admin_phone'] ?? 'N/A') ?></a><br>
                                        <i class="fas fa-envelope fa-fw
mr-2 text-gray-500"></i>
                                        <strong>Email:</strong> <a
href="mailto:<?= htmlspecialchars($settings['admin_email'] ?? '')
?>"><?= htmlspecialchars($settings['admin_email'] ?? 'N/A') ?></a>
```

```
                                </p>
                            </div>

                            <hr>
                            <div class="text-center">
                                <a class="small" href="<?=
url('/register') ?>">Δημιουργία Λογαριασμού!</a>
                            </div>
                            <div class="text-center">
                                <a class="small" href="<?=
url('/login') ?>">Έχετε ήδη λογαριασμό; Συνδεθείτε!</a>
                            </div>
                        </div>
                    </div>
                </div>
            </div>

        </div>

    </div>

</div>

<?php require_once BASE_PATH . '/views/includes/footer.inc.php'; ?>
```

Γ.3 Διαχείριση Χρηστών

Αρχείο Μοντέλου (`app/models/User.php`)

```
<?php

class User
{
    private $pdo;

    public function __construct(PDO $pdo)
    {
        $this->pdo = $pdo;
    }

    /**
     * Βρίσκει έναν χρήστη με βάση το username του.
     */
    public function findByUsername(string $username)
    {
        $stmt = $this->pdo->prepare("SELECT * FROM users WHERE username
= ?");
        $stmt->execute([$username]);
```

```
        return $stmt->fetch();
    }

    /**
     * Βρίσκει έναν χρήστη με βάση το username του, μαζί με το όνομα
του ρόλου του.
     */
    public function findByUsernameWithRole(string $username)
    {
        $stmt = $this->pdo->prepare(
            "SELECT u.*, r.roleName
             FROM users u
             JOIN roles r ON u.idRole = r.idRole
             WHERE u.username = ?"
        );
        $stmt->execute([$username]);
        return $stmt->fetch();
    }

    /**
     * Επιστρέφει όλους τους χρήστες μαζί με το όνομα του ρόλου τους.
     */
    public function getAll()
    {
        $sql = "SELECT
                    u.idUser, u.username, u.firstname, u.lastname,
u.mail, u.phone, u.company, u.active, r.roleName
                FROM
                    users u
                JOIN
                    roles r ON u.idRole = r.idRole
                ORDER BY
                    u.lastname ASC, u.firstname ASC";

        $stmt = $this->pdo->query($sql);
        return $stmt->fetchAll();
    }

    /**
     * Δημιουργεί έναν νέο χρήστη στη βάση δεδομένων.
     * @param array $data Τα δεδομένα του χρήστη.
     * @param bool $isAdminAction Αν η δημιουργία γίνεται από admin
(για να οριστεί ρόλος).
     * @return bool
     */
    public function create(array $data, bool $isAdminAction = false):
bool
    {
```

```
        // Hashάρουμε τον κωδικό πριν την αποθήκευση!
        $password_hash = password_hash($data['password'],
PASSWORD_BCRYPT);

        // Αν η δημιουργία γίνεται από τον admin, παίρνουμε τον ρόλο
από τα data.
        // Αλλιώς, είναι δημόσια εγγραφή και ο ρόλος είναι πάντα 'user'
(ID=4).
        $roleId = $isAdminAction ? ($data['idRole'] ?? 4) : 4;

        $sql = "INSERT INTO users (username, firstname, lastname,
password, phone, mail, company, school, idRole)
                VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)";

        $stmt = $this->pdo->prepare($sql);

        try {
            return $stmt->execute([
                $data['username'],
                $data['firstname'],
                $data['lastname'],
                $password_hash,
                $data['phone'],
                $data['mail'],
                $data['company'] ?? null,
                $data['school'] ?? null,
                $roleId
            ]);
        } catch (PDOException $e) {
            // Έλεγχος για διπλότυπο username ή email
            if ($e->errorInfo[1] == 1062) {
                if (strpos($e->getMessage(), 'username') !== false) {
                    flash('danger', 'Αυτό το όνομα χρήστη
χρησιμοποιείται ήδη.');
                } elseif (strpos($e->getMessage(), 'mail') !== false) {
                    flash('danger', 'Αυτή η διεύθυνση email
χρησιμοποιείται ήδη.');
                }
            } else {
                 flash('danger', 'Παρουσιάστηκε ένα σφάλμα κατά την
εγγραφή.');
            }
            error_log("Database Error: " . $e->getMessage());
            return false;
        }
    }

    /**
```

```
     * Βρίσκει έναν χρήστη με βάση το ID του.
     */
    public function findById(int $id)
    {
        $stmt = $this->pdo->prepare("SELECT * FROM users WHERE idUser =
?");
        $stmt->execute([$id]);
        return $stmt->fetch();
    }

    /**
     * Ενημερώνει τον κωδικό ενός χρήστη.
     */
    public function updatePassword(int $id, string $new_password): bool
    {
        // Κάνουμε hash τον νέο κωδικό πριν την αποθήκευση
        $password_hash = password_hash($new_password, PASSWORD_BCRYPT);

        $sql = "UPDATE users SET password = ? WHERE idUser = ?";
        $stmt = $this->pdo->prepare($sql);

        try {
            return $stmt->execute([$password_hash, $id]);
        } catch (PDOException $e) {
            error_log("Database Error: " . $e->getMessage());
            return false;
        }
    }

    /**
     * Ενημερώνει τα στοιχεία του προφίλ ενός χρήστη.
     */
    public function updateProfile(int $id, array $data): bool
    {
        // Έλεγχος αν το νέο email χρησιμοποιείται ήδη από άλλον χρήστη
        $stmt = $this->pdo->prepare("SELECT idUser FROM users WHERE
mail = ? AND idUser != ?");
        $stmt->execute([$data['mail'], $id]);
        if ($stmt->fetch()) {
            flash('danger', 'Αυτή η διεύθυνση email χρησιμοποιείται ήδη
από άλλον λογαριασμό.');
            return false;
        }

        $sql = "UPDATE users
                SET firstname = ?, lastname = ?, mail = ?, phone = ?,
company = ?, school = ?
                WHERE idUser = ?";
```

```
        $stmt = $this->pdo->prepare($sql);

        return $stmt->execute([
            $data['firstname'],
            $data['lastname'],
            $data['mail'],
            $data['phone'],
            $data['company'] ?? null,
            $data['school'] ?? null,
            $id
        ]);
    }

    /**
     * Αλλάζει την κατάσταση (active/inactive) ενός χρήστη.
     * @param int $id Το ID του χρήστη.
     * @return array Έναν πίνακα με 'success' (true/false) και
'new_status' (0 ή 1).
     */
    public function toggleActiveStatus(int $id): array
    {
        // Βρίσκουμε την τρέχουσα κατάσταση
        $currentUser = $this->findById($id);
        if (!$currentUser) {
            return ['success' => false];
        }

        // Αντιστρέφουμε την κατάσταση
        $newStatus = 1 - (int)$currentUser['active'];

        $sql = "UPDATE users SET active = ? WHERE idUser = ?";
        $stmt = $this->pdo->prepare($sql);

        try {
            $success = $stmt->execute([$newStatus, $id]);
            return [
                'success' => $success,
                'new_status' => $newStatus
            ];
        } catch (PDOException $e) {
            error_log("Database Error: " . $e->getMessage());
            return ['success' => false];
        }
    }

    /**
     * Επιστρέφει όλους τους διαθέσιμους ρόλους από τη βάση δεδομένων.
```

```
     */
    public function getAllRoles()
    {
        $stmt = $this->pdo->query("SELECT * FROM roles ORDER BY idRole
ASC");
        return $stmt->fetchAll();
    }

    /**
     * Ενημερώνει τα στοιχεία και τον ρόλο ενός χρήστη (από τον admin).
     */
    public function updateUserAsAdmin(int $id, array $data): bool
    {
        $currentUser = $this->findById($id);
        if (!$currentUser) {
            flash('danger', 'Ο χρήστης δεν βρέθηκε.');
            return false;
        }

        if ($id === $_SESSION['user_id'] && $currentUser['idRole'] == 1
&& $data['idRole'] != 1) {
            flash('danger', 'Δεν μπορείτε να αφαιρέσετε τον ρόλο του
διαχειριστή από τον δικό σας λογαριασμό.');
            return false;
        }

        // Αποτρέπουμε την αφαίρεση του administrator role από τον
τελευταίο admin
        $stmt = $this->pdo->prepare("SELECT COUNT(*) as admin_count
FROM users WHERE idRole = 1");
        $stmt->execute();
        $adminCount = $stmt->fetchColumn();

        if ($currentUser['idRole'] == 1 && $adminCount <= 1 &&
$data['idRole'] != 1) {
             flash('danger', 'Δεν μπορείτε να αφαιρέσετε τον ρόλο του
διαχειριστή από τον τελευταίο εναπομείναντα διαχειριστή.');
             return false;
        }

        $sql = "UPDATE users SET firstname = ?, lastname = ?, mail = ?,
phone = ?, company = ?, school = ?, idRole = ? WHERE idUser = ?";
        $stmt = $this->pdo->prepare($sql);

        try {
            return $stmt->execute([
                $data['firstname'],
                $data['lastname'],
```

```
                $data['mail'],
                $data['phone'],
                $data['company'] ?? null,
                $data['school'] ?? null,
                $data['idRole'],
                $id
            ]);
        } catch (PDOException $e) {
            if ($e->errorInfo[1] == 1062) {
                flash('danger', 'Αυτή η διεύθυνση email χρησιμοποιείται
ήδη από άλλον λογαριασμό.');
            } else {
                flash('danger', 'Παρουσιάστηκε ένα σφάλμα κατά την
ενημέρωση.');
            }
            error_log("Database Error: " . $e->getMessage());
            return false;
        }
    }
}
```

Αρχείο Ελεγκτή (`app/controllers/UserController.php`)

```
<?php

class UserController
{
    private $userModel;
    private $logModel;

    public function __construct(User $userModel, ActivityLog $logModel)
    {
        $this->userModel = $userModel;
        $this->logModel = $logModel;
    }

    /**
     * Εμφανίζει τη λίστα όλων των χρηστών.
     */
    public function index()
    {
        // Μόνο ο super admin μπορεί να δει τη λίστα των χρηστών.
        if (!is_super_admin()) {
            http_response_code(403);
            require_once BASE_PATH . '/views/errors/403.php';
            exit();
        }

        $users = $this->userModel->getAll();
```

```
        require_once BASE_PATH . '/views/users/index.php';
    }

    /**
     * Κάνει επαναφορά τον κωδικό ενός χρήστη στον προκαθορισμένο.
     */
    public function resetPassword(int $id)
    {
        if (!is_super_admin()) {
            http_response_code(403);
            require_once BASE_PATH . '/views/errors/403.php';
            exit();
        }

        $user = $this->userModel->findById($id);

        if (!$user) {
            flash('danger', 'Ο χρήστης δεν βρέθηκε.');
            header('Location: ' . url('/users'));
            exit();
        }

        $newPassword = 'Museapp1!';
        $success = $this->userModel->updatePassword($id, $newPassword);

        if ($success) {
            $fullName = $user['firstname'] . ' ' . $user['lastname'];
            flash('success', "Ο κωδικός του χρήστη {$fullName}
επαναφέρθηκε με επιτυχία σε: <strong>Museapp1!</strong>", true);

            // --- ΠΡΟΣΘΗΚΗ LOG ---
            $adminUsername = $_SESSION['username'];
            $this->logModel->logAction(
                'ADMIN_USER_RESET_PASS',
                "Ο διαχειριστής '{$adminUsername}' επανέφερε τον κωδικό
του χρήστη '{$user['username']}'.",
                $_SESSION['user_id']
            );
            // --- ΤΕΛΟΣ LOG ---

        } else {
            flash('danger', 'Παρουσιάστηκε σφάλμα κατά την επαναφορά
του κωδικού.');
        }

        header('Location: ' . url('/users'));
        exit();
    }
```

```
    /**
     * Εναλλάσσει την κατάσταση ενός χρήστη (ενεργός/ανενεργός).
     */
    public function toggleStatus(int $id)
    {
        if ($id === $_SESSION['user_id']) {
            flash('danger', 'Δεν μπορείτε να απενεργοποιήσετε τον δικό
σας λογαριασμό.');
            header('Location: ' . url('/users'));
            exit();
        }

        if (!is_super_admin() || $_SERVER['REQUEST_METHOD'] !== 'POST')
{
            http_response_code(403);
            require_once BASE_PATH . '/views/errors/403.php';
            exit();
        }

        $user = $this->userModel->findById($id);
        if (!$user) {
            flash('danger', 'Ο χρήστης δεν βρέθηκε.');
            header('Location: ' . url('/users'));
            exit();
        }

        $result = $this->userModel->toggleActiveStatus($id);

        if ($result['success']) {
            $fullName = $user['firstname'] . ' ' . $user['lastname'];
            $statusText = $result['new_status'] == 1 ? "ενεργοποίησε" :
"απενεργοποίησε";
            $message = "Ο χρήστης {$fullName} {$statusText} με
επιτυχία.";
            flash('success', $message);

            // --- ΠΡΟΣΘΗΚΗ LOG ---
            $adminUsername = $_SESSION['username'];
            $this->logModel->logAction(
                'ADMIN_USER_TOGGLE_STATUS',
                "Ο διαχειριστής '{$adminUsername}' {$statusText} τον
χρήστη '{$user['username']}'.",
                $_SESSION['user_id']
            );
            // --- ΤΕΛΟΣ LOG ---

        } else {
```

```
            flash('danger', 'Παρουσιάστηκε σφάλμα κατά την αλλαγή
κατάστασης του χρήστη.');
        }

        header('Location: ' . url('/users'));
        exit();
    }

    /**
     * Εμφανίζει τη φόρμα επεξεργασίας για έναν χρήστη.
     */
    public function edit(int $id)
    {
        if (!is_super_admin()) {
            http_response_code(403);
            require_once BASE_PATH . '/views/errors/403.php';
            exit();
        }

        $user = $this->userModel->findById($id);
        if (!$user) {
            flash('danger', 'Ο χρήστης δεν βρέθηκε.');
            header('Location: ' . url('/users'));
            exit();
        }

        $roles = $this->userModel->getAllRoles();

        require_once BASE_PATH . '/views/users/edit.php';
    }

    /**
     * Ενημερώνει τα στοιχεία ενός χρήστη μετά την υποβολή της φόρμας.
     */
    public function update(int $id)
    {
        if (!is_super_admin() || $_SERVER['REQUEST_METHOD'] !== 'POST')
{
            http_response_code(403);
            require_once BASE_PATH . '/views/errors/403.php';
            exit();
        }

        // --- Παίρνουμε τα τρέχοντα στοιχεία του χρήστη ΠΡΙΝ την
αλλαγή ---
        $targetUser = $this->userModel->findById($id);
        if (!$targetUser) {
```

```
            flash('danger', 'Ο χρήστης που προσπαθήσατε να
επεξεργαστείτε δεν υπάρχει.');
            header('Location: ' . url('/users'));
            exit();
        }

        // --- Validation ---
        $errors = [];
        if (empty($_POST['firstname'])) { $errors[] = "Το πεδίο 'Όνομα'
είναι υποχρεωτικό."; }
        if (empty($_POST['lastname'])) { $errors[] = "Το πεδίο
'Επώνυμο' είναι υποχρεωτικό."; }
        if (empty($_POST['mail']) || !filter_var($_POST['mail'],
FILTER_VALIDATE_EMAIL)) { $errors[] = "Το πεδίο 'Email' δεν είναι
έγκυρο."; }
        if (empty($_POST['phone'])) { $errors[] = "Το πεδίο 'Τηλέφωνο'
είναι υποχρεωτικό."; }
        if (empty($_POST['idRole'])) { $errors[] = "Πρέπει να επιλέξετε
έναν ρόλο."; }

        if (!empty($errors)) {
            $error_message = "<ul>";
            foreach ($errors as $error) { $error_message .= "<li>" .
$error . "</li>"; }
            $error_message .= "</ul>";

            flash('danger', $error_message, true);
            header('Location: ' . url('/users/edit/' . $id));
            exit();
        }

        // --- Αν όλα πήγαν καλά, προχωράμε στην ενημέρωση ---
        $data = [
            'firstname' => trim($_POST['firstname']),
            'lastname' => trim($_POST['lastname']),
            'mail' => trim($_POST['mail']),
            'phone' => trim($_POST['phone']),
            'company' => trim($_POST['company']) ?: null,
            'school' => trim($_POST['school']) ?: null,
            'idRole' => (int)$_POST['idRole']
        ];

        $success = $this->userModel->updateUserAsAdmin($id, $data);

        if ($success) {
            // --- Χρησιμοποιούμε το όνομα που πήραμε για το μήνυμα ---
```

```
            flash('success', "Τα στοιχεία του χρήστη <strong>" .
htmlspecialchars($targetUser['username']) . "</strong> ενημερώθηκαν με
επιτυχία.", true);

            // --- LOGGING ---
            $adminUsername = $_SESSION['username'];
            $this->logModel->logAction(
                'ADMIN_USER_UPDATE',
                "Ο διαχειριστής '{$adminUsername}' ενημέρωσε τα
στοιχεία του χρήστη '{$targetUser['username']}'.",
                $_SESSION['user_id']
            );
        }

        header('Location: ' . url('/users'));
        exit();
    }

    /**
     * Εμφανίζει τη φόρμα δημιουργίας νέου χρήστη.
     */
    public function create()
    {
        if (!is_super_admin()) {
            http_response_code(403);
            require_once BASE_PATH . '/views/errors/403.php';
            exit();
        }

        // Παίρνουμε τους ρόλους για να τους εμφανίσουμε στο dropdown
        $roles = $this->userModel->getAllRoles();
        require_once BASE_PATH . '/views/users/create.php';
    }

    /**
     * Αποθηκεύει τον νέο χρήστη στη βάση δεδομένων.
     */
    public function store()
    {
        if (!is_super_admin() || $_SERVER['REQUEST_METHOD'] !== 'POST')
{
            http_response_code(403);
            require_once BASE_PATH . '/views/errors/403.php';
            exit();
        }

        // --- Validation ---
        $errors = [];
```

```
        if (empty($_POST['username'])) { $errors[] = "Το πεδίο
'Username' είναι υποχρεωτικό."; }
        if (empty($_POST['firstname'])) $errors[] = "Το πεδίο 'Όνομα'
είναι υποχρεωτικό.";
        if (empty($_POST['lastname'])) $errors[] = "Το πεδίο 'Επώνυμο'
είναι υποχρεωτικό.";
        if (empty($_POST['mail']) || !filter_var($_POST['mail'],
FILTER_VALIDATE_EMAIL)) $errors[] = "Το πεδίο 'Email' δεν είναι
έγκυρο.";
        if (empty($_POST['phone'])) $errors[] = "Το πεδίο 'Τηλέφωνο'
είναι υποχρεωτικό.";
        if (empty($_POST['idRole'])) $errors[] = "Πρέπει να επιλέξετε
έναν ρόλο.";

        if (!empty($errors)) {
            $error_message = "<ul>";
            foreach ($errors as $error) { $error_message .= "<li>" .
$error . "</li>"; }
            $error_message .= "</ul>";

            flash('danger', $error_message, true);
            header('Location: ' . url('/users/create'));
            exit();
        }

        $data = [
            'username'  => trim($_POST['username']),
            'firstname' => trim($_POST['firstname']),
            'lastname'  => trim($_POST['lastname']),
            'mail'      => trim($_POST['mail']),
            'phone'     => trim($_POST['phone']),
            'company'   => trim($_POST['company']) ?: null,
            'school'    => trim($_POST['school']) ?: null,
            'idRole'    => (int)$_POST['idRole'],
            'password'  => 'Museapp1!' // Ο προκαθορισμένος κωδικός
        ];

        $success = $this->userModel->create($data, true); // true =
isAdminAction

        if ($success) {
            flash('success', "Ο χρήστης
<strong>{$data['username']}</strong> δημιουργήθηκε με επιτυχία.<br>Ο
προσωρινός κωδικός είναι: <strong>Museapp1!</strong>", true);
            header('Location: ' . url('/users'));
            $adminUsername = $_SESSION['username'];
            $this->logModel->logAction(
                'ADMIN_USER_CREATE',
```

```
                "Ο διαχειριστής '{$adminUsername}' δημιούργησε τον νέο
χρήστη '{$data['username']}'.",
                $_SESSION['user_id']
            );
        } else {
            // Το Model έχει ήδη βάλει το κατάλληλο flash message
            header('Location: ' . url('/users/create'));
        }
        exit();
    }
}
```

Αρχεία Προβολής (app/views/users/)

`index.php`

```
<?php require_once BASE_PATH . '/views/includes/header.inc.php'; ?>

<!-- DataTables CSS -->
<link href="<?=
asset('assets/vendor/datatables/dataTables.bootstrap4.min.css') ?>"
rel="stylesheet">

<!-- Begin Page Content -->
<div class="container-fluid">

    <!-- Page Heading -->
    <div class="d-sm-flex align-items-center justify-content-between
mb-4">
        <h1 class="h3 mb-2 text-gray-800">Διαχείριση Χρηστών</h1>
            <a href="<?= url('/users/create') ?>" class="btn btn-
primary btn-sm">
                <i class="fas fa-plus fa-sm text-white-50"></i>
Δημιουργία νέου χρήστη
            </a>
    </div>
    <p class="mb-4">Λίστα όλων των εγγεγραμμένων χρηστών στην
πλατφόρμα. Μπορείτε να κάνετε αναζήτηση ή ταξινόμηση σε οποιαδήποτε
στήλη.</p>

    <?php flash(); ?>

    <!-- Users DataTable -->
    <div class="card shadow mb-4">
        <div class="card-header py-3">
            <h6 class="m-0 font-weight-bold text-primary">Εγγεγραμμένοι
Χρήστες</h6>
        </div>
        <div class="card-body">
```

```
            <div class="table-responsive">
                <table class="table table-bordered" id="dataTable"
width="100%" cellspacing="0">
                    <thead>
                        <tr>
                            <th>ID</th>
                            <th>Ονοματεπώνυμο</th>
                            <th>Username</th>
                            <th>Email</th>
                            <th>Ρόλος</th>
                            <th>Κατάσταση</th>
                            <th>Ενέργειες</th>
                        </tr>
                    </thead>
                    <tbody>
                        <?php foreach ($users as $user): ?>
                            <tr>
                                <td><?=
htmlspecialchars($user['idUser']) ?></td>
                                <td><?=
htmlspecialchars($user['lastname'] . ' ' . $user['firstname']) ?></td>
                                <td><?=
htmlspecialchars($user['username']) ?></td>
                                <td><?= htmlspecialchars($user['mail'])
?></td>
                                <td>
                                    <?php
                                        $roleName =
htmlspecialchars($user['roleName']);
                                        $badgeClass = '';
                                        switch ($roleName) {
                                            case 'administrator':
                                                $badgeClass = 'badge-
danger';
                                                break;
                                            case 'tour_manager':
                                                $badgeClass = 'badge-
success';
                                                break;
                                            case 'venue_manager':
                                                $badgeClass = 'badge-
primary';
                                                break;
                                            default:
                                                $badgeClass = 'badge-
secondary';
                                        }
                                    ?>
```

```
                                    <span class="badge <?= $badgeClass
?>"><?= $roleName ?></span>
                                </td>
                                <td>
                                    <?php if ($user['active']): ?>
                                        <span class="badge badge-
success">Ενεργός</span>
                                    <?php else: ?>
                                        <span class="badge badge-
secondary">Ανενεργός</span>
                                    <?php endif; ?>
                                </td>
                                <td class="text-center">
                                    <?php if ($user['idUser'] ===
$_SESSION['user_id']): ?>
                                        <!-- Εμφάνιση απενεργοποιημένων
κουμπιών για τον εαυτό του -->
                                        <a href="#" class="btn btn-info
btn-sm disabled" title="Δεν μπορείτε να επεξεργαστείτε τον εαυτό σας"
aria-disabled="true">
                                            <i class="fas fa-edit"></i>
                                        </a>
                                        <button class="btn btn-
secondary btn-sm" title="Δεν μπορείτε να αλλάξετε την κατάσταση του
εαυτού σας" disabled>
                                            <i class="fas fa-toggle-
on"></i>
                                        </button>
                                        <button type="button"
class="btn btn-warning btn-sm" title="Δεν μπορείτε να επαναφέρετε τον
κωδικό του εαυτού σας από εδώ" disabled>
                                            <i class="fas fa-key"></i>
                                        </button>
                                    <?php else: ?>
                                        <!-- Κανονική εμφάνιση κουμπιών
για τους άλλους χρήστες -->
                                        <a href="<?= url('/users/edit/'
. $user['idUser']) ?>" class="btn btn-info btn-sm" title="Επεξεργασία &
Αλλαγή Ρόλου">
                                            <i class="fas fa-edit"></i>
                                        </a>
                                        <form method="POST" action="<?=
url('/users/toggle/' . $user['idUser']) ?>" class="d-inline">
                                            <?php if ($user['active']):
?>
                                                <button type="submit"
class="btn btn-secondary btn-sm" title="Απενεργοποίηση Χρήστη">
```

```
                                                    <i class="fas fa-
toggle-off"></i>
                                                </button>
                                            <?php else: ?>
                                                <button type="submit"
class="btn btn-success btn-sm" title="Ενεργοποίηση Χρήστη">
                                                    <i class="fas fa-
toggle-on"></i>
                                                </button>
                                            <?php endif; ?>
                                        </form>
                                        <button type="button"
class="btn btn-warning btn-sm" title="Επαναφορά Κωδικού"
                                            data-toggle="modal"
                                            data-
target="#resetPasswordModal"
                                            data-action="<?=
url('/users/reset/' . $user['idUser']) ?>">
                                            <i class="fas fa-key"></i>
                                        </button>
                                    <?php endif; ?>
                                </td>
                            </tr>
                        <?php endforeach; ?>

                        <?php if (empty($users)): ?>
                            <tr>
                                <td colspan="6" class="text-center">Δεν
βρέθηκαν χρήστες.</td>
                            </tr>
                        <?php endif; ?>
                    </tbody>
                </table>
            </div>
        </div>
    </div>
</div>
<!-- /.container-fluid -->

<?php require_once BASE_PATH . '/views/includes/footer.inc.php'; ?>

<!-- Page level plugins for DataTables -->
<script src="<?=
asset('assets/vendor/datatables/jquery.dataTables.min.js')
?>"></script>
<script src="<?=
asset('assets/vendor/datatables/dataTables.bootstrap4.min.js')
?>"></script>
```

```
<!-- Page level custom script -->
<script>
$(document).ready(function() {
  $('#dataTable').DataTable({
      "order": [[ 0, "asc" ]], // Ταξινόμηση με βάση το Ονοματεπώνυμο
(πρώτη στήλη) αύξουσα
      // Παραμετροποίηση μηνυμάτων στα Ελληνικά
      "language": {
            "lengthMenu": "Εμφάνιση _MENU_ εγγραφών ανά σελίδα",
            "zeroRecords": "Δεν βρέθηκαν εγγραφές",
            "info": "Εμφάνιση σελίδας _PAGE_ από _PAGES_",
            "infoEmpty": "Δεν υπάρχουν διαθέσιμες εγγραφές",
            "infoFiltered": "(φιλτραρισμένο από _MAX_ συνολικές
εγγραφές)",
            "search": "Αναζήτηση:",
            "paginate": {
                "first": "Πρώτη",
                "last": "Τελευταία",
                "next": "Επόμενη",
                "previous": "Προηγούμενη"
            }
        }
  });
});
</script>
```

create.php

```
<?php require_once BASE_PATH . '/views/includes/header.inc.php'; ?>

<div class="container-fluid">

    <!-- Page Heading -->
    <div class="d-sm-flex align-items-center justify-content-between
mb-4">
        <h1 class="h3 mb-0 text-gray-800">Δημιουργία Νέου Χρήστη</h1>
        <a href="<?= url('/users') ?>" class="btn btn-secondary btn-
sm">
            <i class="fas fa-arrow-left fa-sm text-white-50"></i>
Ακύρωση
        </a>
    </div>

    <!-- Εμφάνιση Flash Messages -->
    <?php flash(); ?>

    <div class="card shadow mb-4">
        <div class="card-header py-3">
```

```
            <h6 class="m-0 font-weight-bold text-primary">Στοιχεία Νέου
Χρήστη</h6>
        </div>
        <div class="card-body">
            <form method="POST" action="<?= url('/users') ?>">

                <div class="form-row">
                    <div class="form-group col-md-6">
                        <label for="username">Username</label>
                        <input type="text" class="form-control"
id="username" name="username" required>
                    </div>
                     <div class="form-group col-md-6">
                        <label for="idRole">Ρόλος Χρήστη</label>
                        <select class="form-control" id="idRole"
name="idRole" required>
                            <option value="" disabled selected>--
Επιλέξτε ρόλο --</option>
                            <?php foreach ($roles as $role): ?>
                                <option value="<?= $role['idRole'] ?>">
                                    <?=
htmlspecialchars(ucfirst($role['roleName'])) ?>
                                </option>
                            <?php endforeach; ?>
                        </select>
                    </div>
                </div>

                <div class="form-row">
                    <div class="form-group col-md-6">
                        <label for="firstname">Όνομα</label>
                        <input type="text" class="form-control"
id="firstname" name="firstname" required>
                    </div>
                    <div class="form-group col-md-6">
                        <label for="lastname">Επώνυμο</label>
                        <input type="text" class="form-control"
id="lastname" name="lastname" required>
                    </div>
                </div>

                <div class="form-row">
                    <div class="form-group col-md-6">
                        <label for="mail">Email</label>
                        <input type="email" class="form-control"
id="mail" name="mail" required>
                    </div>
                    <div class="form-group col-md-6">
```

```
                        <label for="phone">Τηλέφωνο</label>
                        <input type="text" class="form-control"
id="phone" name="phone" required>
                    </div>
                </div>

                <hr>

                <div class="form-row">
                    <div class="form-group col-md-6">
                        <label for="company">Εταιρεία (για κρατήσεις
αιθουσών)</label>
                        <input type="text" class="form-control"
id="company" name="company">
                    </div>
                    <div class="form-group col-md-6">
                        <label for="school">Σχολείο (για σχολικές
επισκέψεις)</label>
                        <input type="text" class="form-control"
id="school" name="school">
                    </div>
                </div>

                <hr>

                 <div class="alert alert-info">
                    <i class="fas fa-info-circle"></i> Ο
προκαθορισμένος κωδικός πρόσβασης για τον νέο χρήστη θα είναι:
<strong>Museapp1!</strong>
                </div>

                <button type="submit" class="btn btn-primary">
                    <i class="fas fa-plus-circle"></i> Δημιουργία
Χρήστη
                </button>

            </form>
        </div>
    </div>
</div>
<!-- /.container-fluid -->

<?php require_once BASE_PATH . '/views/includes/footer.inc.php'; ?>
```

edit.php

```
<?php require_once BASE_PATH . '/views/includes/header.inc.php'; ?>

<div class="container-fluid">
```

```
    <!-- Page Heading -->
    <div class="d-sm-flex align-items-center justify-content-between
mb-4">
        <h1 class="h3 mb-0 text-gray-800">Επεξεργασία Χρήστη: <?=
htmlspecialchars($user['firstname'] . ' ' . $user['lastname']) ?></h1>
        <a href="<?= url('/users') ?>" class="btn btn-secondary btn-
sm">
            <i class="fas fa-arrow-left fa-sm text-white-50"></i>
Ακύρωση
        </a>
    </div>

    <!-- Εμφάνιση Flash Messages -->
    <?php flash(); ?>

    <div class="card shadow mb-4">
        <div class="card-header py-3">
            <h6 class="m-0 font-weight-bold text-primary">Στοιχεία
Χρήστη</h6>
        </div>
        <div class="card-body">
            <form method="POST" action="<?= url('/users/update/' .
$user['idUser']) ?>">

                <div class="form-row">
                    <div class="form-group col-md-6">
                        <label for="firstname">Όνομα</label>
                        <input type="text" class="form-control"
id="firstname" name="firstname" value="<?=
htmlspecialchars($user['firstname']) ?>" required>
                    </div>
                    <div class="form-group col-md-6">
                        <label for="lastname">Επώνυμο</label>
                        <input type="text" class="form-control"
id="lastname" name="lastname" value="<?=
htmlspecialchars($user['lastname']) ?>" required>
                    </div>
                </div>

                <div class="form-row">
                    <div class="form-group col-md-6">
                        <label for="mail">Email</label>
                        <input type="email" class="form-control"
id="mail" name="mail" value="<?= htmlspecialchars($user['mail']) ?>"
required>
                    </div>
                    <div class="form-group col-md-6">
```

```
                        <label for="phone">Τηλέφωνο</label>
                        <input type="text" class="form-control"
id="phone" name="phone" value="<?= htmlspecialchars($user['phone']) ?>"
required>
                    </div>
                </div>

                <hr>

                <div class="form-row">
                    <div class="form-group col-md-6">
                        <label for="company">Εταιρεία (για κρατήσεις
αιθουσών)</label>
                        <input type="text" class="form-control"
id="company" name="company" value="<?=
htmlspecialchars($user['company'] ?? '') ?>">
                    </div>
                    <div class="form-group col-md-6">
                        <label for="school">Σχολείο (για σχολικές
επισκέψεις)</label>
                        <input type="text" class="form-control"
id="school" name="school" value="<?= htmlspecialchars($user['school']
?? '') ?>">
                    </div>
                </div>

                <hr>

                <div class="form-group">
                    <label for="idRole"><strong>Ρόλος
Χρήστη</strong></label>
                    <select class="form-control" id="idRole"
name="idRole" required>
                        <?php foreach ($roles as $role): ?>
                            <option value="<?= $role['idRole'] ?>" <?=
($user['idRole'] == $role['idRole']) ? 'selected' : '' ?>>
                                <?=
htmlspecialchars(ucfirst($role['roleName'])) ?>
                            </option>
                        <?php endforeach; ?>
                    </select>
                </div>

                <button type="submit" class="btn btn-primary">
                    <i class="fas fa-save"></i> Αποθήκευση Αλλαγών
                </button>

            </form>
```

```
        </div>
    </div>
</div>
<!-- /.container-fluid -->

<?php require_once BASE_PATH . '/views/includes/footer.inc.php'; ?>
```

Γ.4 Διαχείριση Προφίλ

Αρχείο Ελεγκτή

`app/controllers/ProfileController.php`

```
<?php

class ProfileController
{
    private $userModel;
    private $logModel;

    public function __construct(User $userModel, ActivityLog $logModel)
    {
        $this->userModel = $userModel;
        $this->logModel = $logModel;
    }

    /**
     * Εμφανίζει τη φόρμα αλλαγής κωδικού.
     */
    public function showChangePasswordForm()
    {
        require_once BASE_PATH . '/views/profile/change-password.php';
    }

    /**
     * Επεξεργάζεται την υποβολή της φόρμας αλλαγής κωδικού.
     */
    public function updatePassword()
    {
        $userId = $_SESSION['user_id'];
        $currentPassword = $_POST['current_password'];
        $newPassword = $_POST['new_password'];
        $confirmPassword = $_POST['confirm_password'];

        $user = $this->userModel->findById($userId);

        if (!password_verify($currentPassword, $user['password'])) {
            flash('danger', 'Ο τρέχων κωδικός που εισαγάγατε είναι
λανθασμένος.');
            header('Location: ' . url('/profile/change-password'));
```

```
            exit();
        }

        if ($newPassword !== $confirmPassword) {
            flash('danger', 'Οι νέοι κωδικοί δεν ταιριάζουν.');
            header('Location: ' . url('/profile/change-password'));
            exit();
        }

        $errors = [];
        if (strlen($newPassword) < 8) { $errors[] = 'Τουλάχιστον 8
χαρακτήρες.'; }
        if (!preg_match('/[A-Z]/', $newPassword)) { $errors[] =
'Τουλάχιστον ένα κεφαλαίο γράμμα.'; }
        if (!preg_match('/[a-z]/', $newPassword)) { $errors[] =
'Τουλάχιστον ένα μικρό γράμμα.'; }
        if (!preg_match('/[0-9]/', $newPassword)) { $errors[] =
'Τουλάχιστον έναν αριθμό.'; }
        if (!preg_match('/[!@#$%^&*]/', $newPassword)) { $errors[] =
'Τουλάχιστον έναν ειδικό χαρακτήρα (!@#$%^&*).'; }

        if (!empty($errors)) {
            $errorMessage = 'Ο νέος κωδικός δεν πληροί τις
προϋποθέσεις:<ul>';
            foreach ($errors as $error) { $errorMessage .= '<li>' .
$error . '</li>'; }
            $errorMessage .= '</ul>';
            flash('danger', $errorMessage, true);
            header('Location: ' . url('/profile/change-password'));
            exit();
        }

        $success = $this->userModel->updatePassword($userId,
$newPassword);

        if ($success) {
            flash('success', 'Ο κωδικός σας άλλαξε με επιτυχία!');

            // --- ΠΡΟΣΘΗΚΗ LOG ---
            $this->logModel->logAction(
                'PROFILE_PASS_CHANGE',
                "Ο χρήστης '{$user['username']}' άλλαξε τον κωδικό
του.",
                $userId
            );
            // --- ΤΕΛΟΣ LOG ---

            header('Location: ' . url('/dashboard'));
```

```
        } else {
            flash('danger', 'Παρουσιάστηκε ένα σφάλμα. Παρακαλώ
δοκιμάστε ξανά.');
            header('Location: ' . url('/profile/change-password'));
        }
        exit();
    }


    /**
     * Εμφανίζει τη φόρμα επεξεργασίας προφίλ.
     */
    public function showProfileForm()
    {
        $userId = $_SESSION['user_id'];
        $user = $this->userModel->findById($userId);

        // Παίρνουμε τα "παλιά" δεδομένα αν υπάρχουν (από αποτυχημένη
προσπάθεια)
        $old_post = $_SESSION['old_post'] ?? [];
        unset($_SESSION['old_post']);

        if (!$user) {
            flash('danger', 'Δεν ήταν δυνατή η φόρτωση των στοιχείων
σας.');
            header('Location: ' . url('/dashboard'));
            exit();
        }

        require_once BASE_PATH . '/views/profile/edit.php';
    }

    /**
     * Επεξεργάζεται την υποβολή της φόρμας επεξεργασίας προφίλ.
     */
    public function updateProfile()
    {
        $userId = $_SESSION['user_id'];

        $_SESSION['old_post'] = $_POST;

        if (empty($_POST['firstname']) || empty($_POST['lastname']) ||
empty($_POST['mail']) || empty($_POST['phone'])) {
            flash('danger', 'Παρακαλώ συμπληρώστε όλα τα απαιτούμενα
πεδία.');
            header('Location: ' . url('/profile/edit'));
            exit();
        }
```

```
        $data = [
            'firstname' => trim($_POST['firstname']),
            'lastname'  => trim($_POST['lastname']),
            'mail'      => trim($_POST['mail']),
            'phone'     => trim($_POST['phone']),
            'company'   => trim($_POST['company']) ?: null,
            'school'    => trim($_POST['school']) ?: null,
        ];

        $success = $this->userModel->updateProfile($userId, $data);

        if ($success) {
            unset($_SESSION['old_post']);
            flash('success', 'Το προφίλ σας ενημερώθηκε με επιτυχία!');

            // --- ΠΡΟΣΘΗΚΗ LOG ---
            $username = $_SESSION['username']; // Παίρνουμε το username
από το session
            $this->logModel->logAction(
                'PROFILE_UPDATE',
                "Ο χρήστης '{$username}' ενημέρωσε τα στοιχεία του
προφίλ του.",
                $userId
            );
            // --- ΤΕΛΟΣ LOG ---
        }

        header('Location: ' . url('/profile/edit'));
        exit();
    }
}
```

Αρχεία Προβολής

`app/views/profile/edit.php`

```
<?php require_once BASE_PATH . '/views/includes/header.inc.php'; ?>

<!-- Begin Page Content -->
<div class="container-fluid">
    <div class="row justify-content-center">
        <div class="col-lg-8">
            <div class="card shadow mb-4">
                <div class="card-header py-3">
                    <h6 class="m-0 font-weight-bold text-
primary">Επεξεργασία Προφίλ</h6>
                </div>
                <div class="card-body">
                    <?php flash(); ?>
```

```
                    <form action="<?= url('/profile/edit') ?>"
method="POST">
                        <div class="form-row">
                            <div class="form-group col-md-6">
                                <label for="firstname">Όνομα</label>
                                <input type="text" class="form-control"
id="firstname" name="firstname" value="<?=
htmlspecialchars($user['firstname']) ?>" required>
                            </div>
                            <div class="form-group col-md-6">
                                <label for="lastname">Επώνυμο</label>
                                <input type="text" class="form-control"
id="lastname" name="lastname" value="<?=
htmlspecialchars($user['lastname']) ?>" required>
                            </div>
                        </div>
                        <div class="form-row">
                             <div class="form-group col-md-6">
                                <label for="mail">Email</label>
                                <input type="email" class="form-
control" id="mail" name="mail" value="<?=
htmlspecialchars($user['mail']) ?>" required>
                            </div>
                            <div class="form-group col-md-6">
                                <label for="username">Όνομα
Χρήστη</label>
                                <input type="text" class="form-control"
id="username" value="<?= htmlspecialchars($user['username']) ?>"
readonly>
                                <small class="form-text text-muted">Το
όνομα χρήστη δεν μπορεί να αλλάξει μετά την εγγραφή.</small>
                            </div>
                        </div>
                         <div class="form-group">
                            <label for="phone">Τηλέφωνο</label>
                            <input type="text" class="form-control"
id="phone" name="phone" value="<?= htmlspecialchars($user['phone']) ?>"
required>
                        </div>
                        <div class="form-group">
                            <label for="company">Εταιρεία /
Οργανισμός</label>
                            <input type="text" class="form-control"
id="company" name="company" value="<?=
htmlspecialchars($user['company'] ?? '') ?>">
                        </div>
                        <div class="form-group">
                            <label for="school">Σχολείο</label>
```

```
                            <input type="text" class="form-control"
id="school" name="school" value="<?= htmlspecialchars($user['school']
?? '') ?>">
                        </div>

                        <button type="submit" class="btn btn-primary
btn-block col-5 justify-content-center mx-auto">Αποθήκευση
Αλλαγών</button>
                    </form>
                </div>
            </div>
        </div>
    </div>
</div>
<!-- /.container-fluid -->

<?php require_once BASE_PATH . '/views/includes/footer.inc.php'; ?>
```

app/views/profile/change-password.php

```
<?php require_once BASE_PATH . '/views/includes/header.inc.php'; ?>

<!-- Begin Page Content -->
<div class="container-fluid">
    <div class="row justify-content-center">
        <div class="col-lg-6">
            <div class="card shadow mb-4">
                <div class="card-header py-3">
                    <h6 class="m-0 font-weight-bold text-
primary">Αλλαγή Κωδικού Πρόσβασης</h6>
                </div>
                <div class="card-body">
                    <?php flash(); ?>
                    <form action="<?= url('/profile/change-password')
?>" method="POST">
                        <div class="form-group">
                            <label for="current_password">Τρέχων
Κωδικός</label>
                            <input type="password" class="form-control"
id="current_password" name="current_password" required>
                        </div>
                        <hr>
                        <div class="form-group">
                            <label for="new_password">Νέος
Κωδικός</label>
                            <input type="password" class="form-control"
id="new_password" name="new_password" required
                                pattern="(?=.*\d)(?=.*[a-z])(?=.*[A-
Z])(?=.*[!@#$%^&*]).{8,}"
```

```
                                    title="Ο κωδικός πρέπει να περιέχει
τουλάχιστον 8 χαρακτήρες, ένα μικρό γράμμα, ένα κεφαλαίο γράμμα, έναν
αριθμό και έναν ειδικό χαρακτήρα (!@#$%^&*).">
                        </div>
                        <div class="form-group">
                            <label for="confirm_password">Επιβεβαίωση
Νέου Κωδικού</label>
                            <input type="password" class="form-control"
id="confirm_password" name="confirm_password" required>
                        </div>
                        <button type="submit" class="btn btn-primary
btn-block col-5 justify-content-center mx-auto">Αποθήκευση
Κωδικού</button>
                    </form>
                </div>
            </div>
        </div>
    </div>
</div>
<!-- /.container-fluid -->

<script>
document.addEventListener('DOMContentLoaded', function() {
    const newPassword = document.getElementById('new_password');
    const confirmPassword =
document.getElementById('confirm_password');

    function validateNewPassword() {
        if (newPassword.value !== confirmPassword.value) {
            confirmPassword.setCustomValidity("Οι νέοι κωδικοί δεν
ταιριάζουν.");
        } else {
            confirmPassword.setCustomValidity('');
        }
    }
    newPassword.addEventListener('change', validateNewPassword);
    confirmPassword.addEventListener('keyup', validateNewPassword);
});
</script>

<?php require_once BASE_PATH . '/views/includes/footer.inc.php'; ?>
```

Γ.5 Διαχείριση Αιθουσών (Venues)

Αρχεία Μοντέλων

Venue.php

```php
<?php

class Venue
{
    private $pdo;

    public function __construct(PDO $pdo)
    {
        $this->pdo = $pdo;
    }

    /**
     * Επιστρέφει τις διαθέσιμες αίθουσες.
     * @param bool $onlyActive Αν θα φέρει μόνο τις ενεργές (για τις δημόσιες σελίδες).
     */
    public function getAll(bool $onlyActive = false): array
    {
        $sql = "SELECT * FROM venues";
        if ($onlyActive) {
            $sql .= " WHERE active = 1";
        }
        $sql .= " ORDER BY venueName ASC";

        $stmt = $this->pdo->query($sql);
        return $stmt->fetchAll();
    }

    /**
     * Βρίσκει μία αίθουσα με βάση το ID της.
     */
    public function findById(int $id)
    {
        $stmt = $this->pdo->prepare("SELECT * FROM venues WHERE idVenue = ?");
        $stmt->execute([$id]);
        return $stmt->fetch();
    }


    /**
     * Δημιουργεί μία νέα αίθουσα (απλή έκδοση).
     * @return bool True σε επιτυχία, false σε αποτυχία.
     */
    public function create(array $data): bool
    {
        $sql = "INSERT INTO venues (
                    venueName, capacity, description, price,
                    valid_from, valid_until,
```

```
                    monday, tuesday, wednesday, thursday, friday,
saturday, sunday
                ) VALUES (
                    :venueName, :capacity, :description, :price,
                    :valid_from, :valid_until,
                    :monday, :tuesday, :wednesday, :thursday, :friday,
:saturday, :sunday
                )";

        $stmt = $this->pdo->prepare($sql);
        try {
            return $stmt->execute([
                ':venueName' => $data['venueName'],
                ':capacity' => $data['capacity'],
                ':description' => $data['description'],
                ':price' => $data['price'],
                ':valid_from' => empty($data['valid_from']) ? null :
$data['valid_from'],
                ':valid_until' => empty($data['valid_until']) ? null :
$data['valid_until'],
                ':monday' => $data['days']['monday'] ?? 0,
                ':tuesday' => $data['days']['tuesday'] ?? 0,
                ':wednesday' => $data['days']['wednesday'] ?? 0,
                ':thursday' => $data['days']['thursday'] ?? 0,
                ':friday' => $data['days']['friday'] ?? 0,
                ':saturday' => $data['days']['saturday'] ?? 0,
                ':sunday' => $data['days']['sunday'] ?? 0,
            ]);
        } catch (PDOException $e) {
            flash('danger', 'Παρουσιάστηκε ένα απρόβλεπτο σφάλμα κατά
την αποθήκευση.');
            return false;
        }
    }

    /**
     * Δημιουργεί μία νέα αίθουσα και επιστρέφει το ID της.
     * @return int|false Το ID της νέας εγγραφής ή false σε αποτυχία.
     */
    public function createAndGetId(array $data)
    {
        if ($this->create($data)) {
            return $this->pdo->lastInsertId();
        }
        return false;
    }

    /**
```

```
     * Ενημερώνει μία αίθουσα.
     */
    public function update(int $id, array $data): bool
    {
        $sql = "UPDATE venues SET
                    venueName = :venueName, capacity = :capacity,
description = :description, price = :price,
                    valid_from = :valid_from, valid_until =
:valid_until,
                    monday = :monday, tuesday = :tuesday, wednesday =
:wednesday,
                    thursday = :thursday, friday = :friday, saturday =
:saturday, sunday = :sunday
                WHERE idVenue = :idVenue";

        $stmt = $this->pdo->prepare($sql);

        try {
            return $stmt->execute([
                ':venueName' => $data['venueName'],
                ':capacity' => $data['capacity'],
                ':description' => $data['description'],
                ':price' => $data['price'],
                ':valid_from' => empty($data['valid_from']) ? null :
$data['valid_from'],
                ':valid_until' => empty($data['valid_until']) ? null :
$data['valid_until'],
                ':monday' => $data['days']['monday'] ?? 0,
                ':tuesday' => $data['days']['tuesday'] ?? 0,
                ':wednesday' => $data['days']['wednesday'] ?? 0,
                ':thursday' => $data['days']['thursday'] ?? 0,
                ':friday' => $data['days']['friday'] ?? 0,
                ':saturday' => $data['days']['saturday'] ?? 0,
                ':sunday' => $data['days']['sunday'] ?? 0,
                ':idVenue' => $id
            ]);
        } catch (PDOException $e) {
            if ($e->errorInfo[1] == 1062) {
                flash('danger', 'Μια αίθουσα με αυτό το όνομα υπάρχει
ήδη.');
            } else {
                flash('danger', 'Παρουσιάστηκε ένα απρόβλεπτο
σφάλμα.');
            }
            error_log($e->getMessage());
            return false;
        }
    }
```

```
    /**
     * Αλλάζει την κατάσταση (active/inactive) μιας αίθουσας.
     */
    public function toggleActiveStatus(int $id): array
    {
        $currentVenue = $this->findById($id);
        if (!$currentVenue) {
            return ['success' => false];
        }

        $newStatus = 1 - (int)$currentVenue['active'];
        $sql = "UPDATE venues SET active = ? WHERE idVenue = ?";
        $stmt = $this->pdo->prepare($sql);

        try {
            $success = $stmt->execute([$newStatus, $id]);
            return ['success' => $success, 'new_status' => $newStatus];
        } catch (PDOException $e) {
            error_log("Database Error: " . $e->getMessage());
            return ['success' => false];
        }
    }

    /**
     * Διαγράφει μια αίθουσα ΜΟΝΟ αν δεν έχει κρατήσεις.
     */
    public function deleteIfUnused(int $id): bool
    {
        $stmt_check = $this->pdo->prepare("SELECT COUNT(*) FROM
bookVenues WHERE idVenue = ?");
        $stmt_check->execute([$id]);
        if ($stmt_check->fetchColumn() > 0) {
            flash('danger', 'Αυτή η αίθουσα δεν μπορεί να διαγραφεί
οριστικά γιατί υπάρχουν συνδεδεμένες κρατήσεις. Μπορείτε όμως να την
απενεργοποιήσετε.');
            return false;
        }


        try {
            $stmt_delete = $this->pdo->prepare("DELETE FROM venues
WHERE idVenue = ?");
            return $stmt_delete->execute([$id]);
        } catch (PDOException $e) {
            flash('danger', 'Παρουσιάστηκε ένα απρόβλεπτο σφάλμα κατά
τη διαγραφή.');
            error_log($e->getMessage());
```

```
            return false;
        }
    }

    /**
     * Ελέγχει ολοκληρωμένα αν μια συγκεκριμένη αίθουσα είναι διαθέσιμη
σε μια συγκεκριμένη ημερομηνία.
     * @param int $idVenue
     * @param string $date (Y-m-d)
     * @return bool
     */
    public function isVenueAvailableOnDate(int $idVenue, string $date):
bool
    {
        // 1. Βρίσκουμε τα στοιχεία της αίθουσας
        $venue = $this->findById($idVenue);
        if (!$venue || !$venue['active']) {
            return false; // Δεν υπάρχει ή δεν είναι ενεργή
        }

        $dt = new DateTime($date);
        $day_of_week = strtolower($dt->format('l'));

        // 2. Έλεγχος ημέρας εβδομάδας
        if (empty($venue[$day_of_week])) {
            return false;
        }

        // 3. Έλεγχος περιόδου ισχύος
        if (($venue['valid_from'] && $date < $venue['valid_from']) ||
($venue['valid_until'] && $date > $venue['valid_until'])) {
            return false;
        }

        // 4. Έλεγχος για εξαιρέσεις
        // Έλεγχος για γενική αργία (idVenue IS NULL)
        $ex_sql_general = "SELECT 1 FROM venue_exceptions WHERE
exception_date = ? AND idVenue IS NULL LIMIT 1";
        $stmt_ex_general = $this->pdo->prepare($ex_sql_general);
        $stmt_ex_general->execute([$date]);
        if ($stmt_ex_general->fetch()) {
            return false;
        }

        // Έλεγχος για ειδική εξαίρεση για τη συγκεκριμένη αίθουσα
        $ex_sql_specific = "SELECT 1 FROM venue_exceptions WHERE
exception_date = ? AND idVenue = ? LIMIT 1";
        $stmt_ex_specific = $this->pdo->prepare($ex_sql_specific);
```

```
        $stmt_ex_specific->execute([$date, $idVenue]);
        if ($stmt_ex_specific->fetch()) {
            return false;
        }

        // 5. Έλεγχος αν είναι ήδη κλεισμένη (pending ή accepted)
        $booking_sql = "SELECT 1 FROM bookVenues WHERE idVenue = ? AND
date = ? AND idStatus IN (1, 2) LIMIT 1";
        $stmt_booking = $this->pdo->prepare($booking_sql);
        $stmt_booking->execute([$idVenue, $date]);
        if ($stmt_booking->fetch()) {
            return false;
        }

        // Αν πέρασε όλους τους ελέγχους, είναι διαθέσιμη!
        return true;
    }

}
```

Photo.php

```
<?php

class Photo
{
    private $pdo;

    public function __construct(PDO $pdo)
    {
        $this->pdo = $pdo;
    }

    public function findById(int $id)
    {
        $stmt = $this->pdo->prepare("SELECT * FROM photo WHERE idPhoto
= ?");
        $stmt->execute([$id]);
        return $stmt->fetch();
    }

    public function findByVenueId(int $venueId): array
    {
        $stmt = $this->pdo->prepare("SELECT * FROM photo WHERE idVenue
= ? ORDER BY is_primary DESC, idPhoto ASC");
        $stmt->execute([$venueId]);
        return $stmt->fetchAll();
    }
```

```php
    public function create(string $path, int $venueId): bool
    {
        // Έλεγχος αν είναι η πρώτη φωτογραφία που ανεβαίνει για αυτή την αίθουσα
        $stmt_check = $this->pdo->prepare("SELECT COUNT(*) FROM photo WHERE idVenue = ?");
        $stmt_check->execute([$venueId]);
        $isFirstPhoto = ($stmt_check->fetchColumn() == 0);

        // Μετατρέπουμε ρητά το boolean (true/false) σε ακέραιο (1/0)
        $isPrimaryValue = $isFirstPhoto ? 1 : 0;

        $sql = "INSERT INTO photo (photoPath, idVenue, is_primary) VALUES (?, ?, ?)";
        $stmt = $this->pdo->prepare($sql);
        return $stmt->execute([$path, $venueId, $isPrimaryValue]);
    }

    public function setAsPrimary(int $venueId, int $photoId): bool
    {
        $this->pdo->beginTransaction();
        try {
            // 1. Αφαιρούμε το is_primary από όλες τις άλλες φωτογραφίες αυτής της αίθουσας
            $stmt1 = $this->pdo->prepare("UPDATE photo SET is_primary = 0 WHERE idVenue = ?");
            $stmt1->execute([$venueId]);

            // 2. Ορίζουμε τη συγκεκριμένη φωτογραφία ως is_primary = 1
            $stmt2 = $this->pdo->prepare("UPDATE photo SET is_primary = 1 WHERE idPhoto = ? AND idVenue = ?");
            $stmt2->execute([$photoId, $venueId]);

            $this->pdo->commit();
            return true;
        } catch (PDOException $e) {
            $this->pdo->rollBack();
            error_log($e->getMessage());
            return false;
        }
    }

    public function delete(int $id): bool
    {
        $stmt = $this->pdo->prepare("DELETE FROM photo WHERE idPhoto = ?");
        return $stmt->execute([$id]);
    }
```

```
}
```

VenueException.php

```
<?php

class VenueException
{
    private $pdo;
    private $base_sql = "SELECT
                            ve.idException, ve.exception_date,
ve.description, v.venueName
                        FROM venue_exceptions ve
                        LEFT JOIN venues v ON ve.idVenue = v.idVenue";

    public function __construct(PDO $pdo)
    {
        $this->pdo = $pdo;
    }

    /**
     * Επιστρέφει τις μελλοντικές εξαιρέσεις (από σήμερα και μετά).
     */
    public function getFutureExceptions()
    {
        $sql = $this->base_sql . " WHERE ve.exception_date >= CURDATE()
ORDER BY ve.exception_date ASC";
        $stmt = $this->pdo->query($sql);
        return $stmt->fetchAll();
    }

    /**
     * Επιστρέφει τις παρελθοντικές εξαιρέσεις.
     */
    public function getPastExceptions()
    {
        $sql = $this->base_sql . " WHERE ve.exception_date < CURDATE()
ORDER BY ve.exception_date DESC";
        $stmt = $this->pdo->query($sql);
        return $stmt->fetchAll();
    }

    /**
     * Δημιουργεί πολλαπλές νέες εξαιρέσεις.
     */
    public function createMultiple(array $data): bool
    {
        $sql = "INSERT INTO venue_exceptions (exception_date,
description, idVenue)
```

```
                VALUES (:exception_date, :description, :idVenue)";
        $stmt = $this->pdo->prepare($sql);

        $this->pdo->beginTransaction();
        try {
            $venueIds = $data['venues'];
            foreach ($venueIds as $venueId) {
                 $stmt->execute([
                    ':exception_date' => $data['exception_date'],
                    ':description' => $data['description'],
                    ':idVenue' => (int)$venueId
                ]);
            }
            $this->pdo->commit();
            return true;
        } catch (PDOException $e) {
            $this->pdo->rollBack();
            flash('danger', 'Παρουσιάστηκε σφάλμα. Πιθανόν κάποια
εξαίρεση για αυτή την ημέρα/αίθουσα να υπάρχει ήδη.');
            error_log($e->getMessage());
            return false;
        }
    }

    /**
     * Διαγράφει μια εξαίρεση.
     */
    public function delete(int $id): bool
    {
        try {
            $stmt = $this->pdo->prepare("DELETE FROM venue_exceptions
WHERE idException = ?");
            return $stmt->execute([$id]);
        } catch (PDOException $e) {
            flash('danger', 'Παρουσιάστηκε σφάλμα κατά τη διαγραφή της
εξαίρεσης.');
            error_log($e->getMessage());
            return false;
        }
    }

    /**
     * Επιστρέφει όλες τις εξαιρέσεις μέσα σε ένα εύρος ημερομηνιών.
     */
    public function getExceptionsByDateRange(string $start_date, string
$end_date): array
    {
        $sql = "SELECT exception_date, idVenue
```

```
                FROM venue_exceptions
                WHERE exception_date BETWEEN ? AND ?";

        $stmt = $this->pdo->prepare($sql);
        $stmt->execute([$start_date, $end_date]);
        return $stmt->fetchAll();
    }
}
```

Αρχεία Ελεγκτών

VenueController.php

```
<?php

class VenueController
{
    private $venueModel;
    private $photoModel;
    private $logModel;

    public function __construct(Venue $venueModel, Photo $photoModel,
ActivityLog $logModel)
    {
        $this->venueModel = $venueModel;
        $this->photoModel = $photoModel;
        $this->logModel = $logModel;
    }

    /**
     * Εμφανίζει τη λίστα με όλες τις αίθουσες (ενεργές και ανενεργές).
     */
    public function index()
    {
        if (!is_super_admin() && !is_venue_manager()) {
            http_response_code(403);
            require_once BASE_PATH . '/views/errors/403.php';
            exit();
        }
        $venues = $this->venueModel->getAll();
        require_once BASE_PATH . '/views/venues/index.php';
    }

    /**
     * Εμφανίζει τη φόρμα δημιουργίας νέας αίθουσας.
     */
    public function create()
    {
        if (!is_super_admin() && !is_venue_manager()) {
```

```
            http_response_code(403);
            require_once BASE_PATH . '/views/errors/403.php';
            exit();
        }
        require_once BASE_PATH . '/views/venues/create.php';
    }

    /**
     * Αποθηκεύει τη νέα αίθουσα και την προαιρετική πρώτη φωτογραφία.
     */
    public function store()
    {
        if (!is_super_admin() && !is_venue_manager()) {
            http_response_code(403);
            require_once BASE_PATH . '/views/errors/403.php';
            exit();
        }

        if (empty($_POST['venueName']) || empty($_POST['capacity']) ||
empty($_POST['price'])) {
            flash('danger', 'Τα πεδία Όνομα, Χωρητικότητα και Τιμή
είναι υποχρεωτικά.');
            $_SESSION['old_post'] = $_POST;
            header('Location: ' . url('/venues/create'));
            exit();
        }

        $days_map = ['monday', 'tuesday', 'wednesday', 'thursday',
'friday', 'saturday', 'sunday'];
        $days_data = [];
        foreach ($days_map as $day) {
            $days_data[$day] = $_POST['days'][$day] ?? 0;
        }

        $data = [
            'venueName'   => trim($_POST['venueName']),
            'capacity'    => (int)$_POST['capacity'],
            'description' => trim($_POST['description']),
            'price'       => (float)$_POST['price'],
            'valid_from'  => empty(trim($_POST['valid_from'])) ? null :
trim($_POST['valid_from']),
            'valid_until' => empty(trim($_POST['valid_until'])) ? null
: trim($_POST['valid_until']),
            'days'        => $days_data
        ];

        // Δημιουργούμε την αίθουσα και παίρνουμε το ID της
        $newVenueId = $this->venueModel->createAndGetId($data);
```

```
        if ($newVenueId) {
            $adminUsername = $_SESSION['username'];
            $this->logModel->logAction('VENUE_CREATE', "Ο χρήστης
'{$adminUsername}' δημιούργησε την αίθουσα '{$data['venueName']}'.",
$_SESSION['user_id']);

            if (isset($_FILES['photo']) &&
!empty($_FILES['photo']['name']) && $_FILES['photo']['error'] ===
UPLOAD_ERR_OK) {
                $this->uploadPhoto($newVenueId, url('/venues'));
                exit();
            }

            flash('success', "Η αίθουσα '{$data['venueName']}'
δημιουργήθηκε με επιτυχία.");
            header('Location: ' . url('/venues'));

        } else {
            $_SESSION['old_post'] = $_POST;
            header('Location: ' . url('/venues/create'));
        }
        exit();
    }

    /**
     * Εμφανίζει τη φόρμα επεξεργασίας.
     */
    public function edit(int $id)
    {
        if (!is_super_admin() && !is_venue_manager()) {
            http_response_code(403);
            require_once BASE_PATH . '/views/errors/403.php';
            exit();
        }

        $venue = $this->venueModel->findById($id);
        if (!$venue) {
            http_response_code(404);
            require_once BASE_PATH . '/views/errors/404.php';
            exit();
        }

        $photos = $this->photoModel->findByVenueId($id);

        require_once BASE_PATH . '/views/venues/edit.php';
    }
```

```php
    /**
     * Ενημερώνει μια αίθουσα.
     */
    public function update(int $id)
    {
        if (!is_super_admin() && !is_venue_manager()) {
            http_response_code(403);
            require_once BASE_PATH . '/views/errors/403.php';
            exit();
        }

        if (empty($_POST['venueName']) || empty($_POST['capacity']) ||
empty($_POST['price'])) {
            flash('danger', 'Τα πεδία Όνομα, Χωρητικότητα και Τιμή
είναι υποχρεωτικά.');
            header('Location: ' . url('/venues/edit/' . $id));
            exit();
        }

        $days_map = ['monday', 'tuesday', 'wednesday', 'thursday',
'friday', 'saturday', 'sunday'];
        $days_data = [];
        foreach ($days_map as $day) {
            $days_data[$day] = $_POST['days'][$day] ?? 0;
        }

        $data = [
            'venueName'   => trim($_POST['venueName']),
            'capacity'    => (int)$_POST['capacity'],
            'description' => trim($_POST['description']),
            'price'       => (float)$_POST['price'],
            'valid_from'  => empty(trim($_POST['valid_from'])) ? null :
trim($_POST['valid_from']),
            'valid_until' => empty(trim($_POST['valid_until'])) ? null
: trim($_POST['valid_until']),
            'days'        => $days_data
        ];

        $success = $this->venueModel->update($id, $data);

        if ($success) {
            flash('success', "Η αίθουσα '{$data['venueName']}'
ενημερώθηκε με επιτυχία.");

            // LOGGING
            $adminUsername = $_SESSION['username'];
            $this->logModel->logAction(
                'VENUE_UPDATE',
```

```
                "Ο χρήστης '{$adminUsername}' ενημέρωσε την αίθουσα
'{$data['venueName']}'.",
                $_SESSION['user_id']
            );
        }

        header('Location: ' . url('/venues/edit/' . $id));
        exit();
    }

    /**
     * Εναλλάσσει την κατάσταση μιας αίθουσας (ενεργή/ανενεργή).
     */
    public function toggleStatus(int $id)
    {
        if (!is_super_admin() && !is_venue_manager()) {
            http_response_code(403);
            require_once BASE_PATH . '/views/errors/403.php';
            exit();
        }

        $venue = $this->venueModel->findById($id);
        if (!$venue) {
            flash('danger', 'Η αίθουσα δεν βρέθηκε.');
            header('Location: ' . url('/venues'));
            exit();
        }

        $result = $this->venueModel->toggleActiveStatus($id);

        if ($result['success']) {
            $statusText = $result['new_status'] == 1 ? "ενεργοποιήθηκε"
: "απενεργοποιήθηκε";
            flash('success', "Η αίθουσα '{$venue['venueName']}'
{$statusText} με επιτυχία.");

            $adminUsername = $_SESSION['username'];
            $this->logModel->logAction(
                'VENUE_TOGGLE_STATUS',
                "Ο χρήστης '{$adminUsername}' {$statusText} την αίθουσα
'{$venue['venueName']}'.",
                $_SESSION['user_id']
            );
        } else {
            flash('danger', 'Παρουσιάστηκε σφάλμα κατά την αλλαγή
κατάστασης.');
        }
```

```
        header('Location: ' . url('/venues'));
        exit();
    }

    /**
     * Διαγράφει οριστικά μια αίθουσα και τις φωτογραφίες της, ΜΟΝΟ αν
δεν έχει κρατήσεις.
     */
    public function destroyUnused(int $id)
    {
        if (!is_super_admin() && !is_venue_manager()) {
            http_response_code(403);
            require_once BASE_PATH . '/views/errors/403.php';
            exit();
        }

        $venue = $this->venueModel->findById($id);
        if (!$venue) {
            flash('danger', 'Η αίθουσα δεν βρέθηκε.');
            header('Location: ' . url('/venues'));
            exit();
        }

        $photos = $this->photoModel->findByVenueId($id);

        $success = $this->venueModel->deleteIfUnused($id);

        if ($success) {
            // Αν η διαγραφή της αίθουσας από τη βάση πετύχει, τότε
διέγραψε και τα αρχεία
            foreach ($photos as $photo) {
                $filePath = BASE_PATH . '/public/' .
$photo['photoPath'];
                if (file_exists($filePath)) {
                    @unlink($filePath); // Χρησιμοποιούμε @ για να μην
βγάλει warning αν το αρχείο δεν υπάρχει
                }
            }

            flash('success', "Η αίθουσα '{$venue['venueName']}' και οι
φωτογραφίες της διαγράφηκαν οριστικά.");

            // LOGGING
            $adminUsername = $_SESSION['username'];
            $this->logModel->logAction(
                'VENUE_DELETE',
                "Ο χρήστης '{$adminUsername}' διέγραψε οριστικά την
αίθουσα '{$venue['venueName']}'.",
```

```
                $_SESSION['user_id']
            );
        }
        // Αν αποτύχει, το model έχει ήδη βάλει το μήνυμα (π.χ.
"υπάρχουν κρατήσεις")

        header('Location: ' . url('/venues'));
        exit();
    }

    // --- PHOTO MANAGEMENT METHODS ---

    /**
     * Ανεβάζει μια νέα φωτογραφία για μια αίθουσα.
     * @param int $venueId Το ID της αίθουσας.
     * @param string|null $redirectUrl Το URL για ανακατεύθυνση. Αν
είναι null, πάει στο edit.
     */
    public function uploadPhoto(int $venueId, ?string $redirectUrl =
null)
    {
        if (!is_super_admin() && !is_venue_manager()) {
            http_response_code(403);
            require_once BASE_PATH . '/views/errors/403.php';
            exit();
        }

        // 1. Έλεγχος αν υπάρχει η αίθουσα

        $finalRedirectUrl = $redirectUrl ?? url('/venues/edit/' .
$venueId);


        $venue = $this->venueModel->findById($venueId);
        if (!$venue) {
            flash('danger', 'Η αίθουσα δεν βρέθηκε.');
            header('Location: ' . url('/venues'));
            exit();
        }

        // 2. Έλεγχος αν έχει σταλεί αρχείο και αν είναι έγκυρο
        if (isset($_FILES['photo']) && $_FILES['photo']['error'] ===
UPLOAD_ERR_OK) {
            $fileTmpPath = $_FILES['photo']['tmp_name'];
            $fileName = $_FILES['photo']['name'];
            $fileSize = $_FILES['photo']['size'];
            $fileType = $_FILES['photo']['type'];
            $fileNameCmps = explode(".", $fileName);
            $fileExtension = strtolower(end($fileNameCmps));
```

```
            // 3. Δημιουργία μοναδικού ονόματος αρχείου
            $newFileName = md5(time() . $fileName) . '.' .
$fileExtension;

            // 4. Ορισμός φακέλου αποθήκευσης
            $uploadFileDir = BASE_PATH . '/public/uploads/venues/';
            $dest_path = $uploadFileDir . $newFileName;

            // 5. Έλεγχος τύπου αρχείου
            $allowedfileExtensions = ['jpg', 'gif', 'png', 'jpeg'];
            if (in_array($fileExtension, $allowedfileExtensions)) {
                // 6. Μετακίνηση του αρχείου
                if(move_uploaded_file($fileTmpPath, $dest_path)) {
                    // 7. Αποθήκευση στη βάση
                    $db_path = 'uploads/venues/' . $newFileName;
                    $this->photoModel->create($db_path, $venueId);

                    flash('success', 'Η φωτογραφία ανέβηκε με
επιτυχία.');
                    // LOGGING
                    $adminUsername = $_SESSION['username'];
                    $this->logModel->logAction('VENUE_PHOTO_UPLOAD', "Ο
χρήστης '{$adminUsername}' ανέβασε μια νέα φωτογραφία για την αίθουσα
'{$venue['venueName']}'.", $_SESSION['user_id']);

                } else {
                    flash('danger', 'Παρουσιάστηκε σφάλμα κατά τη
μετακίνηση του αρχείου.');
                }
            } else {
                flash('danger', 'Μη επιτρεπτός τύπος αρχείου.
Επιτρέπονται μόνο: ' . implode(', ', $allowedfileExtensions));
            }
        } else {
            flash('danger', 'Δεν επιλέχθηκε αρχείο ή παρουσιάστηκε
σφάλμα κατά το ανέβασμα.');
        }

        header('Location: ' . $finalRedirectUrl);
        exit();
    }

    /**
     * Ορίζει μια φωτογραφία ως την κύρια (primary).
     */
    public function setPrimaryPhoto(int $venueId, int $photoId)
    {
```

```
        if (!is_super_admin() && !is_venue_manager()) {
            http_response_code(403);
            require_once BASE_PATH . '/views/errors/403.php';
            exit();
        }

        $venue = $this->venueModel->findById($venueId); // Για το
logging
        if (!$venue) {
            flash('danger', 'Η αίθουσα δεν βρέθηκε.');
            header('Location: ' . url('/venues'));
            exit();
        }

        $success = $this->photoModel->setAsPrimary($venueId, $photoId);

        if ($success) {
            flash('success', 'Η κύρια φωτογραφία άλλαξε με επιτυχία.');
            // LOGGING
            $adminUsername = $_SESSION['username'];
            $this->logModel->logAction('VENUE_PHOTO_PRIMARY', "Ο
χρήστης '{$adminUsername}' όρισε νέα κύρια φωτογραφία για την αίθουσα
'{$venue['venueName']}'.", $_SESSION['user_id']);
        } else {
            flash('danger', 'Παρουσιάστηκε σφάλμα.');
        }

        header('Location: ' . url('/venues/edit/' . $venueId));
        exit();
    }

    /**
     * Διαγράφει μια φωτογραφία.
     */
    public function deletePhoto(int $venueId, int $photoId)
    {
        if (!is_super_admin() && !is_venue_manager()) {
            http_response_code(403);
            require_once BASE_PATH . '/views/errors/403.php';
            exit();
        }

        $venue = $this->venueModel->findById($venueId); // Για το
logging
        if (!$venue) {
            flash('danger', 'Η αίθουσα δεν βρέθηκε.');
            header('Location: ' . url('/venues'));
            exit();
```

```
        }

        // 1. Βρίσκουμε το μονοπάτι του αρχείου ΠΡΙΝ το διαγράψουμε από
τη βάση
        $photo = $this->photoModel->findById($photoId);
        if ($photo) {
            $filePath = BASE_PATH . '/public/' . $photo['photoPath'];

            // 2. Διαγράφουμε την εγγραφή από τη βάση
            $success = $this->photoModel->delete($photoId);

            if ($success) {
                // 3. Αν η διαγραφή από τη βάση πετύχει, διαγράφουμε
και το φυσικό αρχείο
                if (file_exists($filePath)) {
                    @unlink($filePath);
                }
                flash('success', 'Η φωτογραφία διαγράφηκε με
επιτυχία.');
                // LOGGING
                $adminUsername = $_SESSION['username'];
                $this->logModel->logAction('VENUE_PHOTO_DELETE', "Ο
χρήστης '{$adminUsername}' διέγραψε μια φωτογραφία από την αίθουσα
'{$venue['venueName']}'.", $_SESSION['user_id']);
            } else {
                 flash('danger', 'Παρουσιάστηκε σφάλμα κατά τη
διαγραφή.');
            }
        } else {
            flash('danger', 'Η φωτογραφία δεν βρέθηκε.');
        }

        header('Location: ' . url('/venues/edit/' . $venueId));
        exit();
    }
}
```

VenueExceptionController.php

```
<?php

class VenueExceptionController
{
    private $exceptionModel;
    private $venueModel;
    private $logModel;
```

```
    public function __construct(VenueException $exceptionModel, Venue
$venueModel, ActivityLog $logModel)
    {
        $this->exceptionModel = $exceptionModel;
        $this->venueModel = $venueModel;
        $this->logModel = $logModel;
    }

    /**
     * Εμφανίζει τη λίστα με τις εξαιρέσεις και τη φόρμα προσθήκης.
     */
    public function index()
    {
        if (!is_super_admin() && !is_venue_manager()) {
            http_response_code(403);
            require_once BASE_PATH . '/views/errors/403.php';
            exit();
        }

        // --- Φέρνουμε και ομαδοποιούμε τα δεδομένα ---
        $raw_future = $this->exceptionModel->getFutureExceptions();
        $future_exceptions = $this->groupExceptions($raw_future);

        $raw_past = $this->exceptionModel->getPastExceptions();
        $past_exceptions = $this->groupExceptions($raw_past);

        // Παίρνουμε τις ενεργές αίθουσες για το dropdown της φόρμας
        $venues = $this->venueModel->getAll(true);

        require_once BASE_PATH . '/views/venue-exceptions/index.php';
    }

    /**
     * Αποθηκεύει μια νέα εξαίρεση.
     */
    public function store()
    {
        if (!is_super_admin() && !is_venue_manager()) {
            http_response_code(403);
            require_once BASE_PATH . '/views/errors/403.php';
            exit();
         }

        if (empty($_POST['exception_date']) || empty($_POST['venues']))
{
            flash('danger', 'Η ημερομηνία και τουλάχιστον μία αίθουσα
είναι υποχρεωτικά.');
            header('Location: ' . url('/venue-exceptions'));
```

```
            exit();
        }

        $success = $this->exceptionModel->createMultiple($_POST);

        if ($success) {
            flash('success', 'Η εξαίρεση/ες αποθηκεύτηκε/αν με
επιτυχία.');
            // LOGGING
            $adminUsername = $_SESSION['username'];
            $this->logModel->logAction(
                'VENUE_EXCEPTION_CREATE',
                "Ο χρήστης '{$adminUsername}' πρόσθεσε εξαίρεση/ες
αιθουσών για την ημερομηνία {$_POST['exception_date']}.",
                $_SESSION['user_id']
            );
        }

        header('Location: ' . url('/venue-exceptions'));
        exit();
    }

    /**
     * Διαγράφει μια εξαίρεση.
     */
    public function destroy(int $id)
    {
        if (!is_super_admin() && !is_venue_manager()) {
            http_response_code(403);
            require_once BASE_PATH . '/views/errors/403.php';
            exit();
         }

        $success = $this->exceptionModel->delete($id);

        if ($success) {
            flash('success', 'Η εξαίρεση διαγράφηκε με επιτυχία.');
            // LOGGING
            $adminUsername = $_SESSION['username'];
            $this->logModel->logAction(
                'VENUE_EXCEPTION_DELETE',
                "Ο χρήστης '{$adminUsername}' διέγραψε μια εξαίρεση
αίθουσας (ID: {$id}).",
                $_SESSION['user_id']
            );
        }

        header('Location: ' . url('/venue-exceptions'));
```

```
        exit();
    }

    /**
     * Βοηθητική μέθοδος για την ομαδοποίηση των εξαιρέσεων ανά
ημερομηνία.
     */
    private function groupExceptions(array $raw_data): array
    {
        $grouped_data = [];
        foreach ($raw_data as $ex) {
            $date = $ex['exception_date'];
            if (!isset($grouped_data[$date])) {
                $grouped_data[$date] = ['date' => $date, 'description'
=> $ex['description'], 'items' => []];
            }
            $grouped_data[$date]['items'][] = ['name' =>
$ex['venueName'], 'id' => $ex['idException']];
        }
        return $grouped_data;
    }
}
```

Αρχεία Προβολής

`Views/venues/index`

```
<?php require_once BASE_PATH . '/views/includes/header.inc.php'; ?>

<!-- DataTables CSS -->
<link href="<?=
asset('assets/vendor/datatables/dataTables.bootstrap4.min.css') ?>"
rel="stylesheet">

<!-- Begin Page Content -->
<div class="container-fluid">
    <div class="d-sm-flex align-items-center justify-content-between
mb-4">
        <h1 class="h3 mb-0 text-gray-800">Διαχείριση Αιθουσών</h1>
        <a href="<?= url('/venues/create') ?>" class="btn btn-primary
btn-sm">
            <i class="fas fa-plus fa-sm text-white-50"></i> Νέα αίθουσα
        </a>
    </div>

    <?php flash(); ?>

    <div class="card shadow mb-4">
        <div class="card-header py-3">
```

```
            <h6 class="m-0 font-weight-bold text-primary">Λίστα
Αιθουσών</h6>
        </div>
        <div class="card-body">
            <div class="table-responsive">
                <table class="table table-bordered" id="dataTable"
width="100%" cellspacing="0">
                    <thead>
                        <tr>
                            <th>Όνομα Αίθουσας</th>
                            <th>Χωρητικότητα</th>
                            <th>Τιμή (€)</th>
                            <th>Περίοδος Ισχύος</th>
                            <th class="text-center">Κατάσταση</th>
                            <th class="text-center">Ενέργειες</th>
                        </tr>
                    </thead>
                    <tbody>
                        <?php foreach ($venues as $venue): ?>
                            <tr class="<?= $venue['active'] ? '' :
'table-secondary text-muted' ?>">
                                <td><?=
htmlspecialchars($venue['venueName']) ?></td>
                                <td><?=
htmlspecialchars($venue['capacity']) ?></td>
                                <td><?=
htmlspecialchars(number_format($venue['price'], 2, ',', '.')) ?></td>
                                <td>
                                    <?php
                                        $from = $venue['valid_from'] ?
date('d/m/Y', strtotime($venue['valid_from'])) : '...';
                                        $until = $venue['valid_until']
? date('d/m/Y', strtotime($venue['valid_until'])) : '...';
                                        echo "{$from} - {$until}";
                                    ?>
                                </td>
                                <td class="text-center">
                                    <?php if ($venue['active']): ?>
                                        <span class="badge badge-
success">Ενεργή</span>
                                    <?php else: ?>
                                        <span class="badge badge-
secondary">Ανενεργή</span>
                                    <?php endif; ?>
                                </td>
                                <td class="text-center">
                                    <form method="POST" action="<?=
url('/venues/toggle/' . $venue['idVenue']) ?>" class="d-inline">
```

```
                                        <?php if ($venue['active']): ?>
                                            <button type="submit"
class="btn btn-secondary btn-sm" title="Απενεργοποίηση"><i class="fas
fa-toggle-off"></i></button>
                                        <?php else: ?>
                                            <button type="submit"
class="btn btn-success btn-sm" title="Ενεργοποίηση"><i class="fas fa-
toggle-on"></i></button>
                                        <?php endif; ?>
                                    </form>
                                    <a href="<?= url('/venues/edit/' .
$venue['idVenue']) ?>" class="btn btn-info btn-sm"
title="Επεξεργασία"><i class="fas fa-edit"></i></a>
                                </td>
                            </tr>
                        <?php endforeach; ?>
                    </tbody>
                </table>
            </div>
        </div>
    </div>
</div>
<!-- /.container-fluid -->

<?php require_once BASE_PATH . '/views/includes/footer.inc.php'; ?>

<!-- Page level plugins for DataTables -->
<script src="<?=
asset('assets/vendor/datatables/jquery.dataTables.min.js')
?>"></script>
<script src="<?=
asset('assets/vendor/datatables/dataTables.bootstrap4.min.js')
?>"></script>
<script>
$(document).ready(function() {
    $('#dataTable').DataTable({
      "order": [[ 0, "desc" ]], // Ταξινόμηση με βάση την πρώτη στήλη
(ημερομηνία) φθίνουσα
      // Παραμετροποίηση μηνυμάτων στα Ελληνικά
      "language": {
            "lengthMenu": "Εμφάνιση _MENU_ εγγραφών ανά σελίδα",
            "zeroRecords": "Δεν βρέθηκαν εγγραφές",
            "info": "Εμφάνιση σελίδας _PAGE_ από _PAGES_",
            "infoEmpty": "Δεν υπάρχουν διαθέσιμες εγγραφές",
            "infoFiltered": "(φιλτραρισμένο από _MAX_ συνολικές
εγγραφές)",
            "search": "Αναζήτηση:",
            "paginate": {
```

```
                "first": "Πρώτη",
                "last": "Τελευταία",
                "next": "Επόμενη",
                "previous": "Προηγούμενη"
            }
        }
    });
});
</script>
```

Views/venues-exceptions/index

```
<?php require_once BASE_PATH . '/views/includes/header.inc.php'; ?>

<!-- DataTables CSS -->
<link href="<?=
asset('assets/vendor/datatables/dataTables.bootstrap4.min.css') ?>"
rel="stylesheet">

<!-- Begin Page Content -->
<div class="container-fluid">

    <h1 class="h3 mb-4 text-gray-800">Διαχείριση Εξαιρέσεων & Αργιών
(Αίθουσες)</h1>
    <?php flash(); ?>

    <div class="row">
        <!-- FORM COLUMN -->
        <div class="col-lg-4">
            <div class="card shadow mb-4">
                <div class="card-header py-3">
                    <h6 class="m-0 font-weight-bold text-
primary">Προσθήκη Νέας Εξαίρεσης</h6>
                </div>
                <div class="card-body">
                    <form action="<?= url('/venue-exceptions') ?>"
method="POST">
                        <div class="form-group">
                            <label
for="exception_date">Ημερομηνία</label>
                            <input type="date" class="form-control"
name="exception_date" id="exception_date" required>
                        </div>
                        <div class="form-group">
                            <label for="description">Περιγραφή
(Προαιρετικά)</label>
                            <input type="text" class="form-control"
name="description" id="description" placeholder="π.χ., Κλειστά λόγω
εκδήλωσης">
```

```
                        </div>
                        <div class="form-group">
                            <label>Εφαρμογή σε Αίθουσα</label>
                            <div class="border rounded p-2" style="max-
height: 150px; overflow-y: auto;">
                                <div class="custom-control custom-
checkbox">
                                    <input type="checkbox"
class="custom-control-input" id="check_all_venues">
                                    <label class="custom-control-label"
for="check_all_venues"><strong>-- ΕΠΙΛΟΓΗ ΟΛΩΝ --</strong></label>
                                </div>
                                <hr class="my-2">
                                <?php foreach ($venues as $venue): ?>
                                    <div class="custom-control custom-
checkbox">
                                        <input type="checkbox"
class="custom-control-input venue-checkbox" id="venue_<?=
$venue['idVenue'] ?>" name="venues[]" value="<?= $venue['idVenue'] ?>">
                                        <label class="custom-control-
label" for="venue_<?= $venue['idVenue'] ?>"><?=
htmlspecialchars($venue['venueName']) ?></label>
                                    </div>
                                <?php endforeach; ?>
                            </div>
                        </div>
                        <button type="submit" class="btn btn-primary
btn-block">
                            <i class="fas fa-plus-circle"></i> Προσθήκη
                        </button>
                    </form>
                </div>
            </div>
        </div>

        <!-- TABLES COLUMN -->
        <div class="col-lg-8">
            <!-- FUTURE EXCEPTIONS TABLE -->
            <div class="card shadow mb-4">
                <div class="card-header py-3">
                    <h6 class="m-0 font-weight-bold text-
primary">Επερχόμενες Εξαιρέσεις & Αργίες</h6>
                </div>
                <div class="card-body">
                    <div class="table-responsive">
                        <table class="table table-bordered"
id="futureDataTable" width="100%" cellspacing="0">
                            <thead>
```

```
                                <tr>
                                    <th>Ημερομηνία</th>
                                    <th>Περιγραφή</th>
                                    <th>Ισχύει για</th>
                                </tr>
                            </thead>
                            <tbody>
                                <?php foreach ($future_exceptions as
$group): ?>
                                    <tr>
                                        <td class="align-middle" data-
sort="<?= htmlspecialchars($group['date']) ?>"><strong><?=
htmlspecialchars(date('d/m/Y', strtotime($group['date'])))
?></strong></td>
                                        <td class="align-middle"><?=
htmlspecialchars($group['description'] ?? '') ?></td>
                                        <td>
                                            <?php foreach
($group['items'] as $item): ?>
                                                <div class="d-flex
justify-content-between align-items-center mb-1">
                                                    <span class="badge
badge-info mr-2 flex-grow-1"><?= htmlspecialchars($item['name'])
?></span>
                                                    <button class="btn
btn-danger btn-sm py-0 px-1" data-toggle="modal" data-
target="#deleteConfirmationModal" data-action="<?= url('/venue-
exceptions/delete/' . $item['id']) ?>" title="Διαγραφή εξαίρεσης">
                                                        <i class="fas
fa-times"></i>
                                                    </button>
                                                </div>
                                            <?php endforeach; ?>
                                        </td>
                                    </tr>
                                <?php endforeach; ?>
                            </tbody>
                        </table>
                    </div>
                </div>
            </div>

            <!-- PAST EXCEPTIONS -->
            <div id="accordion">
                <div class="card shadow mb-4">
                    <div class="card-header py-3 d-flex flex-row align-
items-center justify-content-between">
```

```
                        <h6 class="m-0 font-weight-bold text-
secondary"><i class="fas fa-history fa-fw"></i> Ιστορικό
Εξαιρέσεων</h6>
                        <a href="#collapseHistory" class="btn btn-sm
btn-light" data-toggle="collapse" role="button" aria-expanded="false"
aria-controls="collapseHistory"><i class="fas fa-chevron-down"></i></a>
                    </div>
                    <div id="collapseHistory" class="collapse" aria-
labelledby="headingOne" data-parent="#accordion">
                        <div class="card-body">
                            <div class="table-responsive">
                                <table class="table table-bordered"
id="pastDataTable" width="100%" cellspacing="0">
                                    <thead>
                                        <tr>
                                            <th>Ημερομηνία</th>
                                            <th>Περιγραφή</th>
                                            <th>Ισχύει για</th>
                                        </tr>
                                    </thead>
                                    <tbody>
                                        <?php foreach ($past_exceptions
as $group): ?>
                                            <tr class="text-muted">
                                                <td class="align-
middle" data-sort="<?= htmlspecialchars($group['date']) ?>"><strong><?=
htmlspecialchars(date('d/m/Y', strtotime($group['date'])))
?></strong></td>
                                                <td class="align-
middle"><?= htmlspecialchars($group['description'] ?? '') ?></td>
                                                <td>
                                                    <?php foreach
($group['items'] as $item): ?>
                                                        <span
class="badge badge-secondary mr-2 mb-1"><?=
htmlspecialchars($item['name']) ?></span>
                                                    <?php endforeach;
?>
                                                </td>
                                            </tr>
                                        <?php endforeach; ?>
                                    </tbody>
                                </table>
                            </div>
                        </div>
                    </div>
                </div>
```

```
        </div>
    </div>
</div>
<!-- /.container-fluid -->

<?php require_once BASE_PATH . '/views/includes/footer.inc.php'; ?>

<!-- Page level plugins for DataTables -->
<script src="<?=
asset('assets/vendor/datatables/jquery.dataTables.min.js')
?>"></script>
<script src="<?=
asset('assets/vendor/datatables/dataTables.bootstrap4.min.js')
?>"></script>

<script>
$(document).ready(function() {
  // Εφαρμογή DataTables στον πρώτο πίνακα (μελλοντικές)
  $('#futureDataTable').DataTable({
      "order": [[ 0, "asc" ]],
      "language": { "lengthMenu": "Εμφάνιση _MENU_ εγγραφών",
"zeroRecords": "Δεν βρέθηκαν εγγραφές", "info": "Σελίδα _PAGE_ από
_PAGES_", "infoEmpty": "Δεν υπάρχουν διαθέσιμες εγγραφές",
"infoFiltered": "(φιλτραρισμένο από _MAX_)", "search": "Αναζήτηση:",
"paginate": { "first": "Πρώτη", "last": "Τελευταία", "next": "Επόμενη",
"previous": "Προηγούμενη" } }
  });

  // Εφαρμογή DataTables στον δεύτερο πίνακα (παρελθοντικές)
  $('#pastDataTable').DataTable({
      "order": [[ 0, "desc" ]],
      "language": { "lengthMenu": "Εμφάνιση _MENU_ εγγραφών",
"zeroRecords": "Δεν βρέθηκαν εγγραφές", "info": "Σελίδα _PAGE_ από
_PAGES_", "infoEmpty": "Δεν υπάρχουν διαθέσιμες εγγραφές",
"infoFiltered": "(φιλτραρισμένο από _MAX_)", "search": "Αναζήτηση:",
"paginate": { "first": "Πρώτη", "last": "Τελευταία", "next": "Επόμενη",
"previous": "Προηγούμενη" } }
  });

  // Λογική για το checkbox "ΕΠΙΛΟΓΗ ΟΛΩΝ"
  $('#check_all_venues').on('change', function() {
      const isChecked = $(this).is(':checked');
      $('.venue-checkbox').prop('checked', isChecked);
  });
});
</script>
```

Γ.6 Διαχείριση Σχολικών Επισκέψεων (Tours)

Αρχεία Μοντέλων

`Tour.php`

```php
<?php

class Tour
{
    private $pdo;

    public function __construct(PDO $pdo)
    {
        $this->pdo = $pdo;
    }

    /**
     * Επιστρέφει τα διαθέσιμα προγράμματα.
     * @param bool $onlyActive Θα φέρει μόνο τα ενεργά (για τις
δημόσιες σελίδες).
     */
    public function getAll(bool $onlyActive = false)
    {
        $sql = "SELECT * FROM tours";
        if ($onlyActive) {
            $sql .= " WHERE active = 1";
        }
        $sql .= " ORDER BY tourTimeStart ASC";

        $stmt = $this->pdo->query($sql);
        return $stmt->fetchAll();
    }

    /**
     * Βρίσκει ένα πρόγραμμα με βάση το ID του.
     */
    public function findById(int $id)
    {
        $stmt = $this->pdo->prepare("SELECT * FROM tours WHERE idTour =
?");
        $stmt->execute([$id]);
        return $stmt->fetch();
    }

    /**
     * Δημιουργεί ένα νέο πρόγραμμα.
     */
    public function create(array $data): bool
```

```
    {
        $sql = "INSERT INTO tours (
                    tourName, tourTimeStart, tourTimeFinish,
                    valid_from, valid_until,
                    monday, tuesday, wednesday, thursday, friday,
saturday, sunday
                ) VALUES (
                    :tourName, :tourTimeStart, :tourTimeFinish,
                    :valid_from, :valid_until,
                    :monday, :tuesday, :wednesday, :thursday, :friday,
:saturday, :sunday
                )";

        $stmt = $this->pdo->prepare($sql);

        try {
            return $stmt->execute([
                ':tourName' => $data['tourName'],
                ':tourTimeStart' => $data['tourTimeStart'],
                ':tourTimeFinish' => $data['tourTimeFinish'],
                ':valid_from' => empty($data['valid_from']) ? null :
$data['valid_from'],
                ':valid_until' => empty($data['valid_until']) ? null :
$data['valid_until'],
                ':monday' => $data['days']['monday'] ?? 0,
                ':tuesday' => $data['days']['tuesday'] ?? 0,
                ':wednesday' => $data['days']['wednesday'] ?? 0,
                ':thursday' => $data['days']['thursday'] ?? 0,
                ':friday' => $data['days']['friday'] ?? 0,
                ':saturday' => $data['days']['saturday'] ?? 0,
                ':sunday' => $data['days']['sunday'] ?? 0,
            ]);
        } catch (PDOException $e) {
            flash('danger', 'Παρουσιάστηκε ένα απρόβλεπτο σφάλμα κατά
την αποθήκευση.');
            return false;
        }
    }

    /**
     * Ενημερώνει ένα πρόγραμμα.
     */
    public function update(int $id, array $data): bool
    {
        $sql = "UPDATE tours SET
                    tourName = :tourName,
                    tourTimeStart = :tourTimeStart,
                    tourTimeFinish = :tourTimeFinish,
```

```
                    valid_from = :valid_from,
                    valid_until = :valid_until,
                    monday = :monday, tuesday = :tuesday, wednesday =
:wednesday,
                    thursday = :thursday, friday = :friday, saturday =
:saturday, sunday = :sunday
                WHERE idTour = :idTour";

        $stmt = $this->pdo->prepare($sql);

        try {
            return $stmt->execute([
                ':tourName' => $data['tourName'],
                ':tourTimeStart' => $data['tourTimeStart'],
                ':tourTimeFinish' => $data['tourTimeFinish'],
                ':valid_from' => empty($data['valid_from']) ? null :
$data['valid_from'],
                ':valid_until' => empty($data['valid_until']) ? null :
$data['valid_until'],
                ':monday' => $data['days']['monday'] ?? 0,
                ':tuesday' => $data['days']['tuesday'] ?? 0,
                ':wednesday' => $data['days']['wednesday'] ?? 0,
                ':thursday' => $data['days']['thursday'] ?? 0,
                ':friday' => $data['days']['friday'] ?? 0,
                ':saturday' => $data['days']['saturday'] ?? 0,
                ':sunday' => $data['days']['sunday'] ?? 0,
                ':idTour' => $id
            ]);
        } catch (PDOException $e) {
            flash('danger', 'Παρουσιάστηκε ένα απρόβλεπτο σφάλμα κατά
την αποθήκευση.');
            return false;
        }
    }

    /**
     * Αλλάζει την κατάσταση (active/inactive) ενός προγράμματος.
     */
    public function toggleActiveStatus(int $id): array
    {
        $currentTour = $this->findById($id);
        if (!$currentTour) {
            return ['success' => false];
        }

        $newStatus = 1 - (int)$currentTour['active'];
        $sql = "UPDATE tours SET active = ? WHERE idTour = ?";
        $stmt = $this->pdo->prepare($sql);
```

```
        try {
            $success = $stmt->execute([$newStatus, $id]);
            return ['success' => $success, 'new_status' => $newStatus];
        } catch (PDOException $e) {
            error_log("Database Error: " . $e->getMessage());
            return ['success' => false];
        }
    }

    /**
     * Διαγράφει ένα πρόγραμμα ΜΟΝΟ αν δεν έχει κρατήσεις.
     */
    public function deleteIfUnused(int $id): bool
    {
        // Έλεγχος αν υπάρχουν κρατήσεις
        $stmt_check = $this->pdo->prepare("SELECT COUNT(*) FROM
bookTours WHERE idTour = ?");
        $stmt_check->execute([$id]);
        if ($stmt_check->fetchColumn() > 0) {
            flash('danger', 'Αυτό το πρόγραμμα δεν μπορεί να διαγραφεί
οριστικά γιατί υπάρχουν συνδεδεμένες κρατήσεις. Μπορείτε όμως να το
απενεργοποιήσετε.');
            return false;
        }

        // Αν δεν υπάρχουν, προχωράμε στη διαγραφή
        try {
            $stmt_delete = $this->pdo->prepare("DELETE FROM tours WHERE
idTour = ?");
            return $stmt_delete->execute([$id]);
        } catch (PDOException $e) {
            flash('danger', 'Παρουσιάστηκε ένα απρόβλεπτο σφάλμα κατά
τη διαγραφή.');
            return false;
        }
    }

    /**
     * Ελέγχει ολοκληρωμένα αν ένα συγκεκριμένο πρόγραμμα είναι
διαθέσιμο σε μια συγκεκριμένη ημερομηνία.
     * @param int $idTour
     * @param string $date (Y-m-d)
     * @return bool
     */
    public function isTourAvailableOnDate(int $idTour, string $date):
bool
    {
```

```
        // 1. Βρίσκουμε τα στοιχεία του προγράμματος
        $tour = $this->findById($idTour);
        if (!$tour || !$tour['active']) {
            return false; // Δεν υπάρχει ή δεν είναι ενεργό
        }

        $dt = new DateTime($date);
        $day_of_week = strtolower($dt->format('l'));

        // 2. Έλεγχος ημέρας εβδομάδας
        if (empty($tour[$day_of_week])) {
            return false;
        }

        // 3. Έλεγχος περιόδου ισχύος
        if (($tour['valid_from'] && $date < $tour['valid_from']) ||
($tour['valid_until'] && $date > $tour['valid_until'])) {
            return false;
        }

        // 4. Έλεγχος για εξαιρέσεις (απαιτεί το TourException Model)
        $exception_sql = "SELECT 1 FROM tour_exceptions WHERE
exception_date = ? AND idTour = ? LIMIT 1";
        $stmt_ex = $this->pdo->prepare($exception_sql);
        $stmt_ex->execute([$date, $idTour]);
        if ($stmt_ex->fetch()) {
            return false; // Βρέθηκε εξαίρεση
        }


        // Αν πέρασε όλους τους ελέγχους, είναι διαθέσιμο!
        return true;
    }
}
```

TourException.php

```
<?php

class TourException
{
    private $pdo;
    private $base_sql = "SELECT
                            te.idException, te.exception_date,
te.description, t.tourName
                        FROM tour_exceptions te
                        LEFT JOIN tours t ON te.idTour = t.idTour";

    public function __construct(PDO $pdo)
```

```
    {
        $this->pdo = $pdo;
    }

    /**
     * Επιστρέφει τις μελλοντικές εξαιρέσεις (από σήμερα και μετά).
     */
    public function getFutureExceptions()
    {
        $sql = $this->base_sql . " WHERE te.exception_date >= CURDATE()
ORDER BY te.exception_date ASC";
        $stmt = $this->pdo->query($sql);
        return $stmt->fetchAll();
    }

    /**
     * Επιστρέφει τις παρελθοντικές εξαιρέσεις.
     */
    public function getPastExceptions()
    {
        $sql = $this->base_sql . " WHERE te.exception_date < CURDATE()
ORDER BY te.exception_date DESC";
        $stmt = $this->pdo->query($sql);
        return $stmt->fetchAll();
    }

    /**
     * Δημιουργεί νέα εξαίρεση.
     */
    public function createMultiple(array $data): bool
    {
        $sql = "INSERT INTO tour_exceptions (exception_date,
description, idTour)
                    VALUES (:exception_date, :description, :idTour)";
        $stmt = $this->pdo->prepare($sql);

        $this->pdo->beginTransaction();
            try {
                $tourIds = $data['tours'];

                // Απλά κάνουμε loop για κάθε επιλεγμένο tour ID
                foreach ($tourIds as $tourId) {
                    $stmt->execute([
                        ':exception_date' => $data['exception_date'],
                        ':description' => $data['description'],
                        ':idTour' => (int)$tourId
                    ]);
                }
```

```
                $this->pdo->commit();
                return true;
            } catch (PDOException $e) {
                $this->pdo->rollBack();
                flash('danger', 'Παρουσιάστηκε σφάλμα κατά την
αποθήκευση. Πιθανόν κάποια εξαίρεση για αυτή την ημέρα/πρόγραμμα να
υπάρχει ήδη.');
                error_log($e->getMessage());
                return false;
            }
    }

    /**
     * Διαγράφει μια εξαίρεση με βάση το ID της.
     */
    public function delete(int $id): bool
    {
        try {
            $stmt = $this->pdo->prepare("DELETE FROM tour_exceptions
WHERE idException = ?");
            return $stmt->execute([$id]);
        } catch (PDOException $e) {
            flash('danger', 'Παρουσιάστηκε σφάλμα κατά τη διαγραφή της
εξαίρεσης.');
            error_log($e->getMessage());
            return false;
        }
    }

    /**
     * Επιστρέφει όλες τις εξαιρέσεις μέσα σε ένα εύρος ημερομηνιών.
     */
    public function getExceptionsByDateRange(string $start_date, string
$end_date): array
    {
        $sql = "SELECT exception_date, idTour
                FROM tour_exceptions
                WHERE exception_date BETWEEN ? AND ?";

        $stmt = $this->pdo->prepare($sql);
        $stmt->execute([$start_date, $end_date]);
        return $stmt->fetchAll();
    }
}
```

Αρχεία Ελεγκτών

`TourController.php`

```php
<?php

class TourController
{
    private $tourModel;
    private $logModel;

    public function __construct(Tour $tourModel, ActivityLog $logModel)
    {
        $this->tourModel = $tourModel;
        $this->logModel = $logModel;
    }

    /**
     * Εμφανίζει τη λίστα με τις διαθέσιμες περιόδους εκδρομών.
     */
    public function index()
    {
        if (!is_super_admin() && !is_tour_manager()) {
            http_response_code(403);
            require_once BASE_PATH . '/views/errors/403.php';
            exit();
        }
        $tours = $this->tourModel->getAll();
        require_once BASE_PATH . '/views/tours/index.php';
    }

    /**
     * Εμφανίζει τη φόρμα δημιουργίας.
     */
    public function create()
    {
        if (!is_super_admin() && !is_tour_manager()) {
            http_response_code(403);
            require_once BASE_PATH . '/views/errors/403.php';
            exit();
        }
        require_once BASE_PATH . '/views/tours/create.php';
    }

    /**
     * Αποθηκεύει τη νέα περίοδο στη βάση.
     */
    public function store()
    {
        if (!is_super_admin() && !is_tour_manager()) {
            http_response_code(403);
            require_once BASE_PATH . '/views/errors/403.php';
```

```
            exit();
        }

        // --- Validation ---
        $errors = [];
        if (empty($_POST['tourName'])) {
            $errors[] = 'Το όνομα του προγράμματος είναι υποχρεωτικό.';
        }
        if (empty($_POST['tourTimeStart']) ||
empty($_POST['tourTimeFinish'])) {
            $errors[] = 'Οι ώρες έναρξης και λήξης είναι
υποχρεωτικές.';
        }
        if (empty($_POST['days'])) {
            $errors[] = 'Πρέπει να επιλέξετε τουλάχιστον μία ημέρα.';
        }

        if (!empty($errors)) {
            // Αν υπάρχουν σφάλματα, τα εμφανίζουμε και επιστρέφουμε
            $errorMessage = "<ul>";
            foreach ($errors as $error) { $errorMessage .= "<li>" .
$error . "</li>"; }
            $errorMessage .= "</ul>";
            flash('danger', $errorMessage, true);

            $_SESSION['old_post'] = $_POST; // Κρατάμε τα παλιά
δεδομένα για να ξαναγεμίσουμε τη φόρμα
            header('Location: ' . url('/tours/create'));
            exit();
        }

        // Τα δεδομένα από τη φόρμα είναι σωστά. Τα προετοιμάζουμε για
το Model.
        $days_map = ['monday', 'tuesday', 'wednesday', 'thursday',
'friday', 'saturday', 'sunday'];
        $days_data = [];
        foreach ($days_map as $day) {
            $days_data[$day] = $_POST['days'][$day] ?? 0;
        }

        $data = [
            'tourName'       => trim($_POST['tourName']),
            'tourTimeStart'  => $_POST['tourTimeStart'],
            'tourTimeFinish' => $_POST['tourTimeFinish'],
            'valid_from'     => empty(trim($_POST['valid_from'])) ?
null : trim($_POST['valid_from']),
            'valid_until'    => empty(trim($_POST['valid_until'])) ?
null : trim($_POST['valid_until']),
```

```
            'days'           => $days_data
        ];

        $success = $this->tourModel->create($data);

        if ($success) {
            // --- LOGGING ---
            $adminUsername = $_SESSION['username'];
            $this->logModel->logAction(
                'TOUR_CREATE',
                "Ο χρήστης '{$adminUsername}' δημιούργησε το πρόγραμμα
'{$data['tourName']}'.",
                $_SESSION['user_id']
            );
            // ---

            flash('success', "Το πρόγραμμα '{$data['tourName']}'
δημιουργήθηκε με επιτυχία!");
            header('Location: ' . url('/tours'));
        } else {
            // Το model έχει ήδη θέσει το μήνυμα σφάλματος. Κρατάμε τα
παλιά δεδομένα.
            $_SESSION['old_post'] = $_POST;
            header('Location: ' . url('/tours/create'));
        }
        exit();
    }

    /**
     * Εμφανίζει τη φόρμα επεξεργασίας.
     */
    public function edit(int $id)
    {
        if (!is_super_admin() && !is_tour_manager()) {
            http_response_code(403);
            require_once BASE_PATH . '/views/errors/403.php';
            exit();
        }

        $tour = $this->tourModel->findById($id);
        if (!$tour) {
            http_response_code(404);
            require_once BASE_PATH . '/views/errors/404.php';
            exit();
        }

        require_once BASE_PATH . '/views/tours/edit.php';
    }
```

```php
    /**
     * Αποθηκεύει τις αλλαγές από τη φόρμα επεξεργασίας.
     */
    public function update(int $id)
    {
        if (!is_super_admin() && !is_tour_manager()) {
            http_response_code(403);
            require_once BASE_PATH . '/views/errors/403.php';
            exit();
        }

        // --- Validation ---
        $errors = [];
        if (empty($_POST['tourName'])) $errors[] = 'Το όνομα του
προγράμματος είναι υποχρεωτικό.';
        if (empty($_POST['tourTimeStart']) ||
empty($_POST['tourTimeFinish'])) $errors[] = 'Οι ώρες έναρξης και λήξης
είναι υποχρεωτικές.';
        if (empty($_POST['days'])) $errors[] = 'Πρέπει να επιλέξετε
τουλάχιστον μία ημέρα.';

        if (!empty($errors)) {
            $errorMessage = "<ul>";
            foreach ($errors as $error) { $errorMessage .= "<li>" .
$error . "</li>"; }
            $errorMessage .= "</ul>";
            flash('danger', $errorMessage, true);

            header('Location: ' . url('/tours/edit/' . $id));
            exit();
        }

        // Προετοιμασία δεδομένων για το Model
        $days_map = ['monday', 'tuesday', 'wednesday', 'thursday',
'friday', 'saturday', 'sunday'];
        $days_data = [];
        foreach ($days_map as $day) {
            // hidden input
            // Αν το checkbox είναι τσεκαρισμένο, το
$_POST['days'][$day] θα είναι '1'.
            // Αν όχι, θα είναι '0' από το hidden input.
            $days_data[$day] = $_POST['days'][$day] ?? 0;
        }

        $data = [
            'tourName'       => trim($_POST['tourName']),
            'tourTimeStart'  => $_POST['tourTimeStart'],
```

```
            'tourTimeFinish' => $_POST['tourTimeFinish'],
            'valid_from'     => empty(trim($_POST['valid_from'])) ?
null : trim($_POST['valid_from']),
            'valid_until'    => empty(trim($_POST['valid_until'])) ?
null : trim($_POST['valid_until']),
            'days'           => $days_data
        ];

        $success = $this->tourModel->update($id, $data);

        if ($success) {
            // --- LOGGING ---
            $adminUsername = $_SESSION['username'];
            $this->logModel->logAction(
                'TOUR_UPDATE',
                "Ο χρήστης '{$adminUsername}' ενημέρωσε το πρόγραμμα
'{$data['tourName']}'.",
                $_SESSION['user_id']
            );
            // ---

            flash('success', "Το πρόγραμμα '{$data['tourName']}'
ενημερώθηκε με επιτυχία.");
            header('Location: ' . url('/tours'));
        } else {
            // Το model έχει ήδη θέσει το μήνυμα σφάλματος.
            header('Location: ' . url('/tours/edit/' . $id));
        }
        exit();
    }

    /**
     * Εναλλάσσει την κατάσταση ενός προγράμματος (ενεργό/ανενεργό).
     */
    public function toggleStatus(int $id)
    {
        if (!is_super_admin() && !is_tour_manager()) {
            http_response_code(403);
            require_once BASE_PATH . '/views/errors/403.php';
            exit();
        }

        $tour = $this->tourModel->findById($id);
        if (!$tour) {
            flash('danger', 'Το πρόγραμμα δεν βρέθηκε.');
            header('Location: ' . url('/tours'));
            exit();
        }
```

```
        $result = $this->tourModel->toggleActiveStatus($id);

        if ($result['success']) {
            $statusText = $result['new_status'] == 1 ? "ενεργοποιήθηκε"
: "απενεργοποιήθηκε";
            flash('success', "Το πρόγραμμα '{$tour['tourName']}'
{$statusText} με επιτυχία.");

            // LOGGING
            $adminUsername = $_SESSION['username'];
            $this->logModel->logAction(
                'TOUR_TOGGLE_STATUS',
                "Ο χρήστης '{$adminUsername}' {$statusText} το
πρόγραμμα '{$tour['tourName']}'.",
                $_SESSION['user_id']
            );
        } else {
            flash('danger', 'Παρουσιάστηκε σφάλμα κατά την αλλαγή
κατάστασης.');
        }

        header('Location: ' . url('/tours'));
        exit();
    }

    /**
     * Διαγράφει οριστικά ένα πρόγραμμα, ΜΟΝΟ αν δεν έχει κρατήσεις.
     */
    public function destroyUnused(int $id)
    {
        if (!is_super_admin() && !is_tour_manager()) {
            http_response_code(403);
            require_once BASE_PATH . '/views/errors/403.php';
            exit();
        }

        $tour = $this->tourModel->findById($id); // Για το logging
        if (!$tour) {
            flash('danger', 'Το πρόγραμμα δεν βρέθηκε.');
            header('Location: ' . url('/tours'));
            exit();
        }

        $success = $this->tourModel->deleteIfUnused($id);

        if ($success) {
```

```
            flash('success', "Το πρόγραμμα '{$tour['tourName']}'
διαγράφηκε οριστικά.");
             // LOGGING
            $adminUsername = $_SESSION['username'];
            $this->logModel->logAction(
                'TOUR_DELETE',
                "Ο χρήστης '{$adminUsername}' διέγραψε οριστικά το
πρόγραμμα '{$tour['tourName']}'.",
                $_SESSION['user_id']
            );
        }
        // Αν αποτύχει, το model έχει ήδη βάλει το μήνυμα σφάλματος.

        header('Location: ' . url('/tours'));
        exit();
    }

}
```

TourExceptionController.php

```
<?php

class TourExceptionController
{
    private $exceptionModel;
    private $tourModel;
    private $logModel;

    public function __construct(TourException $exceptionModel, Tour
$tourModel, ActivityLog $logModel)
    {
        $this->exceptionModel = $exceptionModel;
        $this->tourModel = $tourModel;
        $this->logModel = $logModel;
    }

    /**
     * Εμφανίζει τη λίστα με τις εξαιρέσεις και τη φόρμα προσθήκης.
     */
    public function index()
    {
        if (!is_super_admin() && !is_tour_manager()) {
            http_response_code(403);
            require_once BASE_PATH . '/views/errors/403.php';
            exit();
        }

        // --- Φέρνουμε τα δεδομένα ξεχωριστά ---
```

```
        $raw_future_exceptions = $this->exceptionModel-
>getFutureExceptions();
        $raw_past_exceptions = $this->exceptionModel-
>getPastExceptions();

        // --- Ομαδοποίηση των ΜΕΛΛΟΝΤΙΚΩΝ εξαιρέσεων ανά ημερομηνία --
-
        $future_exceptions = [];
        foreach ($raw_future_exceptions as $ex) {
            $date = $ex['exception_date'];
            if (!isset($future_exceptions[$date])) {
                $future_exceptions[$date] = ['date' => $date,
'description' => $ex['description'], 'tours' => []];
            }
            $future_exceptions[$date]['tours'][] = ['name' =>
$ex['tourName'], 'id' => $ex['idException']];
        }

        // --- Ομαδοποίηση των ΠΑΡΕΛΘΟΝΤΙΚΩΝ εξαιρέσεων ανά ημερομηνία
---
        $past_exceptions = [];
        foreach ($raw_past_exceptions as $ex) {
            $date = $ex['exception_date'];
            if (!isset($past_exceptions[$date])) {
                $past_exceptions[$date] = ['date' => $date,
'description' => $ex['description'], 'tours' => []];
            }
            $past_exceptions[$date]['tours'][] = ['name' =>
$ex['tourName'], 'id' => $ex['idException']];
        }

        // Παίρνουμε τα ενεργά tours για το dropdown της φόρμας
        $tours = $this->tourModel->getAll(true);

        require_once BASE_PATH . '/views/tour-exceptions/index.php';
    }

    /**
     * Αποθηκεύει μια νέα εξαίρεση.
     */
    public function store()
    {
        if (!is_super_admin() && !is_tour_manager()) {
            http_response_code(403);
            require_once BASE_PATH . '/views/errors/403.php';
            exit();
        }
```

```
        if (empty($_POST['exception_date']) || empty($_POST['tours']))
{
            flash('danger', 'Η ημερομηνία και τουλάχιστον ένα πρόγραμμα
είναι υποχρεωτικά.');
            header('Location: ' . url('/tour-exceptions'));
            exit();
        }

        $success = $this->exceptionModel->createMultiple($_POST);

        if ($success) {
            flash('success', 'Η εξαίρεση/ες αποθηκεύτηκε/αν με
επιτυχία.');
            // LOGGING
            $adminUsername = $_SESSION['username'];
            $this->logModel->logAction(
                'TOUR_EXCEPTION_CREATE',
                "Ο χρήστης '{$adminUsername}' πρόσθεσε εξαίρεση/ες για
την ημερομηνία {$_POST['exception_date']}.",
                $_SESSION['user_id']
            );
        }

        header('Location: ' . url('/tour-exceptions'));
        exit();
    }

    /**
     * Διαγράφει μια εξαίρεση.
     */
    public function destroy(int $id)
    {
        if (!is_super_admin() && !is_tour_manager()) {
            http_response_code(403);
            require_once BASE_PATH . '/views/errors/403.php';
            exit();
        }

        $success = $this->exceptionModel->delete($id);

        if ($success) {
            flash('success', 'Η εξαίρεση διαγράφηκε με επιτυχία.');
            // LOGGING
            $adminUsername = $_SESSION['username'];
            $this->logModel->logAction(
                'TOUR_EXCEPTION_DELETE',
                "Ο χρήστης '{$adminUsername}' διέγραψε μια εξαίρεση
(ID: {$id}).",
```

```
                $_SESSION['user_id']
            );
        }

        header('Location: ' . url('/tour-exceptions'));
        exit();
    }
}
```

Αρχεία Προβολής

`Views/tours/index.php`

```
<?php require_once BASE_PATH . '/views/includes/header.inc.php'; ?>

<!-- DataTables CSS -->
<link href="<?=
asset('assets/vendor/datatables/dataTables.bootstrap4.min.css') ?>"
rel="stylesheet">

<!-- Begin Page Content -->
<div class="container-fluid">
    <?php flash(); ?>
    <div class="card shadow mb-4">
        <div class="card-header py-3">
            <h6 class="m-0 font-weight-bold text-primary">Διαθέσιμα
Προγράμματα</h6>
        </div>
        <div class="card-body">
            <div class="table-responsive">
                <table class="table table-bordered" id="dataTable"
width="100%" cellspacing="0">
                    <thead>
                        <tr>
                            <th>Όνομα</th>
                            <th>Ώρες</th>
                            <th>Ημέρες</th>
                            <th>Περίοδος Ισχύος</th>
                            <th>Κατάσταση</th>
                            <th class="text-center">Ενέργειες</th>
                        </tr>
                    </thead>
                    <tbody>
                        <?php foreach ($tours as $tour): ?>
                            <tr class="<?= $tour['active'] ? '' :
'table-secondary text-muted' ?>">
                                <td><?=
htmlspecialchars($tour['tourName']) ?></td>
```

```
                                <td><strong><?= date('H:i',
strtotime($tour['tourTimeStart'])) ?> - <?= date('H:i',
strtotime($tour['tourTimeFinish'])) ?></strong></td>
                                <td><?php

                                    $days = [];
                                    if ($tour['monday']) $days[] =
'Δε';
                                    if ($tour['tuesday']) $days[] =
'Τρ';
                                    if ($tour['wednesday']) $days[] =
'Τε';
                                    if ($tour['thursday']) $days[] =
'Πε';
                                    if ($tour['friday']) $days[] =
'Πα';
                                    if ($tour['saturday']) $days[] =
'Σα';
                                    if ($tour['sunday']) $days[] =
'Κυ';


                                    echo empty($days) ? '-' :
implode(', ', $days);

                                ?></td>
                                <td>
                                    <?php
                                        $from = $tour['valid_from'] ?
date('d/m/Y', strtotime($tour['valid_from'])) : '...';
                                        $until = $tour['valid_until'] ?
date('d/m/Y', strtotime($tour['valid_until'])) : '...';
                                        echo "{$from} - {$until}";
                                    ?>
                                </td>
                                <td class="text-center">
                                    <?php if ($tour['active']): ?>
                                        <span class="badge badge-
success">Ενεργό</span>
                                    <?php else: ?>
                                        <span class="badge badge-
secondary">Ανενεργό</span>
                                    <?php endif; ?>
                                </td>
                                <td class="text-center">
                                    <form method="POST" action="<?=
url('/tours/toggle/' . $tour['idTour']) ?>" class="d-inline">
                                        <?php if ($tour['active']): ?>
```

```
                                            <button type="submit"
class="btn btn-secondary btn-sm" title="Απενεργοποίηση"><i class="fas
fa-toggle-off"></i></button>
                                        <?php else: ?>
                                            <button type="submit"
class="btn btn-success btn-sm" title="Ενεργοποίηση"><i class="fas fa-
toggle-on"></i></button>
                                        <?php endif; ?>
                                    </form>
                                    <a href="<?= url('/tours/edit/' .
$tour['idTour']) ?>" class="btn btn-info btn-sm" title="Επεξεργασία"><i
class="fas fa-edit"></i></a>
                                    <button class="btn btn-danger btn-
sm"
                                            data-toggle="modal"
                                            data-
target="#deleteConfirmationModal"
                                            data-action="<?=
url('/tours/delete-unused/' . $tour['idTour']) ?>"
                                            title="Οριστική Διαγραφή
(μόνο αν δεν έχει κρατήσεις)">
                                        <i class="fas fa-trash"></i>
                                    </button>
                                </td>
                            </tr>
                        <?php endforeach; ?>
                    </tbody>
                </table>
            </div>
        </div>
    </div>
</div>
<!-- /.container-fluid -->

<?php require_once BASE_PATH . '/views/includes/footer.inc.php'; ?>

<!-- Page level plugins for DataTables -->
<script src="<?=
asset('assets/vendor/datatables/jquery.dataTables.min.js')
?>"></script>
<script src="<?=
asset('assets/vendor/datatables/dataTables.bootstrap4.min.js')
?>"></script>

<!-- Page level custom script -->
<script>
$(document).ready(function() {
  $('#dataTable').DataTable({
```

```
      "order": [[ 0, "asc" ]], // Ταξινόμηση με βάση την ώρα έναρξης
      "language": { "lengthMenu": "Εμφάνιση _MENU_ εγγραφών ανά
σελίδα",
            "zeroRecords": "Δεν βρέθηκαν εγγραφές",
            "info": "Εμφάνιση σελίδας _PAGE_ από _PAGES_",
            "infoEmpty": "Δεν υπάρχουν διαθέσιμες εγγραφές",
            "infoFiltered": "(φιλτραρισμένο από _MAX_ συνολικές
εγγραφές)",
            "search": "Αναζήτηση:",
            "paginate": {
                "first": "Πρώτη",
                "last": "Τελευταία",
                "next": "Επόμενη",
                "previous": "Προηγούμενη"
            }
        }
  });
});
</script>
```

Views/tour-exceptions/index.php

```
<?php require_once BASE_PATH . '/views/includes/header.inc.php'; ?>

<!-- DataTables CSS -->
<link href="<?=
asset('assets/vendor/datatables/dataTables.bootstrap4.min.css') ?>"
rel="stylesheet">

<!-- Begin Page Content -->
<div class="container-fluid">

    <h1 class="h3 mb-4 text-gray-800">Διαχείριση Εξαιρέσεων & Αργιών
(Σχ. Επισκέψεις)</h1>
    <?php flash(); ?>

    <div class="row">
        <!-- FORM COLUMN -->
        <div class="col-lg-4">
            <div class="card shadow mb-4">
                <div class="card-header py-3">
                    <h6 class="m-0 font-weight-bold text-
primary">Προσθήκη Νέας Εξαίρεσης</h6>
                </div>
                <div class="card-body">
                    <form action="<?= url('/tour-exceptions') ?>"
method="POST">
                        <div class="form-group">
```

```
                            <label
for="exception_date">Ημερομηνία</label>
                            <input type="date" class="form-control"
name="exception_date" id="exception_date" required>
                        </div>
                        <div class="form-group">
                            <label for="description">Περιγραφή
(Προαιρετικά)</label>
                            <input type="text" class="form-control"
name="description" id="description" placeholder="π.χ., Εθνική Αργία,
Συντήρηση">
                        </div>
                        <div class="form-group">
                            <label>Εφαρμογή σε Πρόγραμμα</label>
                            <div class="border rounded p-2" style="max-
height: 150px; overflow-y: auto;">
                                <div class="custom-control custom-
checkbox">
                                    <input type="checkbox"
class="custom-control-input" id="check_all_tours">
                                    <label class="custom-control-label"
for="check_all_tours"><strong>-- ΕΠΙΛΟΓΗ ΟΛΩΝ --</strong></label>
                                </div>
                                <hr class="my-2">
                                <?php foreach ($tours as $tour): ?>
                                    <div class="custom-control custom-
checkbox">
                                        <input type="checkbox"
class="custom-control-input tour-checkbox" id="tour_<?= $tour['idTour']
?>" name="tours[]" value="<?= $tour['idTour'] ?>">
                                        <label class="custom-control-
label" for="tour_<?= $tour['idTour'] ?>"><?=
htmlspecialchars($tour['tourName']) ?></label>
                                    </div>
                                <?php endforeach; ?>
                            </div>
                        </div>
                        <button type="submit" class="btn btn-primary
btn-block">
                            <i class="fas fa-plus-circle"></i> Προσθήκη
                        </button>
                    </form>
                </div>
            </div>
        </div>

        <!-- TABLES COLUMN -->
        <div class="col-lg-8">
```

```
            <!-- FUTURE EXCEPTIONS TABLE -->
            <div class="card shadow mb-4">
                <div class="card-header py-3">
                    <h6 class="m-0 font-weight-bold text-
primary">Επερχόμενες Εξαιρέσεις & Αργίες</h6>
                </div>
                <div class="card-body">
                    <div class="table-responsive">
                        <table class="table table-bordered"
id="futureDataTable" width="100%" cellspacing="0">
                            <thead>
                                <tr>
                                    <th>Ημερομηνία</th>
                                    <th>Περιγραφή</th>
                                    <th>Ισχύει για</th>
                                </tr>
                            </thead>
                            <tbody>
                                <?php foreach ($future_exceptions as
$group): ?>
                                    <tr>
                                        <td class="align-middle" data-
sort="<?= htmlspecialchars($group['date']) ?>">
                                            <strong><?=
htmlspecialchars(date('d/m/Y', strtotime($group['date']))) ?></strong>
                                        </td>
                                        <td class="align-middle"><?=
htmlspecialchars($group['description'] ?? '') ?></td>
                                        <td>
                                            <?php foreach
($group['tours'] as $tour_info): ?>
                                                <div class="d-flex
justify-content-between align-items-center mb-1">
                                                    <span class="badge
badge-info mr-2 flex-grow-1"><?= htmlspecialchars($tour_info['name'])
?></span>
                                                    <button class="btn
btn-danger btn-sm py-0 px-1"
                                                            data-
toggle="modal"
                                                            data-
target="#deleteConfirmationModal"
                                                            data-
action="<?= url('/tour-exceptions/delete/' . $tour_info['id']) ?>"
                                                            title="Διαγ
ραφή εξαίρεσης">
                                                        <i class="fas
fa-times"></i>
```

```
                                                    </button>
                                                </div>
                                            <?php endforeach; ?>
                                        </td>
                                    </tr>
                                <?php endforeach; ?>
                            </tbody>
                        </table>
                    </div>
                </div>
            </div>

            <!-- PAST EXCEPTIONS ACCORDION -->
            <div id="accordion">
                <div class="card shadow mb-4">
                    <div class="card-header py-3" id="headingOne">
                        <h6 class="m-0 font-weight-bold text-
secondary">
                            <i class="fas fa-history fa-fw"></i>
Ιστορικό Εξαιρέσεων (Παρελθοντικές)

                        <a href="#collapseHistory" class="btn btn-sm
btn-light" data-toggle="collapse" role="button" aria-expanded="false"
aria-controls="collapseHistory">
                            <i class="fas fa-chevron-down"></i>
                        </a>
                        </h6>
                    </div>
                    <div id="collapseHistory" class="collapse" aria-
labelledby="headingOne" data-parent="#accordion">
                        <div class="card-body">
                            <div class="table-responsive">
                                <table class="table table-bordered"
id="pastDataTable" width="100%" cellspacing="0">
                                    <thead>
                                        <tr>
                                            <th>Ημερομηνία</th>
                                            <th>Περιγραφή</th>
                                            <th>Ισχύει για</th>
                                        </tr>
                                    </thead>
                                    <tbody>
                                        <?php foreach ($past_exceptions
as $group): ?>
                                            <tr class="text-muted">
                                                <td class="align-
middle" data-sort="<?= htmlspecialchars($group['date']) ?>">
```

```
                                                    <strong><?=
htmlspecialchars(date('d/m/Y', strtotime($group['date']))) ?></strong>
                                                </td>
                                                <td class="align-
middle"><?= htmlspecialchars($group['description'] ?? '') ?></td>
                                                <td>
                                                    <?php foreach
($group['tours'] as $tour_info): ?>
                                                        <div class="d-
flex justify-content-between align-items-center mb-1">
                                                            <span
class="badge badge-secondary mr-2 flex-grow-1"><?=
htmlspecialchars($tour_info['name']) ?></span>
                                                            <!-- Κουμπί
διαγραφής απενεργοποιημένο -->
                                                            <button
class="btn btn-secondary btn-sm py-0 px-1" disabled>
                                                                <i
class="fas fa-times"></i>
                                                            </button>
                                                        </div>
                                                    <?php endforeach;
?>
                                                </td>
                                            </tr>
                                        <?php endforeach; ?>
                                    </tbody>
                                </table>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>
</div>
<!-- /.container-fluid -->

<?php require_once BASE_PATH . '/views/includes/footer.inc.php'; ?>

<!-- Page level plugins for DataTables -->
<script src="<?=
asset('assets/vendor/datatables/jquery.dataTables.min.js')
?>"></script>
<script src="<?=
asset('assets/vendor/datatables/dataTables.bootstrap4.min.js')
?>"></script>
```

```
<script>
$(document).ready(function() {
  // Εφαρμογή DataTables στον πρώτο πίνακα (μελλοντικές)
  $('#futureDataTable').DataTable({
      "order": [[ 0, "asc" ]], // Ταξινόμηση αύξουσα (η πιο κοντινή
πρώτη)
      "language": {
            "lengthMenu": "Εμφάνιση _MENU_ εγγραφών",
            "zeroRecords": "Δεν βρέθηκαν εγγραφές",
            "info": "Σελίδα _PAGE_ από _PAGES_",
            "infoEmpty": "Δεν υπάρχουν διαθέσιμες εγγραφές",
            "infoFiltered": "(φιλτραρισμένο από _MAX_)",
            "search": "Αναζήτηση:",
            "paginate": { "first": "Πρώτη", "last": "Τελευταία",
"next": "Επόμενη", "previous": "Προηγούμενη" }
        }
  });

  // Εφαρμογή DataTables στον δεύτερο πίνακα (παρελθοντικές)
  $('#pastDataTable').DataTable({
      "order": [[ 0, "desc" ]], // Ταξινόμηση φθίνουσα (η πιο πρόσφατη
παλιά πρώτη)
      "language": {
            "lengthMenu": "Εμφάνιση _MENU_ εγγραφών",
            "zeroRecords": "Δεν βρέθηκαν εγγραφές",
            "info": "Σελίδα _PAGE_ από _PAGES_",
            "infoEmpty": "Δεν υπάρχουν διαθέσιμες εγγραφές",
            "infoFiltered": "(φιλτραρισμένο από _MAX_)",
            "search": "Αναζήτηση:",
            "paginate": { "first": "Πρώτη", "last": "Τελευταία",
"next": "Επόμενη", "previous": "Προηγούμενη" }
        }
  });

  // Λογική για το checkbox "ΕΠΙΛΟΓΗ ΟΛΩΝ"
  $('#check_all_tours').on('change', function() {
      const isChecked = $(this).is(':checked');
      $('.tour-checkbox').prop('checked', isChecked);
  });
});
</script>
```

Γ.7 Σύστημα Κρατήσεων και API

Αρχεία Ελεγκτών

`ApiController.php`

```
<?php
```

```php
class ApiController
{
    private $bookVenueModel;
    private $bookTourModel;
    private $tourModel;
    private $tourExceptionModel;
    private $bookAlertModel;
    private $venueModel;
    private $venueExceptionModel;
    private $userModel;

    public function __construct(BookVenue $bookVenueModel, BookTour
$bookTourModel, Tour $tourModel, TourException $tourExceptionModel,
BookAlert $bookAlertModel, Venue $venueModel, VenueException
$venueExceptionModel, User $userModel)
    {
        $this->bookVenueModel = $bookVenueModel;
        $this->bookTourModel = $bookTourModel;
        $this->tourModel = $tourModel;
        $this->tourExceptionModel = $tourExceptionModel;
        $this->bookAlertModel = $bookAlertModel;
        $this->venueModel = $venueModel;
        $this->venueExceptionModel = $venueExceptionModel;
        $this->userModel = $userModel;
    }

    /**
     * Παρέχει τα "events" για το ημερολόγιο των σχολικών επισκέψεων.
     */
    public function getTourEvents()
    {
        header('Content-Type: application/json');

        $start_date = $_GET['start'] ?? date('Y-m-01');
        $end_date = $_GET['end'] ?? date('Y-m-t');
        $is_manager = is_super_admin() || is_tour_manager();

        // 1. Παίρνουμε τα "ακατέργαστα" δεδομένα
        $active_tours = $this->tourModel->getAll(true);
        $bookings = $this->bookTourModel-
>getAcceptedBookingsByDateRange($start_date, $end_date);
        $exceptions = $this->tourExceptionModel-
>getExceptionsByDateRange($start_date, $end_date);

        // 2. Οργανώνουμε τα δεδομένα σε "χάρτες" για γρήγορη πρόσβαση
        $bookings_by_date = [];
        foreach ($bookings as $booking) {
```

```
            $bookings_by_date[$booking['date']][$booking['idTour']] =
$booking; // Keyed by idTour
        }
        $exceptions_by_date = [];
        foreach ($exceptions as $exception) {
            $exceptions_by_date[$exception['exception_date']][$exceptio
n['idTour']] = true;
        }

        // 3. Ξεκινάμε τη δημιουργία των events για το ημερολόγιο
        $events = [];
        $period = new DatePeriod(new DateTime($start_date), new
DateInterval('P1D'), new DateTime($end_date));

        foreach ($period as $dt) {
            $date_str = $dt->format('Y-m-d');

            $today = new DateTime(); $today->setTime(0,0,0);
            if (!$is_manager && ($dt < $today || $today->diff($dt)-
>days < 6)) {
                continue;
            }

            $day_of_week = strtolower($dt->format('l'));

            $available_tours_today = [];
            $booked_tours_today = [];

            // Βρίσκουμε ποια προγράμματα είναι διαθέσιμα και ποια
κλεισμένα για τη σημερινή ημέρα
            foreach ($active_tours as $tour) {
                // Εφαρμόζουμε όλους τους κανόνες διαθεσιμότητας
                if (empty($tour[$day_of_week])) continue;
                if (($tour['valid_from'] && $date_str <
$tour['valid_from']) || ($tour['valid_until'] && $date_str >
$tour['valid_until'])) continue;
                if
(isset($exceptions_by_date[$date_str][$tour['idTour']])) continue;

                // Αν έφτασε εδώ, το tour είναι θεωρητικά διαθέσιμο
                if
(isset($bookings_by_date[$date_str][$tour['idTour']])) {
                    $booked_tours_today[] = $tour; // Είναι κλεισμένο
                } else {
                    $available_tours_today[] = $tour; // Είναι ελεύθερο
                }
            }
```

```
            $total_slots = count($available_tours_today) +
count($booked_tours_today);
            $booked_count = count($booked_tours_today);

            if ($total_slots > 0) {
                $color = '#28a745'; // Πράσινο
                $title = "{$booked_count}/{$total_slots} Κλεισμένα";
                $is_bookable_user = true;

                if ($booked_count > 0 && $booked_count < $total_slots)
{
                    $color = '#ffc107'; // Πορτοκαλί
                } elseif ($booked_count === $total_slots) {
                    $color = '#dc3545'; // Κόκκινο
                    $is_bookable_user = false;
                }

                $today = new DateTime(); $today->setTime(0,0,0);
                if ($dt < $today || $today->diff($dt)->days < 6) {
                    $is_bookable_user = false;
                }

                $event_data = [
                    'title' => $title,
                    'start' => $date_str,
                    'backgroundColor' => $color,
                    'borderColor' => $color,
                    'allDay' => true,
                    'extendedProps' => [
                        'is_bookable_user' => $is_bookable_user,
                        'is_manager' => $is_manager
                    ]
                ];

                // Αν είναι manager, προσθέτουμε τις επιπλέον
πληροφορίες
                if ($is_manager) {
                    $details = [];
                    // Για τα διαθέσιμα, προσθέτουμε το ID του TOUR
                    foreach ($available_tours_today as $t) {
                        $details[] = [
                            'id' => $t['idTour'],
                            'name' => $t['tourName'],
                            'status' => 'available'
                        ];
                    }
                    // Για τα κλεισμένα, προσθέτουμε το ID του TOUR και
τα στοιχεία της κράτησης
```

```
                    foreach ($booked_tours_today as $t) {
                        $booking_info =
$bookings_by_date[$date_str][$t['idTour']];
                        $school_name = !empty($booking_info['idUser'])
                            ? ($this->userModel-
>findById($booking_info['idUser'])['school'] ?? 'Άγνωστο Σχολείο')
                            : $booking_info['on_behalf_of_school'];
                        $details[] = [
                            'id' => $t['idTour'],
                            'name' => $t['tourName'],
                            'status' => 'booked',
                            'school' => $school_name
                        ];
                    }
                    $event_data['extendedProps']['details'] = $details;
                }

                $events[] = $event_data;
            }
        }

        echo json_encode($events);
        exit();
    }


    /**
     * Επιστρέφει τα ΔΙΑΘΕΣΙΜΑ προγράμματα για μια συγκεκριμένη
ημερομηνία.
     */
    public function getAvailableToursForDate()
    {
        header('Content-Type: application/json');
        $date = $_GET['date'] ?? null;

        if (!$date) {
            echo json_encode(['error' => 'Missing date']);
            exit();
        }

        // 1. Παίρνουμε όλα τα δεδομένα
        $active_tours = $this->tourModel->getAll(true);
        $bookings = $this->bookTourModel-
>getAcceptedBookingsByDateRange($date, $date);
        $exceptions = $this->tourExceptionModel-
>getExceptionsByDateRange($date, $date);

        // 2. Οργανώνουμε τα bookings και τις εξαιρέσεις για εύκολη
πρόσβαση
```

```
        $booked_tour_ids = array_column($bookings, 'idTour');
        $exception_tour_ids = array_column($exceptions, 'idTour');

        // 3. Βρίσκουμε τα διαθέσιμα
        $available_tours = [];
        $day_of_week = strtolower(date('l', strtotime($date)));

        foreach ($active_tours as $tour) {
            // Ελέγχουμε όλους τους κανόνες
            if (empty($tour[$day_of_week])) continue;
            if (($tour['valid_from'] && $date < $tour['valid_from']) ||
($tour['valid_until'] && $date > $tour['valid_until'])) continue;
            if (in_array($tour['idTour'], $exception_tour_ids))
continue;
            if (in_array($tour['idTour'], $booked_tour_ids)) continue;
// <--- Εξαιρούμε τα ήδη κλεισμένα

            // Αν περάσει όλα τα τεστ, είναι διαθέσιμο!
            $available_tours[] = [
                'id' => $tour['idTour'],
                'name' => $tour['tourName'] . ' (' . date('H:i',
strtotime($tour['tourTimeStart'])) . ')'
            ];
        }

        echo json_encode($available_tours);
        exit();
    }


    public function getAlerts()
    {
        header('Content-Type: application/json');

        $alerts = [];
        $unread_count = 0;

        if (is_super_admin()) {
            $alerts = $this->bookAlertModel->getManagerAlerts(true,
true);
            $unread_count = $this->bookAlertModel-
>countUnreadManagerAlerts(true, true);
        } elseif (is_tour_manager()) {
            $alerts = $this->bookAlertModel->getManagerAlerts(true,
false);
            $unread_count = $this->bookAlertModel-
>countUnreadManagerAlerts(true, false);
        } elseif (is_venue_manager()) {
```

```
            $alerts = $this->bookAlertModel->getManagerAlerts(false,
true);
            $unread_count = $this->bookAlertModel-
>countUnreadManagerAlerts(false, true);
        } elseif (is_simple_user()) {
            $alerts = $this->bookAlertModel-
>getUserAlerts($_SESSION['user_id']);
            $unread_count = $this->bookAlertModel-
>countUnreadUserAlerts($_SESSION['user_id']);
        }

        echo json_encode(['alerts' => $alerts, 'unread_count' =>
$unread_count]);
        exit();
    }

    /**
     * Μέθοδος για το "Mark as Read".
     */
    public function markAlertAsRead()
    {
        header('Content-Type: application/json');
        $data = json_decode(file_get_contents('php://input'), true);
        $alertId = $data['id'] ?? null;

        if ($alertId && is_numeric($alertId)) {
            $success = $this->bookAlertModel-
>markAsRead((int)$alertId);
            if ($success) {
                echo json_encode(['status' => 'success']);
            } else {
                echo json_encode(['status' => 'error', 'message' => 'DB
error']);
            }
        } else {
            echo json_encode(['status' => 'error', 'message' =>
'Invalid ID']);
        }
        exit();
    }


public function getVenueEvents()
    {
        header('Content-Type: application/json');

        $start_date = $_GET['start'] ?? date('Y-m-01');
        $end_date = $_GET['end'] ?? date('Y-m-t');
        $current_user_id = $_SESSION['user_id'];
```

```
        $is_manager = is_super_admin() || is_venue_manager();

        // 1. Παίρνουμε όλα τα "ακατέργαστα" δεδομένα
        $active_venues = $this->venueModel->getAll(true);
        $bookings = $this->bookVenueModel-
>getActiveBookingsByDateRange($start_date, $end_date);
        $exceptions = $this->venueExceptionModel-
>getExceptionsByDateRange($start_date, $end_date);

        // 2. Οργανώνουμε τα δεδομένα σε "χάρτες" για γρήγορη πρόσβαση
        $bookings_by_date = [];
        foreach ($bookings as $b) {
$bookings_by_date[$b['date']][$b['idVenue']] = $b; }
        $exceptions_by_date = [];
        foreach ($exceptions as $e) {
$exceptions_by_date[$e['exception_date']][$e['idVenue']] = true; }

        // 3. Ξεκινάμε τη δημιουργία των events για το ημερολόγιο
        $events = [];
        $period = new DatePeriod(new DateTime($start_date), new
DateInterval('P1D'), new DateTime($end_date));
        $today = new DateTime(); $today->setTime(0,0,0);

        foreach ($period as $dt) {
            $date_str = $dt->format('Y-m-d');

            // --- ΑΠΟΚΡΥΨΗ ΗΜΕΡΩΝ ΓΙΑ ΑΠΛΟΥΣ ΧΡΗΣΤΕΣ ---
            if (!$is_manager && ($dt < $today || $today->diff($dt)-
>days < 6)) {
                continue;
            }

            $day_of_week = strtolower($dt->format('l'));

            $available_venues_today = [];
            $booked_venues_today = [];
            $user_booking_status_on_date = null; // null, 'pending', or
'accepted'

            foreach ($active_venues as $venue) {
                // Εφαρμόζουμε όλους τους κανόνες διαθεσιμότητας
                if (empty($venue[$day_of_week])) continue;
                if (($venue['valid_from'] && $date_str <
$venue['valid_from']) || ($venue['valid_until'] && $date_str >
$venue['valid_until'])) continue;
                if (isset($exceptions_by_date[$date_str][null]) ||
isset($exceptions_by_date[$date_str][$venue['idVenue']])) continue;
```

```
                if
(isset($bookings_by_date[$date_str][$venue['idVenue']])) {
                    $booking_info =
$bookings_by_date[$date_str][$venue['idVenue']];
                    $booked_venues_today[] = $venue;

                    if ($booking_info['idUser'] == $current_user_id) {
                        $user_booking_status_on_date =
($booking_info['idStatus'] == 1) ? 'pending' : 'accepted';
                    }
                } else {
                    $available_venues_today[] = $venue;
                }
            }

            $total_slots = count($available_venues_today) +
count($booked_venues_today);
            if ($total_slots == 0) continue;

            // --- ΛΟΓΙΚΗ ΓΙΑ ΤΗ ΔΗΜΙΟΥΡΓΙΑ ΤΟΥ EVENT ---
            $booked_count = count($booked_venues_today);
            $available_for_booking = $total_slots - $booked_count;

            $is_bookable_user = ($available_for_booking > 0);
            $color = '#28a745'; // Πράσινο
            $title = "{$booked_count}/{$total_slots} Κλεισμένες";

            if ($booked_count > 0 && $is_bookable_user) $color =
'#ffc107'; // Πορτοκαλί
            if (!$is_bookable_user) $color = '#dc3545'; // Κόκκινο

            // Ο κανόνας των 6 ημερών εφαρμόζεται μόνο για τους απλούς
χρήστες
            if (!$is_manager && ($dt < $today || $today->diff($dt)-
>days < 6)) {
                $is_bookable_user = false;
            }

            $extendedProps = [
                'is_bookable_user' => $is_bookable_user,
                'is_manager' => $is_manager,
                'user_booking_status' => $user_booking_status_on_date,
                'reason' => null
            ];

            // Αν ο χρήστης έχει pending αίτημα, του βγάζουμε ειδικό
μήνυμα
            if($user_booking_status_on_date === 'pending') {
```

```
                $extendedProps['reason'] = 'Έχετε ήδη ένα αίτημα σε
εκκρεμότητα για αυτή την ημέρα.';
            }


            if ($is_manager) {
                $details = [];
                foreach ($available_venues_today as $v) $details[] =
['id' => $v['idVenue'], 'name' => $v['venueName'], 'status' =>
'available'];
                foreach ($booked_venues_today as $v) {
                    $booking_info =
$bookings_by_date[$date_str][$v['idVenue']];
                    $company_name = !empty($booking_info['idUser'])
                        ? ($this->userModel-
>findById($booking_info['idUser'])['company'] ?? 'N/A')
                        : $booking_info['on_behalf_of_company'];
                    $status_name = ($booking_info['idStatus'] == 1) ?
'Pending' : 'Accepted';
                    $details[] = ['id' => $v['idVenue'], 'name' =>
$v['venueName'], 'status' => 'booked', 'company' => $company_name,
'booking_status' => $status_name];
                }
                $extendedProps['details'] = $details;
            }

            $events[] = ['title' => $title, 'start' => $date_str,
'backgroundColor' => $color, 'borderColor' => $color, 'allDay' => true,
'extendedProps' => $extendedProps];
        }

        echo json_encode($events);
        exit();
    }


    /**
     * Επιστρέφει τις ΔΙΑΘΕΣΙΜΕΣ αίθουσες για μια συγκεκριμένη
ημερομηνία.
     */
    public function getAvailableVenuesForDate()
    {
        header('Content-Type: application/json');
        $date = $_GET['date'] ?? null;
        if (!$date) { echo json_encode(['error' => 'Missing date']);
exit(); }

        // Παίρνουμε όλα τα δεδομένα
        $active_venues = $this->venueModel->getAll(true);
```

```
        $bookings = $this->bookVenueModel-
>getActiveBookingsByDateRange($date, $date);
        $exceptions = $this->venueExceptionModel-
>getExceptionsByDateRange($date, $date);

        $booked_venue_ids = array_column($bookings, 'idVenue');
        $exception_venues_today = isset($exceptions[$date]) ?
array_column($exceptions[$date], 'idVenue') : [];

        $is_general_holiday = in_array(null, $exception_venues_today);
        if ($is_general_holiday) {
            echo json_encode([]); // Αν είναι γενική αργία, καμία
αίθουσα δεν είναι διαθέσιμη
            exit();
        }

        // Βρίσκουμε τις διαθέσιμες
        $available_venues = [];
        $day_of_week = strtolower(date('l', strtotime($date)));

        foreach ($active_venues as $venue) {
            // Ελέγχουμε όλους τους κανόνες...
            if (empty($venue[$day_of_week])) continue;
            if (($venue['valid_from'] && $date < $venue['valid_from'])
|| ($venue['valid_until'] && $date > $venue['valid_until'])) continue;
            if (in_array($venue['idVenue'], $exception_venues_today))
continue;
            if (in_array($venue['idVenue'], $booked_venue_ids))
continue;

            // Αν πέρασε όλα τα τεστ, είναι διαθέσιμη!
            $available_venues[] = [
                'id' => $venue['idVenue'],
                'name' => $venue['venueName'] . ' ( χωρ: ' .
$venue['capacity'] . ', €' . number_format($venue['price'], 2) . ')'
            ];
        }

        echo json_encode($available_venues);
        exit();
    }


    public function getBookingDetails()
    {
        header('Content-Type: application/json');

        $type = $_GET['type'] ?? null;
        $id = $_GET['id'] ?? null;
```

```
        $userId = $_SESSION['user_id'];

        if (!$type || !$id || !is_numeric($id)) {
            echo json_encode(['success' => false, 'message' => 'Μη
έγκυρα δεδομένα.']);
            exit();
        }

        $html = '';
        $booking = null;
        $is_authorized = false;

        if ($type === 'tour') {
            // --- ΟΛΑ ΤΑ ΔΕΔΟΜΕΝΑ ---
            $booking = $this->bookTourModel-
>getBookingDetailsById((int)$id);
            if ($booking) {
                // Έλεγχος δικαιωμάτων
                if ((isset($booking['idUser']) && $booking['idUser'] ==
$userId) || is_super_admin() || is_tour_manager()) {
                    $is_authorized = true;

                    $html = "<h5>Λεπτομέρειες Σχ. Επίσκεψης</h5>" .
                            "<p><strong>Πρόγραμμα:</strong> " .
htmlspecialchars($booking['tourName']) . "</p>" .
                            "<p><strong>Ημερομηνία:</strong> " .
date('d/m/Y', strtotime($booking['date'])) . "</p>" .
                            "<p><strong>Αριθμός Παιδιών:</strong> " .
htmlspecialchars($booking['children']) . "</p>" .
                            "<p><strong>Τάξη:</strong> " .
htmlspecialchars($booking['class']) . "</p>" .
                            "<p><strong>Κατάσταση:</strong> " .
htmlspecialchars($booking['statusName']) . "</p><hr>";

                    // --- ΕΜΦΑΝΙΣΗ ΣΤΟΙΧΕΙΩΝ ---
                    if (!empty($booking['created_by_idUser'])) {
                        // Είναι κράτηση από manager
                        $manager = $this->userModel-
>findById($booking['created_by_idUser']);
                        $managerName = $manager ? $manager['username']
: 'Άγνωστος';
                        $html .= "<h5>Στοιχεία Πελάτη (Καταχώρηση από
{$managerName})</h5>" .
                                 "<p><strong>Σχολείο:</strong> " .
htmlspecialchars($booking['on_behalf_of_school']) . "</p>" .
                                 "<p><strong>Υπεύθυνος:</strong> " .
htmlspecialchars($booking['on_behalf_of_name']) . "</p>" .
```

```
                                "<p><strong>Email:</strong> " .
htmlspecialchars($booking['on_behalf_of_email'] ?? 'N/A') . "</p>" .
                                "<p><strong>Τηλέφωνο:</strong> " .
htmlspecialchars($booking['on_behalf_of_phone']) . "</p>";
                    } else {
                        // Είναι κράτηση από χρήστη
                        $user = $this->userModel-
>findById($booking['idUser']);
                        $html .= "<h5>Στοιχεία Επικοινωνίας</h5>" .
                                "<p><strong>Σχολείο:</strong> " .
htmlspecialchars($user['school'] ?? 'N/A') . "</p>" .
                                "<p><strong>Υπεύθυνος
(Χρήστης):</strong> " . htmlspecialchars($user['username']) . "</p>" .
                                "<p><strong>Email:</strong> " .
htmlspecialchars($user['mail']) . "</p>" .
                                "<p><strong>Τηλέφωνο:</strong> " .
htmlspecialchars($user['phone']) . "</p>";
                    }
                }
            }
        } elseif ($type === 'venue') {
            $booking = $this->bookVenueModel-
>getBookingDetailsById((int)$id);
            if ($booking) {
                if ((isset($booking['idUser']) && $booking['idUser'] ==
$userId) || is_super_admin() || is_venue_manager()) {
                    $is_authorized = true;

                    $html = "<h5>Λεπτομέρειες Κράτησης Αίθουσας</h5>" .
                            "<p><strong>Αίθουσα:</strong> " .
htmlspecialchars($booking['venueName']) . "</p>" .
                            "<p><strong>Ημερομηνία:</strong> " .
date('d/m/Y', strtotime($booking['date'])) . "</p>" .
                            "<p><strong>Κατάσταση:</strong> " .
htmlspecialchars($booking['statusName']) . "</p><hr>";

                    if (!empty($booking['created_by_idUser'])) {
                        // Είναι κράτηση από manager
                        $manager = $this->userModel-
>findById($booking['created_by_idUser']);
                        $managerName = $manager ? $manager['username']
: 'Άγνωστος';
                        $html .= "<h5>Στοιχεία Πελάτη (Καταχώρηση από
{$managerName})</h5>" .
                                "<p><strong>Εταιρεία:</strong> " .
htmlspecialchars($booking['on_behalf_of_company']) . "</p>" .
                                "<p><strong>Υπεύθυνος:</strong> " .
htmlspecialchars($booking['on_behalf_of_name']) . "</p>" .
```

```
                                "<p><strong>Email:</strong> " .
htmlspecialchars($booking['on_behalf_of_email'] ?? 'N/A') . "</p>" .
                                "<p><strong>Τηλέφωνο:</strong> " .
htmlspecialchars($booking['on_behalf_of_phone']) . "</p>";
                    } else {
                        // Είναι κράτηση από χρήστη
                        $user = $this->userModel-
>findById($booking['idUser']);
                        $html .= "<h5>Στοιχεία Πελάτη</h5>" .
                                "<p><strong>Εταιρεία:</strong> " .
htmlspecialchars($user['company'] ?? 'N/A') . "</p>" .
                                "<p><strong>Υπεύθυνος
(Χρήστης):</strong> " . htmlspecialchars($user['username']) . "</p>" .
                                "<p><strong>Email:</strong> " .
htmlspecialchars($user['mail']) . "</p>" .
                                "<p><strong>Τηλέφωνο:</strong> " .
htmlspecialchars($user['phone']) . "</p>";
                }
                }
            }
        }

        if ($is_authorized) {
            echo json_encode(['success' => true, 'html' => $html]);
        } else {
            echo json_encode(['success' => false, 'message' => 'Δεν
έχετε πρόσβαση σε αυτή την κράτηση.']);
        }
        exit();
    }

}
```

BookingController.php

```
<?php

class BookingController
{
    private $bookTourModel;
    private $userModel;
    private $logModel;
    private $bookAlertModel;
    private $tourModel;
    private $venueModel;
    private $bookVenueModel;
```

```
    public function __construct(BookTour $bookTourModel, User
$userModel, ActivityLog $logModel, BookAlert $bookAlertModel, Tour
$tourModel, Venue $venueModel, BookVenue $bookVenueModel)
    {
        $this->bookTourModel = $bookTourModel;
        $this->userModel = $userModel;
        $this->logModel = $logModel;
        $this->bookAlertModel = $bookAlertModel;
        $this->tourModel = $tourModel;
        $this->venueModel = $venueModel;
        $this->bookVenueModel = $bookVenueModel;
    }

    /**
     * Αποθηκεύει μια νέα κράτηση για σχολική επίσκεψη.
     */
    public function storeTourBooking()
    {
        // Έλεγχος δικαιωμάτων
        if (!is_logged_in()) {
            http_response_code(403);
            exit('Access Denied');
        }

        // --- Validation ---
        $errors = [];
        if (empty($_POST['booking_date'])) $errors[] = "Η ημερομηνία
λείπει.";
        if (empty($_POST['idTour'])) $errors[] = "Πρέπει να επιλέξετε
πρόγραμμα.";
        if (empty($_POST['children']) ||
!is_numeric($_POST['children']) || $_POST['children'] < 1) {
            $errors[] = "Ο αριθμός παιδιών πρέπει να είναι έγκυρος
αριθμός μεγαλύτερος του 0.";
        }
        if (empty($_POST['class'])) $errors[] = "Το πεδίο 'Τάξη' είναι
υποχρεωτικό.";

        $user = $this->userModel->findById($_SESSION['user_id']);
        if (empty($user['school'])) {
            $errors[] = 'Πρέπει να έχετε δηλώσει το σχολείο σας στο
προφίλ σας για να κάνετε κράτηση. <a href="' . url('/profile/edit') .
'">Επεξεργασία Προφίλ</a>';
        }

        if (!empty($errors)) {
            $errorMessage = "<ul>";
```

```
            foreach ($errors as $error) { $errorMessage .= "<li>" .
$error . "</li>"; }
            $errorMessage .= "</ul>";
            flash('danger', $errorMessage, true);
            header('Location: ' . url('/tours-calendar'));
            exit();
        }

        // --- ΠΛΗΡΗΣ ΕΛΕΓΧΟΣ ΔΙΑΘΕΣΙΜΟΤΗΤΑΣ ---
        $isAvailable = $this->tourModel-
>isTourAvailableOnDate((int)$_POST['idTour'], $_POST['booking_date']);
        if (!$isAvailable) {
            flash('danger', 'Το πρόγραμμα που επιλέξατε δεν είναι
διαθέσιμο για την συγκεκριμένη ημερομηνία.');
            header('Location: ' . url('/tours-calendar'));
            exit();
        }

        // Έλεγχος σε περίπτωση που κάποιος πρόλαβε να κάνει κράτηση.
        $isAlreadyBooked = $this->bookTourModel-
>doesBookingExist((int)$_POST['idTour'], $_POST['booking_date']);
        if ($isAlreadyBooked) {
            flash('danger', 'Αυτή η θέση μόλις κρατήθηκε από κάποιον
άλλον. Παρακαλώ επιλέξτε μια άλλη ημέρα ή πρόγραμμα.');
            header('Location: ' . url('/tours-calendar'));
            exit();
        }

        // --- Προετοιμασία Δεδομένων ---
        $data = [
            'idTour'   => (int)$_POST['idTour'],
            'idStatus' => 2,
            'idUser'   => (int)$_SESSION['user_id'],
            'date'     => $_POST['booking_date'],
            'children' => (int)$_POST['children'],
            'class'    => trim($_POST['class'])
        ];

        // --- ΔΗΜΙΟΥΡΓΙΑ ΚΡΑΤΗΣΗΣ ---
        // Η μέθοδος create επιστρέφει το ID της νέας κράτησης ή false.
        $newBookingId = $this->bookTourModel->create($data);

        if ($newBookingId) {
            // Η κράτηση ήταν επιτυχής!
            flash('success', "Η κράτησή σας για τις " . date('d/m/Y',
strtotime($data['date'])) . " καταχωρήθηκε με επιτυχία!");

            // 1. LOGGING
```

```
            $this->logModel->logAction(
                'TOUR_BOOKING_CREATE',
                "Ο χρήστης '{$user['username']}' έκανε κράτηση για
σχολική επίσκεψη στις {$data['date']}.",
                $user['idUser']
            );

            // 2. ΔΗΜΙΟΥΡΓΙΑ ΕΙΔΟΠΟΙΗΣΗΣ
            $message = "Νέα σχολική επίσκεψη από το '{$user['school']}'
για τις " . date('d/m/Y', strtotime($data['date'])) . ".";
            $this->bookAlertModel->createForManager($message,
$newBookingId, null); // Χρησιμοποιούμε τη σωστή μέθοδο

        } else {
            // Η κράτηση απέτυχε.
            flash('danger', 'Παρουσιάστηκε ένα σφάλμα. Η κράτηση δεν
καταχωρήθηκε.');
        }

        header('Location: ' . url('/tours-calendar'));
        exit();
    }

    /**
     * Εμφανίζει τη λίστα όλων των κρατήσεων για τον manager.
     */
    public function listTourBookings()
    {
        if (!is_super_admin() && !is_tour_manager()) {
            http_response_code(403);
            require_once BASE_PATH . '/views/errors/403.php';
            exit();
        }

        $bookings = $this->bookTourModel->getAllBookingsDetails();

        require_once BASE_PATH . '/views/tours/all-bookings.php';
    }

    /**
     * Ακυρώνει μια κράτηση (ενέργεια από manager).
     */
    public function cancelTourBookingByManager(int $bookingId)
    {
        if (!is_super_admin() && !is_tour_manager()) {
            http_response_code(403);
            require_once BASE_PATH . '/views/errors/403.php';
            exit();
```

```
        }

        $booking = $this->bookTourModel->findBookingById($bookingId);
        if (!$booking || $booking['idStatus'] == 4) {
            flash('danger', 'Η κράτηση δεν βρέθηκε ή έχει ήδη
ακυρωθεί.');
            header('Location: ' . url('/tours/bookings'));
            exit();
        }

        $success = $this->bookTourModel->updateStatus($bookingId, 4);

        if ($success) {
            flash('success', 'Η κράτηση ακυρώθηκε με επιτυχία.');

            $adminUsername = $_SESSION['username'];

            // Αν η κράτηση είχε γίνει από εγγεγραμμένο χρήστη
            if (!empty($booking['idUser'])) {
                $user = $this->userModel->findById($booking['idUser']);
                $tour = $this->tourModel->findById($booking['idTour']);
                $tourName = $tour ? $tour['tourName'] : '';

                // LOGGING
                $this->logModel->logAction(
                    'TOUR_BOOKING_CANCEL_MANAGER',
                    "Ο χρήστης '{$adminUsername}' ακύρωσε την κράτηση
#{$bookingId} για το σχολείο '{$user['school']}'.",
                    $_SESSION['user_id']
                );

                // ΕΙΔΟΠΟΙΗΣΗ ΠΡΟΣ ΧΡΗΣΤΗ
                $userMessage = "Η κράτησή σας για '{$tourName}' στις "
. date('d/m/Y', strtotime($booking['date'])) . " ακυρώθηκε από τον
διαχειριστή.";
                $this->bookAlertModel->createForUser($userMessage,
$user['idUser'], $bookingId, null);
            } else {
                // Αν η κράτηση είχε γίνει από manager (on-behalf-of)
                // Δεν στέλνουμε ειδοποίηση (δεν ξέρουμε πού), απλά
καταγράφουμε το γεγονός.
                $this->logModel->logAction(
                    'TOUR_BOOKING_CANCEL_MANAGER',
                    "Ο χρήστης '{$adminUsername}' ακύρωσε την on-
behalf-of κράτηση #{$bookingId}.",
                    $_SESSION['user_id']
                );
            }
```

```
        } else {
            flash('danger', 'Παρουσιάστηκε σφάλμα κατά την ακύρωση.');
        }

        header('Location: ' . url('/tours/bookings'));
        exit();
    }


    /**
    * Αποθηκεύει ένα νέο αίτημα κράτησης για αίθουσα.
    */
    public function storeVenueBooking()
    {
        if (!is_logged_in()) {
                http_response_code(403);
                exit('Access Denied');
        }
        // --- Validation ---
        $errors = [];
        if (empty($_POST['booking_date'])) $errors[] = "Η ημερομηνία
λείπει.";
        if (empty($_POST['idVenue'])) $errors[] = "Πρέπει να επιλέξετε
αίθουσα.";

        $user = $this->userModel->findById($_SESSION['user_id']);
        if (empty($user['company'])) {
            $errors[] = 'Πρέπει να έχετε δηλώσει την εταιρεία/οργανισμό
σας στο προφίλ σας για να κάνετε κράτηση. <a href="' .
url('/profile/edit') . '">Επεξεργασία Προφίλ</a>';
        }

        if (!empty($errors)) {
            $errorMessage = "<ul>";
            foreach ($errors as $error) { $errorMessage .= "<li>" .
$error . "</li>"; }
            $errorMessage .= "</ul>";
            flash('danger', $errorMessage, true);
            header('Location: ' . url('/tours-calendar'));
            exit();
        }

        // --- ΠΛΗΡΗΣ ΕΛΕΓΧΟΣ ΔΙΑΘΕΣΙΜΟΤΗΤΑΣ ---
        $isAvailable = $this->venueModel-
>isVenueAvailableOnDate((int)$_POST['idVenue'],
$_POST['booking_date']);
        if (!$isAvailable) {
```

```
            flash('danger', 'Η αίθουσα που επιλέξατε δεν είναι
διαθέσιμη για τη συγκεκριμένη ημερομηνία.');
            header('Location: ' . url('/venues-calendar'));
            exit();
        }

        // --- Έλεγχος ασφαλείας της τελευταίας στιγμής ---
        $isAlreadyBooked = $this->bookVenueModel-
>doesBookingExist((int)$_POST['idVenue'], $_POST['booking_date']);
        if ($isAlreadyBooked) {
            flash('danger', 'Αυτή η αίθουσα μόλις κρατήθηκε από κάποιον
άλλον για αυτή την ημερομηνία. Παρακαλώ επιλέξτε μια άλλη.');
            header('Location: ' . url('/venues-calendar'));
            exit();
        }

        $data = [
            'idVenue'  => (int)$_POST['idVenue'],
            'idStatus' => 1, // 1 = pending
            'idUser'   => (int)$_SESSION['user_id'],
            'date'     => $_POST['booking_date']
        ];

        $newBookingId = $this->bookVenueModel->create($data);

        if ($newBookingId) {
            $venue = $this->venueModel->findById($data['idVenue']);
            flash('success', "Το αίτημά σας για την αίθουσα
'{$venue['venueName']}' στις " . date('d/m/Y',
strtotime($data['date'])) . " υποβλήθηκε με επιτυχία και αναμένει
έγκριση.");

            // LOGGING
            $this->logModel->logAction('VENUE_BOOKING_REQUEST', "Ο
χρήστης '{$user['username']}' υπέβαλε αίτημα για την αίθουσα
'{$venue['venueName']}' στις {$data['date']}.", $user['idUser']);

            // ΕΙΔΟΠΟΙΗΣΗ ΠΡΟΣ MANAGER
            $message = "Νέο αίτημα κράτησης για την αίθουσα
'{$venue['venueName']}' από '{$user['company']}'.";
            $this->bookAlertModel->createForManager($message, null,
$newBookingId);

        } else {
            flash('danger', 'Παρουσιάστηκε σφάλμα. Το αίτημά σας δεν
καταχωρήθηκε.');
        }
```

```
        header('Location: ' . url('/venues-calendar'));
        exit();
    }


    /**
     * Εμφανίζει τη λίστα με τα pending αιτήματα κρατήσεων αιθουσών.
     */
    public function listPendingVenueBookings()
    {
        if (!is_super_admin() && !is_venue_manager()) {
            http_response_code(403);
            require_once BASE_PATH . '/views/errors/403.php';
            exit();
        }

        // Φέρνουμε μόνο τα αιτήματα με status 1 (pending)
        $pendingBookings = $this->bookVenueModel-
>getAllBookingsDetails(1);

        require_once BASE_PATH . '/views/venues/pending-bookings.php';
    }


    public function approveVenueBooking(int $bookingId)
    {
        $this->handleVenueBookingStatus($bookingId, 2); // 2 = accepted
    }

    public function rejectVenueBooking(int $bookingId)
    {
        $this->handleVenueBookingStatus($bookingId, 3); // 3 = rejected
    }

    // Βοηθητική μέθοδος για να μην επαναλαμβάνουμε κώδικα
    private function handleVenueBookingStatus(int $bookingId, int
$newStatus)
    {
        if (!is_super_admin() && !is_venue_manager()) {
            http_response_code(403);
            require_once BASE_PATH . '/views/errors/403.php';
            exit();
        }

        $booking = $this->bookVenueModel->findBookingById($bookingId);
        if (!$booking || $booking['idStatus'] != 1) {
            flash('danger', 'Το αίτημα δεν βρέθηκε ή έχει ήδη
απαντηθεί.');
            header('Location: ' . url('/venues/requests'));
```

```
            exit();
        }

        $success = $this->bookVenueModel->updateStatus($bookingId,
$newStatus);

        $statusActionText = ($newStatus == 2) ? 'εγκρίθηκε' :
'απορρίφθηκε';
        $statusUserText = ($newStatus == 2) ? 'εγκρίθηκε' :
'απορρίφθηκε';

        if ($success) {
            $user = $this->userModel->findById($booking['idUser']);
            $venue = $this->venueModel->findById($booking['idVenue']);
            $venueName = $venue ? $venue['venueName'] : 'Άγνωστη
Αίθουσα';

            flash('success', "Το αίτημα για την αίθουσα '{$venueName}'
{$statusActionText} με επιτυχία.");

            // LOGGING
            $adminUsername = $_SESSION['username'];
            $this->logModel->logAction(
                'VENUE_BOOKING_STATUS_CHANGE',
                "Ο χρήστης '{$adminUsername}' {$statusActionText} το
αίτημα #{$bookingId} για την αίθουσα '{$venueName}'.",
                $_SESSION['user_id']
            );

            // --- ΔΗΜΙΟΥΡΓΙΑ ΕΙΔΟΠΟΙΗΣΗΣ ΠΡΟΣ ΤΟΝ ΧΡΗΣΤΗ ---
            $userMessage = "Το αίτημά σας για την αίθουσα
'{$venueName}' στις " . date('d/m/Y', strtotime($booking['date'])) . "
μόλις {$statusUserText}.";

            // Καλούμε τη μέθοδο createForUser, περνώντας το ID του
χρήστη και το ID της κράτησης της αίθουσας.
            $this->bookAlertModel->createForUser($userMessage,
$user['idUser'], null, $bookingId);
            // --- ΤΕΛΟΣ ΕΙΔΟΠΟΙΗΣΗΣ ---

        } else {
            flash('danger', 'Παρουσιάστηκε σφάλμα.');
        }

        header('Location: ' . url('/venues/requests'));
        exit();
    }
```

```php
    /**
     * Εμφανίζει τη λίστα με τις κρατήσεις του συνδεδεμένου χρήστη.
     */
    public function listUserBookings()
    {
        if (!is_simple_user()) {
            http_response_code(403);
            require_once BASE_PATH . '/views/errors/403.php';
            exit();
        }

        $userId = $_SESSION['user_id'];

        // Φέρνουμε τις κρατήσεις και για τα δύο είδη
        $tourBookings = $this->bookTourModel-
>getBookingsByUserId($userId);
        $venueBookings = $this->bookVenueModel-
>getBookingsByUserId($userId);

        require_once BASE_PATH . '/views/bookings/my-bookings.php';
    }

    /**
     * Ακυρώνει μια κράτηση αίθουσας (ενέργεια από τον χρήστη).
     */
    public function cancelVenueBookingByUser(int $bookingId)
    {
        if (!is_simple_user()) {
            http_response_code(403);
            require_once BASE_PATH . '/views/errors/403.php';
            exit();
         }

        $userId = $_SESSION['user_id'];
        $booking = $this->bookVenueModel->findBookingById($bookingId);

        if (!$booking || $booking['idUser'] != $userId) {
            flash('danger', 'Μη εξουσιοδοτημένη ενέργεια.');
            header('Location: ' . url('/my-bookings'));
            exit();
        }

        // Επιτρέπουμε ακύρωση μόνο για pending ή accepted
        if (!in_array($booking['idStatus'], [1, 2])) {
            flash('danger', 'Αυτή η κράτηση δεν μπορεί να ακυρωθεί.');
            header('Location: ' . url('/my-bookings'));
            exit();
        }
```

```
        $success = $this->bookVenueModel->updateStatus($bookingId, 4);
// 4 = cancelled

        if ($success) {
            flash('success', 'Η κράτησή σας για την αίθουσα ακυρώθηκε
με επιτυχία.');

            $user = $this->userModel->findById($userId);
            $venue = $this->venueModel->findById($booking['idVenue']);
            $venueName = $venue ? $venue['venueName'] : '';

            $message = "Ο χρήστης '{$user['username']}' (Εταιρεία:
{$user['company']}) ακύρωσε την κράτηση #{$bookingId} για την αίθουσα
'{$venueName}'.";
            $this->logModel->logAction('VENUE_BOOKING_CANCEL_USER',
$message, $userId);
            $this->bookAlertModel->createForManager($message, null,
$bookingId);

        } else {
            flash('danger', 'Παρουσιάστηκε σφάλμα.');
        }

        header('Location: ' . url('/my-bookings'));
        exit();
    }


    /**
     * Ακυρώνει μια κράτηση σχολικής επίσκεψης (ενέργεια από τον
χρήστη).
     */
    public function cancelTourBookingByUser(int $bookingId)
    {
        // 1. Έλεγχος δικαιωμάτων: Πρέπει να είναι απλός χρήστης
        if (!is_simple_user()) {
            http_response_code(403);
            require_once BASE_PATH . '/views/errors/403.php';
            exit();
        }

        $userId = $_SESSION['user_id'];
        $booking = $this->bookTourModel->findBookingById($bookingId);

        // 2. Έλεγχος ασφαλείας: Ο χρήστης μπορεί να ακυρώσει μόνο
δικές του κρατήσεις
        if (!$booking || $booking['idUser'] != $userId) {
```

```
            flash('danger', 'Μη εξουσιοδοτημένη ενέργεια. Δεν έχετε
πρόσβαση σε αυτή την κράτηση.');
            header(('Location: ' . url('/my-bookings')));
            exit();
        }

        // 3. Επιχειρησιακός κανόνας: Επιτρέπουμε ακύρωση μόνο για
'accepted' κρατήσεις
        if ($booking['idStatus'] != 2) { // 2 = accepted
            flash('danger', 'Αυτή η κράτηση δεν μπορεί να ακυρωθεί
καθώς δεν είναι σε κατάσταση "Accepted".');
            header(('Location: ' . url('/my-bookings')));
            exit();
        }

        $bookingDate = new DateTime($booking['date']);
        $today = new DateTime();
        $today->setTime(0, 0, 0);
        $diff = $today->diff($bookingDate)->days;

        if ($diff < 3) {
            flash('danger', 'Η κράτηση δεν μπορεί να ακυρωθεί καθώς
απομένουν λιγότερες από 3 ημέρες.');
            header('Location: ' . url('/my-bookings'));
            exit();
        }

        // 5. Αν όλοι οι έλεγχοι περάσουν, προχωράμε στην ακύρωση
        $success = $this->bookTourModel->updateStatus($bookingId, 4);
// 4 = cancelled

        if ($success) {
            flash('success', 'Η κράτησή σας για τη σχολική επίσκεψη
ακυρώθηκε με επιτυχία.');

            // Παίρνουμε τα δεδομένα για το log και την ειδοποίηση
            $user = $this->userModel->findById($userId);
            $tour = $this->tourModel->findById($booking['idTour']);
            $tourName = $tour ? $tour['tourName'] : '';

            // LOGGING
            $logMessage = "Ο χρήστης '{$user['username']}' (Σχολείο:
{$user['school']}) ακύρωσε την κράτηση #{$bookingId} για το πρόγραμμα
'{$tourName}'.";
            $this->logModel->logAction('TOUR_BOOKING_CANCEL_USER',
$logMessage, $userId);

            // ΕΙΔΟΠΟΙΗΣΗ ΠΡΟΣ MANAGER
```

```
            $alertMessage = "Ακυρώθηκε η κράτηση #{$bookingId} για το
σχολείο '{$user['school']}' στις " . date('d/m/Y',
strtotime($booking['date'])) . ".";
            $this->bookAlertModel->createForManager($alertMessage,
$bookingId, null);

        } else {
            flash('danger', 'Παρουσιάστηκε ένα σφάλμα κατά την
προσπάθεια ακύρωσης.');
        }

        header('Location: ' . url('/my-bookings'));
        exit();
    }

    /**
     * Εμφανίζει τη λίστα με ΟΛΕΣ τις κρατήσεις αιθουσών για τον
manager.
     */
    public function listAllVenueBookings()
    {
        if (!is_super_admin() && !is_venue_manager()) {
            http_response_code(403);
            require_once BASE_PATH . '/views/errors/403.php';
            exit();
        }

        // Καλούμε την getAllBookingsDetails() χωρίς παράμετρο για να
τα φέρει όλα
        $allBookings = $this->bookVenueModel->getAllBookingsDetails();

        require_once BASE_PATH . '/views/venues/all-bookings.php';
    }

    /**
     * Εμφανίζει τη φόρμα για να δημιουργήσει ο manager μια κράτηση
Tour.
     */
    public function showCreateTourBookingFormForManager()
    {
        if (!is_super_admin() && !is_tour_manager()) {
            http_response_code(403);
            require_once BASE_PATH . '/views/errors/403.php';
            exit();
        }

        // Παίρνουμε όλα τα ενεργά προγράμματα για να τα βάλουμε στο
dropdown
```

```
        $tours = $this->tourModel->getAll(true);

        require_once BASE_PATH . '/views/tours/create-booking-
manager.php';
    }

    /**
     * Αποθηκεύει μια νέα κράτηση Tour που δημιουργήθηκε από manager.
     */
    public function storeTourBookingByManager()
    {
        if (!is_super_admin() && !is_tour_manager()) {
            http_response_code(403);
            exit('Access Denied');
        }

        // --- Validation ---
        $errors = [];
        if (empty($_POST['on_behalf_of_name'])) $errors[] = "Το
Ονοματεπώνυμο Υπεύθυνου είναι υποχρεωτικό.";
        if (empty($_POST['on_behalf_of_phone'])) $errors[] = "Το
Τηλέφωνο είναι υποχρεωτικό.";
        if (!empty($_POST['on_behalf_of_phone']) &&
!ctype_digit(str_replace(' ', '', $_POST['on_behalf_of_phone'])))  {
            $errors[] = "Το τηλέφωνο πρέπει να περιέχει μόνο
αριθμούς.";
        }
        if (empty($_POST['on_behalf_of_school'])) $errors[] = "Το
Σχολείο είναι υποχρεωτικό.";
        if (empty($_POST['date'])) $errors[] = "Η Ημερομηνία είναι
υποχρεωτική.";
        if (empty($_POST['idTour'])) $errors[] = "Το Πρόγραμμα είναι
υποχρεωτικό.";
        if (empty($_POST['children']) ||
!is_numeric($_POST['children'])) $errors[] = "Ο Αριθμός Παιδιών είναι
υποχρεωτικός.";
        if (empty($_POST['class'])) $errors[] = "Η Τάξη είναι
υποχρεωτική.";

        if (!empty($errors)) {
            $errorMessage = "<ul>";
            foreach ($errors as $error) { $errorMessage .= "<li>" .
$error . "</li>"; }
            $errorMessage .= "</ul>";
            flash('danger', $errorMessage, true); // true για να
επιτρέψει το HTML

            $_SESSION['old_post'] = $_POST;
```

```
            header('Location: ' . url('/tours/create-booking'));
            exit();
        }

        // --- ΠΛΗΡΗΣ ΕΛΕΓΧΟΣ ΔΙΑΘΕΣΙΜΟΤΗΤΑΣ ---
        $isAvailable = $this->tourModel-
>isTourAvailableOnDate((int)$_POST['idTour'], $_POST['date']);
        if (!$isAvailable) {
            flash('danger', 'Το πρόγραμμα που επιλέξατε δεν είναι
διαθέσιμο για την συγκεκριμένη ημερομηνία.');
            $_SESSION['old_post'] = $_POST;
            header('Location: ' . url('/tours/create-booking'));
            exit();
        }

        // Έλεγχος διαθεσιμότητας της τελευταίας στιγμής
        $isAlreadyBooked = $this->bookTourModel-
>doesBookingExist((int)$_POST['idTour'], $_POST['date']);
        if ($isAlreadyBooked) {
            flash('danger', "Αυτή η θέση (Πρόγραμμα/Ημερομηνία) είναι
ήδη κρατημένη.");
            header('Location: ' . url('/tours/create-booking'));
            exit();
        }

        // --- Προετοιμασία Δεδομένων ---
        $data = [
            'idTour'    => (int)$_POST['idTour'],
            'idStatus' => 2, // Accepted
            'date'      => $_POST['date'],
            'children' => (int)$_POST['children'],
            'class'     => trim($_POST['class']),
            'idUser'    => null, // Δεν γίνεται από εγγεγραμμένο χρήστη
            'created_by_idUser' => $_SESSION['user_id'], // Ποιος
manager την έκανε
            'on_behalf_of_name' => trim($_POST['on_behalf_of_name']),
            'on_behalf_of_email' => empty($_POST['on_behalf_of_email'])
? null : trim($_POST['on_behalf_of_email']),
            'on_behalf_of_phone' => trim($_POST['on_behalf_of_phone']),
            'on_behalf_of_school' =>
trim($_POST['on_behalf_of_school']),
        ];

        $newBookingId = $this->bookTourModel->create($data);

        if ($newBookingId) {
```

```
            flash('success', "Η κράτηση για το σχολείο
'{$data['on_behalf_of_school']}' καταχωρήθηκε με επιτυχία.");

            $adminUsername = $_SESSION['username'];
            $this->logModel->logAction(
                'TOUR_BOOKING_CREATE_MANAGER',
                "Ο χρήστης '{$adminUsername}' δημιούργησε κράτηση για
το σχολείο '{$data['on_behalf_of_school']}'.",
                $_SESSION['user_id']
            );

            header('Location: ' . url('/tours/bookings')); // Πήγαινε
στη λίστα κρατήσεων
        } else {
            $_SESSION['old_post'] = $_POST;
            flash('danger', 'Παρουσιάστηκε σφάλμα. Η κράτηση δεν
καταχωρήθηκε.');
            header('Location: ' . url('/tours/create-booking'));
        }
        exit();
    }


    /**
     * Εμφανίζει τη φόρμα για να δημιουργήσει ο manager μια κράτηση
Venue.
     */
    public function showCreateVenueBookingFormForManager()
    {
        if (!is_super_admin() && !is_venue_manager()) {
            http_response_code(403);
            require_once BASE_PATH . '/views/errors/403.php';
            exit();
        }

        // Παίρνουμε όλες τις ενεργές αίθουσες για το dropdown
        $venues = $this->venueModel->getAll(true);

        require_once BASE_PATH . '/views/venues/create-booking-
manager.php';
    }


    /**
     * Αποθηκεύει μια νέα κράτηση Venue που δημιουργήθηκε από manager.
     */
    public function storeVenueBookingByManager()
    {
        if (!is_super_admin() && !is_venue_manager()) {
```

```
            http_response_code(403);
            exit('Access Denied');
        }

        // --- Validation ---
        $errors = [];
        if (empty($_POST['on_behalf_of_name'])) $errors[] = "Το
Ονοματεπώνυμο Υπεύθυνου είναι υποχρεωτικό.";
        if (empty($_POST['on_behalf_of_phone'])) $errors[] = "Το
Τηλέφωνο είναι υποχρεωτικό.";
        if (!empty($_POST['on_behalf_of_phone']) &&
!ctype_digit(str_replace(' ', '', $_POST['on_behalf_of_phone'])))  {
            $errors[] = "Το τηλέφωνο πρέπει να περιέχει μόνο
αριθμούς.";
        }
        if (empty($_POST['on_behalf_of_company'])) $errors[] = "Η
Εταιρεία είναι υποχρεωτική.";
        if (empty($_POST['date'])) $errors[] = "Η Ημερομηνία είναι
υποχρεωτική.";
        if (empty($_POST['idVenue'])) $errors[] = "Η Αίθουσα είναι
υποχρεωτική.";

        if (!empty($errors)) {
            $errorMessage = "<ul>";
            foreach ($errors as $error) { $errorMessage .= "<li>" .
$error . "</li>"; }
            $errorMessage .= "</ul>";
            flash('danger', $errorMessage, true);
            $_SESSION['old_post'] = $_POST;
            header('Location: ' . url('/venues/create-booking'));
            exit();
        }

        // --- ΠΛΗΡΗΣ ΕΛΕΓΧΟΣ ΔΙΑΘΕΣΙΜΟΤΗΤΑΣ ---
        $isAvailable = $this->venueModel-
>isVenueAvailableOnDate((int)$_POST['idVenue'], $_POST['date']);
        if (!$isAvailable) {
            flash('danger', 'Η αίθουσα που επιλέξατε δεν είναι
διαθέσιμη για τη συγκεκριμένη ημερομηνία (π.χ. λόγω αργίας, άλλης
κράτησης, ή κανόνων διαθεσιμότητας).');
            $_SESSION['old_post'] = $_POST;
            header('Location: ' . url('/venues/create-booking'));
            exit();
        }

        // Έλεγχος διαθεσιμότητας της τελευταίας στιγμής
        $isAlreadyBooked = $this->bookVenueModel-
>doesBookingExist((int)$_POST['idVenue'], $_POST['date']);
```

```
        if ($isAlreadyBooked) {
            flash('danger', "Αυτή η αίθουσα είναι ήδη κρατημένη για τη
συγκεκριμένη ημερομηνία.");
            $_SESSION['old_post'] = $_POST;
            header('Location: ' . url('/venues/create-booking'));
            exit();
        }

        // --- Προετοιμασία Δεδομένων ---
        $data = [
            'idVenue'  => (int)$_POST['idVenue'],
            'idStatus' => 2, // 2 = Accepted
            'date'     => $_POST['date'],
            'idUser'   => null,
            'created_by_idUser' => $_SESSION['user_id'],
            'on_behalf_of_name' => trim($_POST['on_behalf_of_name']),
            'on_behalf_of_email' => empty($_POST['on_behalf_of_email'])
? null : trim($_POST['on_behalf_of_email']),
            'on_behalf_of_phone' => trim($_POST['on_behalf_of_phone']),
            'on_behalf_of_company' =>
trim($_POST['on_behalf_of_company']),
        ];

        $newBookingId = $this->bookVenueModel->create($data);

        if ($newBookingId) {
            flash('success', "Η κράτηση για την εταιρεία
'{$data['on_behalf_of_company']}' καταχωρήθηκε με επιτυχία.");

            $adminUsername = $_SESSION['username'];
            $this->logModel->logAction(
                'VENUE_BOOKING_CREATE_MANAGER',
                "Ο χρήστης '{$adminUsername}' δημιούργησε κράτηση για
την εταιρεία '{$data['on_behalf_of_company']}'.",
                $_SESSION['user_id']
            );

            header('Location: ' . url('/venues/bookings'));
        } else {
            flash('danger', 'Παρουσιάστηκε σφάλμα. Η κράτηση δεν
καταχωρήθηκε.');
            $_SESSION['old_post'] = $_POST;
            header('Location: ' . url('/venues/create-booking'));
        }
        exit();
    }


    /**
```

```
     * Ακυρώνει μια κράτηση αίθουσας (ενέργεια από manager).
     */
    public function cancelVenueBookingByManager(int $bookingId)
    {
        if (!is_super_admin() && !is_venue_manager()) {
            http_response_code(403);
            require_once BASE_PATH . '/views/errors/403.php';
            exit();
        }

        $booking = $this->bookVenueModel->findBookingById($bookingId);
        if (!$booking || !in_array($booking['idStatus'], [1, 2])) { //
1=pending, 2=accepted
            flash('danger', 'Η κράτηση δεν βρέθηκε ή δεν μπορεί να
ακυρωθεί (είναι ήδη απορριφθείσα/ακυρωμένη).');
            // Κάνουμε redirect στην κατάλληλη σελίδα, ανάλογα από πού
ήρθε ο manager
            $redirectUrl = strpos($_SERVER['HTTP_REFERER'], 'requests')
? '/venues/requests' : '/venues/bookings';
            header('Location: ' . url($redirectUrl));
            exit();
        }

        $success = $this->bookVenueModel->updateStatus($bookingId, 4);
// 4 = cancelled

        if ($success) {
            flash('success', 'Η κράτηση ακυρώθηκε με επιτυχία.');

            $adminUsername = $_SESSION['username'];

            // Αν η κράτηση είχε γίνει από εγγεγραμμένο χρήστη,
στέλνουμε ειδοποίηση
            if (!empty($booking['idUser'])) {
                $user = $this->userModel->findById($booking['idUser']);
                $venue = $this->venueModel-
>findById($booking['idVenue']);
                $venueName = $venue ? $venue['venueName'] : '';

                // LOGGING
                $this->logModel->logAction(
                    'VENUE_BOOKING_CANCEL_MANAGER',
                    "Ο χρήστης '{$adminUsername}' ακύρωσε την κράτηση
#{$bookingId} για την εταιρεία '{$user['company']}'.",
                    $_SESSION['user_id']
                );

                // ΕΙΔΟΠΟΙΗΣΗ ΠΡΟΣ ΧΡΗΣΤΗ
```

```
                $userMessage = "Η κράτησή σας για την αίθουσα
'{$venueName}' στις " . date('d/m/Y', strtotime($booking['date'])) . "
ακυρώθηκε από τον διαχειριστή.";
                $this->bookAlertModel->createForUser($userMessage,
$user['idUser'], null, $bookingId);
            } else {
                // Αν η κράτηση είχε γίνει από manager (on-behalf-of)
                $this->logModel->logAction(
                    'VENUE_BOOKING_CANCEL_MANAGER',
                    "Ο χρήστης '{$adminUsername}' ακύρωσε την on-
behalf-of κράτηση #{$bookingId}.",
                    $_SESSION['user_id']
                );
            }
        } else {
            flash('danger', 'Παρουσιάστηκε σφάλμα κατά την ακύρωση.');
        }

        $redirectUrl = strpos($_SERVER['HTTP_REFERER'], 'requests') ?
'/venues/requests' : '/venues/bookings';
        header('Location: ' . url($redirectUrl));
        exit();
    }


}
```

Αρχεία Μοντέλων

BookVenue.php

```
<?php
class BookVenue
{
    private $pdo;

    public function __construct(PDO $pdo) { $this->pdo = $pdo; }

    /**
     * Επιστρέφει όλες τις ενεργές κρατήσεις (pending & accepted)
     * μέσα σε ένα συγκεκριμένο εύρος ημερομηνιών.
     */
    public function getActiveBookingsByDateRange(string $start_date,
string $end_date): array
    {
        $sql = "SELECT bv.*
                FROM bookVenues bv
                WHERE bv.idStatus IN (1, 2)
                AND bv.date BETWEEN ? AND ?";
```

```
        $stmt = $this->pdo->prepare($sql);
        $stmt->execute([$start_date, $end_date]);
        return $stmt->fetchAll();
    }



    /**
     * Ελέγχει αν υπάρχει ήδη ενεργή κράτηση (pending ή accepted) για
μια αίθουσα σε μια ημερομηνία.
     */
    public function doesBookingExist(int $idVenue, string $date): bool
    {
        $sql = "SELECT 1 FROM bookVenues WHERE idVenue = ? AND date = ?
AND idStatus IN (1, 2) LIMIT 1";
        $stmt = $this->pdo->prepare($sql);
        $stmt->execute([$idVenue, $date]);
        return $stmt->fetch() !== false;
    }

    /**
     * Δημιουργεί ένα νέο αίτημα κράτησης και επιστρέφει το ID του.
     * Χειρίζεται και κρατήσεις από χρήστες και από managers.
     */
    public function create(array $data): int|false
    {
        $sql = "INSERT INTO bookVenues (
                    idVenue, idStatus, date,
                    idUser, created_by_idUser,
                    on_behalf_of_name, on_behalf_of_email,
on_behalf_of_phone, on_behalf_of_company
                ) VALUES (
                    :idVenue, :idStatus, :date,
                    :idUser, :created_by_idUser,
                    :on_behalf_of_name, :on_behalf_of_email,
:on_behalf_of_phone, :on_behalf_of_company
                )";

        $stmt = $this->pdo->prepare($sql);

        $params = [
            ':idVenue'  => $data['idVenue'],
            ':idStatus' => $data['idStatus'],
            ':date'     => $data['date'],
            ':idUser'   => $data['idUser'] ?? null,
            ':created_by_idUser' => $data['created_by_idUser'] ?? null,
            ':on_behalf_of_name' => $data['on_behalf_of_name'] ?? null,
```

```
            ':on_behalf_of_email' => $data['on_behalf_of_email'] ??
null,
            ':on_behalf_of_phone' => $data['on_behalf_of_phone'] ??
null,
            ':on_behalf_of_company' => $data['on_behalf_of_company'] ??
null,
        ];

        try {
            if ($stmt->execute($params)) {
                return $this->pdo->lastInsertId();
            }
            return false;
        } catch (PDOException $e) {
            error_log($e->getMessage());
            return false;
        }
    }

    /**
     * Επιστρέφει όλες τις κρατήσεις με λεπτομερή στοιχεία.
     * Μπορεί να φιλτράρει με βάση το status.
     */
    public function getAllBookingsDetails(?int $statusId = null): array
    {
        $sql = "SELECT
                    bv.idBookVenue, bv.date,
                    bv.on_behalf_of_name, bv.on_behalf_of_company,
                    v.venueName,
                    s.statusName,
                    u.username, u.company
                FROM bookVenues bv
                JOIN venues v ON bv.idVenue = v.idVenue
                JOIN statuses s ON bv.idStatus = s.idStatus
                LEFT JOIN users u ON bv.idUser = u.idUser";

        if ($statusId !== null) {
            $sql .= " WHERE bv.idStatus = :statusId";
        }

        $sql .= " ORDER BY bv.date DESC";

        $stmt = $this->pdo->prepare($sql);
        if ($statusId !== null) {
            $stmt->bindParam(':statusId', $statusId, PDO::PARAM_INT);
        }
        $stmt->execute();
        return $stmt->fetchAll();
```

```
    }

    public function findBookingById(int $id)
    {
        $stmt = $this->pdo->prepare("SELECT * FROM bookVenues WHERE
idBookVenue = ?");
        $stmt->execute([$id]);
        return $stmt->fetch();
    }

    public function updateStatus(int $id, int $idStatus): bool
    {
        $sql = "UPDATE bookVenues SET idStatus = ? WHERE idBookVenue =
?";
        $stmt = $this->pdo->prepare($sql);
        try { return $stmt->execute([$idStatus, $id]); } catch
(PDOException $e) { error_log($e->getMessage()); return false; }
    }

    public function getBookingsByUserId(int $userId): array
    {
        $sql = "SELECT bv.*, v.venueName, s.statusName
                FROM bookVenues bv
                JOIN venues v ON bv.idVenue = v.idVenue
                JOIN statuses s ON bv.idStatus = s.idStatus
                WHERE bv.idUser = ?
                ORDER BY bv.date DESC";
        $stmt = $this->pdo->prepare($sql);
        $stmt->execute([$userId]);
        return $stmt->fetchAll();
    }

    public function getBookingDetailsById(int $id)
    {
        $sql = "SELECT bv.*, v.venueName, s.statusName
                FROM bookVenues bv
                JOIN venues v ON bv.idVenue = v.idVenue
                JOIN statuses s ON bv.idStatus = s.idStatus
                WHERE bv.idBookVenue = ?";
        $stmt = $this->pdo->prepare($sql);
        $stmt->execute([$id]);
        return $stmt->fetch();
    }


    public function getStats(): array
    {
        $stats = [];
        // Αιτήματα σε εκκρεμότητα
```

```
        $stats['pending_requests'] = $this->pdo->query("SELECT COUNT(*)
FROM bookVenues WHERE idStatus = 1")->fetchColumn();
        // Top αίθουσες
        $stats['top_venues'] = $this->pdo->query("SELECT v.venueName,
COUNT(bv.idBookVenue) as count
                                                  FROM bookVenues bv
                                                  JOIN venues v ON
bv.idVenue = v.idVenue
                                                  WHERE bv.idStatus = 2
                                                  GROUP BY v.venueName
                                                  ORDER BY count DESC
                                                  LIMIT 5")-
>fetchAll();
        // Κρατήσεις ανά μήνα
        $stats['bookings_by_month'] = $this->pdo->query("SELECT
DATE_FORMAT(date, '%Y-%m') as month, COUNT(*) as count
                                                        FROM
bookVenues
                                                        WHERE idStatus
= 2 AND date >= DATE_SUB(CURDATE(), INTERVAL 12 MONTH)
                                                        GROUP BY month
                                                        ORDER BY month
ASC")->fetchAll();
        return $stats;
    }
}
```

BookTour.php

```
<?php

class BookTour
{
    private $pdo;

    public function __construct(PDO $pdo)
    {
        $this->pdo = $pdo;
    }

    /**
     * Επιστρέφει όλες τις επιβεβαιωμένες κρατήσεις ('accepted')
     * μέσα σε ένα συγκεκριμένο εύρος ημερομηνιών.
     * @param string $start_date (μορφή Y-m-d)
     * @param string $end_date (μορφή Y-m-d)
     * @return array
     */
    public function getAcceptedBookingsByDateRange(string $start_date,
string $end_date): array
```

```
    {
        // Τώρα φέρνουμε όλα τα πεδία με bt.*
        $sql = "SELECT bt.*
                FROM bookTours bt
                WHERE bt.idStatus = 2
                AND bt.date BETWEEN ? AND ?";

        $stmt = $this->pdo->prepare($sql);
        $stmt->execute([$start_date, $end_date]);
        return $stmt->fetchAll();
    }

    /**
     * Δημιουργεί μια νέα εγγραφή κράτησης.
     * Χειρίζεται και κρατήσεις από χρήστες και από managers.
     */
    public function create(array $data): int|false
    {
        $sql = "INSERT INTO bookTours (
                    idTour, idStatus, date, children, class,
                    idUser, created_by_idUser,
                    on_behalf_of_name, on_behalf_of_email,
on_behalf_of_phone, on_behalf_of_school
                ) VALUES (
                    :idTour, :idStatus, :date, :children, :class,
                    :idUser, :created_by_idUser,
                    :on_behalf_of_name, :on_behalf_of_email,
:on_behalf_of_phone, :on_behalf_of_school
                )";

        $stmt = $this->pdo->prepare($sql);

        // Ορίζουμε τις παραμέτρους με βάση τα δεδομένα που ήρθαν
        $params = [
            ':idTour'   => $data['idTour'],
            ':idStatus' => $data['idStatus'],
            ':date'     => $data['date'],
            ':children' => $data['children'],
            ':class'    => $data['class'],
            // Αν είναι κράτηση από χρήστη, το idUser έχει τιμή.
            // Αν είναι από manager, το idUser είναι null.
            ':idUser'   => $data['idUser'] ?? null,
            ':created_by_idUser' => $data['created_by_idUser'] ?? null,
            ':on_behalf_of_name' => $data['on_behalf_of_name'] ?? null,
            ':on_behalf_of_email' => $data['on_behalf_of_email'] ??
null,
            ':on_behalf_of_phone' => $data['on_behalf_of_phone'] ??
null,
```

```
            ':on_behalf_of_school' => $data['on_behalf_of_school'] ??
null,
        ];

        try {
            if ($stmt->execute($params)) {
                return $this->pdo->lastInsertId();
            }
            return false;
        } catch (PDOException $e) {
            error_log($e->getMessage());
            return false;
        }
    }


    /**
     * Ελέγχει αν υπάρχει ήδη μια 'accepted' κράτηση για ένα
συγκεκριμένο tour σε μια συγκεκριμένη ημερομηνία.
     */
    public function doesBookingExist(int $idTour, string $date): bool
    {
        $sql = "SELECT 1 FROM bookTours WHERE idTour = ? AND date = ?
AND idStatus = 2 LIMIT 1";
        $stmt = $this->pdo->prepare($sql);
        $stmt->execute([$idTour, $date]);

        // Αν το fetch() επιστρέψει κάτι, σημαίνει ότι βρέθηκε εγγραφή,
άρα υπάρχει.
        return $stmt->fetch() !== false;
    }

    /**
     * Επιστρέφει όλες τις κρατήσεις με λεπτομερή στοιχεία.
     */
    public function getAllBookingsDetails(): array
    {
        $sql = "SELECT
                    bt.idBookTour, bt.date, bt.children, bt.class,
                    bt.on_behalf_of_name, bt.on_behalf_of_school,
                    t.tourName,
                    s.statusName,
                    u.username, u.school
                FROM bookTours bt
                JOIN tours t ON bt.idTour = t.idTour
                JOIN statuses s ON bt.idStatus = s.idStatus
                LEFT JOIN users u ON bt.idUser = u.idUser
                ORDER BY bt.date DESC";
```

```
        $stmt = $this->pdo->query($sql);
        return $stmt->fetchAll();
    }

    /**
     * Βρίσκει μια κράτηση με βάση το ID της.
     */
    public function findBookingById(int $id)
    {
        $stmt = $this->pdo->prepare("SELECT * FROM bookTours WHERE
idBookTour = ?");
        $stmt->execute([$id]);
        return $stmt->fetch();
    }

    /**
     * Ενημερώνει μόνο το status μιας κράτησης.
     */
    public function updateStatus(int $id, int $idStatus): bool
    {
        $sql = "UPDATE bookTours SET idStatus = ? WHERE idBookTour =
?";
        $stmt = $this->pdo->prepare($sql);
        try {
            return $stmt->execute([$idStatus, $id]);
        } catch (PDOException $e) {
            error_log($e->getMessage());
            return false;
        }
    }

    public function getBookingsByUserId(int $userId): array
    {
        $sql = "SELECT bt.*, t.tourName, s.statusName
                FROM bookTours bt
                JOIN tours t ON bt.idTour = t.idTour
                JOIN statuses s ON bt.idStatus = s.idStatus
                WHERE bt.idUser = ?
                ORDER BY bt.date DESC";
        $stmt = $this->pdo->prepare($sql);
        $stmt->execute([$userId]);
        return $stmt->fetchAll();
    }

    public function getBookingDetailsById(int $id)
    {
        $sql = "SELECT bt.*, t.tourName, s.statusName
                FROM bookTours bt
```

```
                JOIN tours t ON bt.idTour = t.idTour
                JOIN statuses s ON bt.idStatus = s.idStatus
                WHERE bt.idBookTour = ?";
        $stmt = $this->pdo->prepare($sql);
        $stmt->execute([$id]);
        return $stmt->fetch();
    }

    public function getStats(): array
    {
        $stats = [];
        // Σύνολο κρατήσεων
        $stats['total_bookings'] = $this->pdo->query("SELECT COUNT(*)
FROM bookTours WHERE idStatus = 2")->fetchColumn();
        // Top προγράμματα
        $stats['top_tours'] = $this->pdo->query("SELECT t.tourName,
COUNT(bt.idBookTour) as count
                                                 FROM bookTours bt
                                                 JOIN tours t ON
bt.idTour = t.idTour
                                                 WHERE bt.idStatus = 2
                                                 GROUP BY t.tourName
                                                 ORDER BY count DESC
                                                 LIMIT 5")->fetchAll();
        // Κρατήσεις ανά μήνα
        $stats['bookings_by_month'] = $this->pdo->query("SELECT
DATE_FORMAT(date, '%Y-%m') as month, COUNT(*) as count
                                                         FROM bookTours
                                                         WHERE idStatus
= 2 AND date >= DATE_SUB(CURDATE(), INTERVAL 12 MONTH)
                                                         GROUP BY month
                                                         ORDER BY month
ASC")->fetchAll();
        return $stats;
    }

}
```

Αρχεία Προβολής

`Views/calendars/venues.php`

```
<?php require_once BASE_PATH . '/views/includes/header.inc.php'; ?>

<!-- Begin Page Content -->
<div class="container-fluid">

    <h1 class="h3 mb-4 text-gray-800">Ημερολόγιο Διαθεσιμότητας
Αιθουσών</h1>
```

```
    <?php flash(); ?>

    <div class="row">
        <div class="col-lg-12">
            <div class="card shadow mb-4">
                <div class="card-header py-3">
                    <h6 class="m-0 font-weight-bold text-
primary">Υπόμνημα</h6>
                </div>
                <div class="card-body">
                    <p><span class="badge badge-success"> </span>
Διαθέσιμη ημέρα |
                    <span class="badge badge-warning"> </span>
Μερικώς κλεισμένη |
                    <span class="badge badge-danger"> </span>
Πλήρως κλεισμένη |
                    <span class="badge badge-secondary"> </span>
Μη διαθέσιμη

                    <?php if (is_simple_user()): ?>
                    <br>
                    <i class="fas fa-info-circle text-info" title="Το
αίτημά σας εκκρεμεί"></i> Το αίτημά σας εκκρεμεί |

                    <i class="fas fa-check-circle text-success"
title="Η κράτησή σας έχει εγκριθεί"></i> Η κράτησή σας έχει εγκριθεί



                    <?php endif; ?>
                    </p>
                </div>
            </div>
        </div>
        <div class="col-lg-12">
            <div class="card shadow mb-4">
                <div class="card-body">
                    <div id="calendar"></div>
                </div>
            </div>
        </div>
    </div>
</div>
<!-- /.container-fluid -->

<!-- Venue Booking Modal -->
<div class="modal fade" id="bookingModal" tabindex="-1" role="dialog">
    <div class="modal-dialog" role="document">
```

```
        <div class="modal-content">
            <form id="bookingForm" method="POST" action="<?=
url('/venues/book') ?>">
                <div class="modal-header">
                    <h5 class="modal-title">Αίτημα Κράτησης
Αίθουσας</h5>
                    <button type="button" class="close" data-
dismiss="modal" aria-label="Close">
                        <span aria-hidden="true">&times;</span>
                    </button>
                </div>
                <div class="modal-body">
                    <input type="hidden" name="booking_date"
id="booking_date">

                    <div class="form-group">
                        <label>Ημερομηνία</label>
                        <input type="text" class="form-control"
id="booking_date_display" readonly>
                    </div>

                    <div class="form-group">
                        <label for="booking_venue_id">Επιλογή
Διαθέσιμης Αίθουσας</label>
                        <select class="form-control" name="idVenue"
id="booking_venue_id" required>
                            <!-- Οι επιλογές θα προστεθούν εδώ από το
JS -->
                        </select>
                    </div>

                    <div class="alert alert-info small">
                        Τα στοιχεία σας (Όνομα, Τηλέφωνο, Email,
Εταιρεία) θα ληφθούν αυτόματα από το προφίλ σας. Μετά την υποβολή, το
αίτημά σας θα εξεταστεί από τον διαχειριστή.
                    </div>

                </div>
                <div class="modal-footer">
                    <button type="button" class="btn btn-secondary"
data-dismiss="modal">Ακύρωση</button>
                    <button type="submit" class="btn btn-
primary">Υποβολή Αιτήματος</button>
                </div>
            </form>
        </div>
    </div>
</div>
```

```
<!-- Details Modal (for Manager) -->
<div class="modal fade" id="detailsModal" tabindex="-1" role="dialog">
    <div class="modal-dialog" role="document">
        <div class="modal-content">
            <div class="modal-header">
                <h5 class="modal-title">Λεπτομέρειες Ημέρας</h5>
                <button class="close" type="button" data-
dismiss="modal"><span aria-hidden="true">×</span></button>
            </div>
            <div class="modal-body"></div>
            <div class="modal-footer">
                <button class="btn btn-secondary" type="button" data-
dismiss="modal">Κλείσιμο</button>
            </div>
        </div>
    </div>
</div>

<?php require_once BASE_PATH . '/views/includes/footer.inc.php'; ?>

<script>
document.addEventListener('DOMContentLoaded', function() {
    var calendarEl = document.getElementById('calendar');
    var calendar = new FullCalendar.Calendar(calendarEl, {
        initialView: 'dayGridMonth',
        locale: 'el',
        firstDay: 1,
        headerToolbar: {
            left: 'prev,next today',
            center: 'title',
            right: 'dayGridMonth,dayGridWeek'
        },
        buttonText: {
            today:    'Σήμερα',
            month:    'Μήνας',
            week:     'Εβδομάδα',
            day:      'Ημέρα',
            list:     'Λίστα'
        },
        events: '/api/venue-events',


        dayCellDidMount: function(info) {
            if (info.isToday) {
                info.el.style.backgroundColor = '#fffbdd';
            } else {
                const eventsOnDay = calendar.getEvents().filter(event
=>
```

```
                    event.start.toISOString().split('T')[0] ===
info.date.toISOString().split('T')[0]
                );
                if (eventsOnDay.length === 0) {
                    info.el.style.backgroundColor = '#f8f9fc';
                }
            }
        },

        eventClick: function(info) {
            const props = info.event.extendedProps;
            const selectedDate = info.event.startStr;
            const formattedDate = new
Date(selectedDate).toLocaleDateString('el-GR');

            if (props.is_manager) {
                // --- ΛΟΓΙΚΗ ΓΙΑ MANAGER ---
                const today = new Date();
                today.setHours(0, 0, 0, 0);
                const isPastDate = info.event.start < today;

                if (isPastDate) {
                    // --- ΕΝΕΡΓΕΙΑ ΓΙΑ ΠΑΡΕΛΘΟΝΤΙΚΕΣ ΗΜΕΡΟΜΗΝΙΕΣ ---
                    if (props.details && props.details.some(d =>
d.status === 'booked')) {
                        let detailsHtml = '<ul>';
                        props.details.forEach(function(detail) {
                            if (detail.status === 'booked') {
                                const statusBadge =
detail.booking_status === 'Pending' ? 'badge-info' : 'badge-primary';
                                detailsHtml +=
`<li><strong>${detail.name}:</strong> ${detail.company} <span
class="badge ${statusBadge}">${detail.booking_status}</span></li>`;
                            }
                        });
                        detailsHtml += '</ul>';
                        $('#detailsModal .modal-title').text('Ιστορικό
Κρατήσεων για ' + formattedDate);
                        $('#detailsModal .modal-
body').html(detailsHtml);
                        $('#detailsModal').modal('show');
                    } else {
                        $('#infoModalTitle').text('Ενημέρωση');
                        $('#infoModalBody').text('Δεν υπήρχαν κρατήσεις
για αυτή την ημέρα.');
                        $('#infoModal').modal('show');
                    }
                } else {
```

```
                    // --- ΕΝΕΡΓΕΙΑ ΓΙΑ ΣΗΜΕΡΙΝΕΣ & ΜΕΛΛΟΝΤΙΚΕΣ
ΗΜΕΡΟΜΗΝΙΕΣ ---
                    if (props.details && props.details.length > 0) {
                        let availableHtml = '<h5><i class="fas fa-
check-circle mr-2 text-success"></i>Διαθέσιμες Αίθουσες</h5>';
                        let bookedHtml = '<h5><i class="fas fa-lock mr-
2"></i>Κλεισμένες / Σε Εκκρεμότητα</h5><ul>';
                        let availableCount = 0;
                        let bookedCount = 0;

                        props.details.forEach(function(detail) {
                            if (detail.status === 'available') {
                                const bookingUrl = `<?=
url('/venues/create-booking')
?>?date=${selectedDate}&venue_id=${detail.id}`;
                                availableHtml += `<div class="d-flex
justify-content-between align-items-center mb-
2"><span>${detail.name}</span><a href="${bookingUrl}" class="btn btn-sm
btn-success">Κράτηση</a></div>`;
                                availableCount++;
                            } else {
                                const statusBadge =
detail.booking_status === 'Pending' ? 'badge-info' : 'badge-primary';
                                bookedHtml +=
`<li><strong>${detail.name}:</strong> ${detail.company} <span
class="badge ${statusBadge}">${detail.booking_status}</span></li>`;
                                bookedCount++;
                            }
                        });
                        bookedHtml += '</ul>';

                        if (availableCount === 0) availableHtml =
'<p>Δεν υπάρχουν διαθέσιμες αίθουσες για νέα κράτηση.</p>';
                        if (bookedCount === 0) bookedHtml = ''; else
bookedHtml = '<hr>' + bookedHtml;

                        $('#detailsModal .modal-title').text('Κατάσταση
Ημέρας: ' + formattedDate);
                        $('#detailsModal .modal-
body').html(availableHtml + bookedHtml);
                        $('#detailsModal').modal('show');
                    }
                }
            } else {
                // --- ΛΟΓΙΚΗ ΓΙΑ ΑΠΛΟ ΧΡΗΣΤΗ ---
                if (props.is_bookable_user) {
                    document.getElementById('booking_date').value =
selectedDate;
```

```
                    document.getElementById('booking_date_display').val
ue = formattedDate;
                    const venueSelect =
document.getElementById('booking_venue_id');
                    venueSelect.innerHTML = '<option value="">Φόρτωση
αιθουσών...</option>';
                    $('#bookingModal').modal('show');
                    fetch('/api/available-venues?date=' + selectedDate)
                        .then(response => response.json())
                        .then(data => {
                            venueSelect.innerHTML = '';
                            if (data.length > 0) {
                                data.forEach(venue => {
                                    const option =
document.createElement('option');
                                    option.value = venue.id;
                                    option.textContent = venue.name;
                                    venueSelect.appendChild(option);
                                });
                            } else {
                                venueSelect.innerHTML = '<option
value="">Δεν βρέθηκαν διαθέσιμες αίθουσες</option>';
                            }
                        });
                } else {
                    if (props.user_has_pending) {
                        $('#infoModalTitle').text('Ενημέρωση');
                        $('#infoModalBody').text('Έχετε ήδη ένα αίτημα
σε εκκρεμότητα για αυτή την ημέρα. Μπορείτε να υποβάλετε και νέα
αιτήματα για άλλες διαθέσιμες αίθουσες.');
                        $('#infoModal').modal('show');
                    } else {
                        // Αλλιώς, δείξε το γενικό μήνυμα
                        $('#infoModalTitle').text('Μη Διαθέσιμη
Ημέρα');
                        $('#infoModalBody').text('Αυτή η ημέρα δεν
είναι διαθέσιμη για νέα κράτηση.');
                        $('#infoModal').modal('show');
                    }
                }
            }
        },

        eventContent: function(arg) {
            const props = arg.event.extendedProps;
            let iconHtml = '';

            // --- ΛΟΓΙΚΗ ΓΙΑ ΤΟ ΕΙΚΟΝΙΔΙΟ ---
```

```
            if (!props.is_manager) {
                if (props.user_booking_status === 'pending') {
                    // Αν ο χρήστης έχει pending αίτημα, βάζουμε μπλε
εικονίδιο
                    iconHtml = ' <i class="fas fa-info-circle text-
info" title="Το αίτημά σας εκκρεμεί"></i>';
                } else if (props.user_booking_status === 'accepted') {
                    // Αν η κράτησή του έχει εγκριθεί, βάζουμε πράσινο
εικονίδιο
                    iconHtml = ' <i class="fas fa-check-circle text-
success" title="Η κράτησή σας έχει εγκριθεί"></i>';
                }
            }

            return {
                html: `<div class="fc-event-
title">${arg.event.title}${iconHtml}</div>`
            }
        },

        eventMouseEnter: function(info) {
            const props = info.event.extendedProps;
            let isClickable = false;

            if (props.is_manager) {
                // Για τον manager, όλες οι μελλοντικές ημέρες είναι
κλικαμπλ
                const today = new Date(); today.setHours(0,0,0,0);
                if (info.event.start >= today) {
                    isClickable = true;
                }

                // Δημιουργία tooltip για τον manager
                if (props.details && props.details.length > 0) {
                    let tooltipText = '';
                    props.details.forEach(function(detail) {
                        if (detail.status === 'booked') {
                            tooltipText += `${detail.name}:
${detail.company} (${detail.booking_status})\n`;
                        } else {
                            tooltipText += `${detail.name}:
Διαθέσιμη\n`;
                        }
                    });
                    info.el.setAttribute('title', tooltipText.trim());
                }

            } else {
```

```
                // Για τον απλό χρήστη, μόνο οι bookable είναι κλικαμπλ
                if (props.is_bookable_user) {
                    isClickable = true;
                }
            }

            if (isClickable) {
                info.el.style.cursor = 'pointer';
            }
        }
    });
    calendar.render();
});
</script>
```

Views/calendars/tours.php

```
<?php require_once BASE_PATH . '/views/includes/header.inc.php'; ?>

<!-- Begin Page Content -->
<div class="container-fluid">

    <h1 class="h3 mb-4 text-gray-800">Ημερολόγιο Σχολικών
Επισκέψεων</h1>
    <?php flash(); ?>

    <div class="row">
        <div class="col-lg-12">
            <div class="card shadow mb-4">
                <div class="card-header py-3">
                    <h6 class="m-0 font-weight-bold text-
primary">Υπόμνημα</h6>
                </div>
                <div class="card-body">
                    <p><span class="badge badge-success"> </span>
Διαθέσιμη ημέρα |
                    <span class="badge badge-warning"> </span>
Μερικώς κλεισμένη |
                    <span class="badge badge-danger"> </span>
Πλήρως κλεισμένη |
                    <span class="badge badge-secondary"> </span>
Μη διαθέσιμη</p>
                </div>
            </div>
        </div>
        <div class="col-lg-12">
            <div class="card shadow mb-4">
                <div class="card-body">
                    <div id="calendar"></div>
```

```
                </div>
            </div>
        </div>
    </div>
</div>
<!-- /.container-fluid -->


<!-- Booking Modal -->
<div class="modal fade" id="bookingModal" tabindex="-1" role="dialog"
aria-labelledby="bookingModalLabel" aria-hidden="true">
    <div class="modal-dialog" role="document">
        <div class="modal-content">
            <form id="bookingForm" method="POST" action="<?=
url('/tours/book') ?>">
                <div class="modal-header">
                    <h5 class="modal-title"
id="bookingModalLabel">Αίτημα Κράτησης για Σχολική Επίσκεψη</h5>
                    <button type="button" class="close" data-
dismiss="modal" aria-label="Close">
                        <span aria-hidden="true">&times;</span>
                    </button>
                </div>
                <div class="modal-body">
                    <!-- Το περιεχόμενο της φόρμας θα γεμίσει δυναμικά
από το JavaScript -->
                    <input type="hidden" name="booking_date"
id="booking_date">

                    <div class="form-group">
                        <label for="booking_tour_id">Επιλογή
Προγράμματος</label>
                        <select class="form-control" name="idTour"
id="booking_tour_id" required>
                            <!-- Οι επιλογές θα προστεθούν εδώ από το
JS -->
                        </select>
                    </div>

                    <div class="form-row">
                        <div class="form-group col-md-6">
                            <label for="booking_children">Αριθμός
Παιδιών</label>
                            <input type="number" class="form-control"
name="children" id="booking_children" min="1" required>
                        </div>
                        <div class="form-group col-md-6">
                            <label for="booking_class">Τάξη</label>
```

```
                            <input type="text" class="form-control"
name="class" id="booking_class" placeholder="π.χ., Γ' Δημοτικού"
required>
                        </div>
                    </div>

                    <div class="alert alert-info small">
                        Τα στοιχεία επικοινωνίας (Όνομα, Τηλέφωνο,
Email, Σχολείο) θα ληφθούν αυτόματα από το προφίλ σας.
                    </div>

                </div>
                <div class="modal-footer">
                    <button type="button" class="btn btn-secondary"
data-dismiss="modal">Ακύρωση</button>
                    <button type="submit" class="btn btn-
primary">Υποβολή Αιτήματος</button>
                </div>
            </form>
        </div>
    </div>
</div>
<!-- Details Modal (for Manager) -->
<div class="modal fade" id="detailsModal" tabindex="-1" role="dialog">
    <div class="modal-dialog" role="document">
        <div class="modal-content">
            <div class="modal-header">
                <h5 class="modal-title">Λεπτομέρειες Ημέρας</h5>
                <button class="close" type="button" data-
dismiss="modal"><span aria-hidden="true">×</span></button>
            </div>
            <div class="modal-body"></div>
            <div class="modal-footer">
                <button class="btn btn-secondary" type="button" data-
dismiss="modal">Κλείσιμο</button>
            </div>
        </div>
    </div>
</div>

<?php require_once BASE_PATH . '/views/includes/footer.inc.php'; ?>



<script>
document.addEventListener('DOMContentLoaded', function() {
    var calendarEl = document.getElementById('calendar');
    var calendar = new FullCalendar.Calendar(calendarEl, {
        initialView: 'dayGridMonth',
```

```
        locale: 'el',
        firstDay: 1,
        headerToolbar: {
            left: 'prev,next today',
            center: 'title',
            right: 'dayGridMonth,dayGridWeek'
        },
        buttonText: {
            today:    'Σήμερα',
            month:    'Μήνας',
            week:     'Εβδομάδα',
            day:      'Ημέρα',
            list:     'Λίστα'
        },
        events: '/api/tour-events',


        dayCellDidMount: function(info) {
            if (info.isToday) {
                info.el.style.backgroundColor = '#fffbdd';
            } else {
                const eventsOnDay = calendar.getEvents().filter(event
=>
                    event.start.toISOString().split('T')[0] ===
info.date.toISOString().split('T')[0]
                );
                if (eventsOnDay.length === 0) {
                    info.el.style.backgroundColor = '#f8f9fc';
                }
            }
        },

        eventClick: function(info) {
            const props = info.event.extendedProps;
            const selectedDate = info.event.startStr;
            const formattedDate = new
Date(selectedDate).toLocaleDateString('el-GR');

            if (props.is_manager) {
                // --- ΛΟΓΙΚΗ ΓΙΑ MANAGER ---
                const today = new Date();
                today.setHours(0, 0, 0, 0);
                const isPastDate = info.event.start < today;

                if (isPastDate) {
                    // --- ΕΝΕΡΓΕΙΑ ΓΙΑ ΠΑΡΕΛΘΟΝΤΙΚΕΣ ΗΜΕΡΟΜΗΝΙΕΣ ---
                    if (props.details && props.details.some(d =>
d.status === 'booked')) {
```

```
                        let detailsHtml = '<ul>';
                        props.details.forEach(function(detail) {
                            if (detail.status === 'booked') {
                                detailsHtml +=
`<li><strong>${detail.name}:</strong> Κλεισμένο από
${detail.school}</li>`;
                            }
                        });
                        detailsHtml += '</ul>';
                        $('#detailsModal .modal-title').text('Ιστορικό
Κρατήσεων για ' + formattedDate);
                        $('#detailsModal .modal-
body').html(detailsHtml);
                        $('#detailsModal').modal('show');
                    } else {
                        $('#infoModalTitle').text('Ενημέρωση');
                        $('#infoModalBody').text('Δεν υπήρχαν κρατήσεις
για αυτή την ημέρα.');
                        $('#infoModal').modal('show');
                    }
                } else {
                    // --- ΕΝΕΡΓΕΙΑ ΓΙΑ ΣΗΜΕΡΙΝΕΣ & ΜΕΛΛΟΝΤΙΚΕΣ
ΗΜΕΡΟΜΗΝΙΕΣ ---
                    if (props.details && props.details.length > 0) {
                        let availableHtml = '<h5><i class="fas fa-
check-circle mr-2 text-success"></i>Διαθέσιμα Προγράμματα</h5>';
                        let bookedHtml = '<h5><i class="fas fa-lock mr-
2"></i>Κλεισμένα Προγράμματα</h5><ul>';
                        let availableCount = 0;
                        let bookedCount = 0;

                        props.details.forEach(function(detail) {
                            if (detail.status === 'available') {
                                const bookingUrl = `<?=
url('/tours/create-booking')
?>?date=${selectedDate}&tour_id=${detail.id}`;
                                availableHtml += `<div class="d-flex
justify-content-between align-items-center mb-
2"><span>${detail.name}</span><a href="${bookingUrl}" class="btn btn-sm
btn-success">Κράτηση</a></div>`;
                                availableCount++;
                            } else {
                                bookedHtml +=
`<li><strong>${detail.name}:</strong> ${detail.school}</li>`;
                                bookedCount++;
                            }
                        });
                        bookedHtml += '</ul>';
```

```
                        if (availableCount === 0) availableHtml =
'<p>Δεν υπάρχουν διαθέσιμα προγράμματα για νέα κράτηση.</p>';
                        if (bookedCount === 0) bookedHtml = ''; else
bookedHtml = '<hr>' + bookedHtml;

                        $('#detailsModal .modal-title').text('Κατάσταση
Ημέρας: ' + formattedDate);
                        $('#detailsModal .modal-
body').html(availableHtml + bookedHtml);
                        $('#detailsModal').modal('show');
                    }
                }

            } else {
                // --- ΛΟΓΙΚΗ ΓΙΑ ΑΠΛΟ ΧΡΗΣΤΗ ---
                if (props.is_bookable_user) {
                    document.getElementById('booking_date').value =
selectedDate;
                    const tourSelect =
document.getElementById('booking_tour_id');
                    tourSelect.innerHTML = '<option
value="">Φόρτωση...</option>';
                    $('#bookingModal').modal('show');
                    fetch('/api/available-tours?date=' + selectedDate)
                        .then(response => response.json())
                        .then(data => {
                            tourSelect.innerHTML = '';
                            if (data.length > 0) {
                                data.forEach(tour => {
                                    const option =
document.createElement('option');
                                    option.value = tour.id;
                                    option.textContent = tour.name;
                                    tourSelect.appendChild(option);
                                });
                            } else {
                                tourSelect.innerHTML = '<option
value="">Δεν βρέθηκαν διαθέσιμα προγράμματα</option>';
                            }
                        });
                } else {
                    $('#infoModalTitle').text('Μη Διαθέσιμη Ημέρα');
                    $('#infoModalBody').text(info.event.extendedProps.r
eason || 'Αυτή η ημέρα δεν είναι διαθέσιμη για νέα κράτηση.');
                    $('#infoModal').modal('show');
                }
            }
```

```
        },

        eventMouseEnter: function(info) {
            const props = info.event.extendedProps;
            if (props.is_manager && props.details &&
props.details.length > 0) {
                let tooltipText = '';
                props.details.forEach(function(detail) {
                    if (detail.status === 'booked') {
                        tooltipText += `${detail.name}: Κλεισμένο
(${detail.school})\n`;
                    } else {
                        tooltipText += `${detail.name}: Διαθέσιμο\n`;
                    }
                });
                info.el.setAttribute('title', tooltipText.trim());
            }

            // Αλλάζουμε το στυλ του κέρσορα
            if ((props.is_manager && info.event.start >= new Date()) ||
(!props.is_manager && props.is_bookable_user)) {
                info.el.style.cursor = 'pointer';
            }
        }
    });
    calendar.render();
});
</script>
```

Γ.8 Καταγραφή Ενεργειών (Activity Logs)

Αρχείο Μοντέλου (`app/models/ActivityLog.php`)

```
<?php

class ActivityLog
{
    private $pdo;

    public function __construct(PDO $pdo)
    {
        $this->pdo = $pdo;
    }

    /**
     * Επιστρέφει τα πιο πρόσφατα logs.
     */
    public function getRecentLogs(int $limit = 100)
    {
```

```
        $sql = "SELECT l.timestamp, l.action_type,
l.action_description, u.username
                FROM activity_logs l
                LEFT JOIN users u ON l.idUser = u.idUser
                ORDER BY l.timestamp DESC
                LIMIT :limit";

        $stmt = $this->pdo->prepare($sql);
        $stmt->bindParam(':limit', $limit, PDO::PARAM_INT);
        $stmt->execute();

        return $stmt->fetchAll();
    }

    /**
     * Γενική μέθοδος για την καταγραφή μιας ενέργειας.
     * @param string $action_type Ο τύπος της ενέργειας (π.χ.,
'LOGIN_SUCCESS').
     * @param string $description Η περιγραφή της ενέργειας.
     * @param int|null $userId Το ID του χρήστη που έκανε την ενέργεια.
     */
    public function logAction(string $action_type, string $description,
?int $userId = null): void
    {
        $sql = "INSERT INTO activity_logs (idUser, action_type,
action_description) VALUES (?, ?, ?)";
        $stmt = $this->pdo->prepare($sql);
        $stmt->execute([$userId, $action_type, $description]);
    }
```

Αρχείο Ελεγκτή (`app/controllers/ActivityLogController.php`)

```
<?php

class ActivityLogController
{
    private $logModel;

    public function __construct(ActivityLog $logModel)
    {
        $this->logModel = $logModel;
    }

    /**
     * Εμφανίζει τη σελίδα με τα πιο πρόσφατα logs.
     */
    public function index()
    {
        // Μόνο ο super admin μπορεί να δει τα logs
        if (!is_super_admin()) {
```

```
            http_response_code(403);
            require_once BASE_PATH . '/views/errors/403.php';
            exit();
        }

        // Παίρνουμε τα logs από το model
        $logs = $this->logModel->getRecentLogs(100);

        // Φορτώνουμε τη view
        require_once BASE_PATH . '/views/logs/index.php';
    }

}
```

Αρχείο Προβολής (`app/views/logs/index.php`)

```
<?php require_once BASE_PATH . '/views/includes/header.inc.php'; ?>

<!-- DataTables CSS -->
<link href="<?=
asset('assets/vendor/datatables/dataTables.bootstrap4.min.css') ?>"
rel="stylesheet">

<!-- Begin Page Content -->
<div class="container-fluid">

    <!-- Page Heading -->
    <h1 class="h3 mb-2 text-gray-800">Αρχείο Καταγραφής
Δραστηριότητας</h1>
    <p class="mb-4">Εδώ εμφανίζονται οι πιο πρόσφατες σημαντικές
ενέργειες που έχουν καταγραφεί στο σύστημα.</p>

    <!-- Logs Table -->
    <div class="card shadow mb-4">
        <div class="card-header py-3">
            <h6 class="m-0 font-weight-bold text-
primary">Καταγραφές</h6>
        </div>
        <div class="card-body">
            <div class="table-responsive">
                <table class="table table-bordered table-striped"
id="dataTable" width="100%" cellspacing="0">
                    <thead class="thead-light">
                        <tr>
                            <th style="width: 15%;">Ημερομηνία &
Ώρα</th>
                            <th style="width: 15%;">Χρήστης</th>
                            <th style="width: 20%;">Τύπος
Ενέργειας</th>
```

```
                            <th>Περιγραφή</th>
                        </tr>
                    </thead>
                    <tbody>
                        <?php foreach ($logs as $log): ?>
                            <tr>
                                <td><?= htmlspecialchars(date('d/m/Y
H:i:s', strtotime($log['timestamp']))) ?></td>
                                <td>
                                    <?php if ($log['username']): ?>
                                        <span class="badge badge-
info"><?= htmlspecialchars($log['username']) ?></span>
                                    <?php else: ?>
                                        <span class="badge badge-
secondary">Σύστημα</span>
                                    <?php endif; ?>
                                </td>
                                <td>
                                    <?php
                                        $actionType =
htmlspecialchars($log['action_type']);
                                        $badgeClass = 'badge-primary';
// Default χρώμα

                                        if (str_contains($actionType,
'SUCCESS') || str_contains($actionType, 'CREATE')) {
                                            $badgeClass = 'badge-
success'; // Πράσινο για επιτυχία/δημιουργία
                                        } elseif
(str_contains($actionType, 'FAILURE') || str_contains($actionType,
'DELETE')) {
                                            $badgeClass = 'badge-
danger'; // Κόκκινο για αποτυχία/διαγραφή
                                        } elseif
(str_contains($actionType, 'UPDATE') || str_contains($actionType,
'DISABLED')) {
                                            $badgeClass = 'badge-
warning'; // Πορτοκαλί για ενημέρωση/ειδοποίηση
                                        }
                                    ?>
                                    <span class="badge <?= $badgeClass
?>"><?= $actionType ?></span>
                                </td>
                                <td><?=
htmlspecialchars($log['action_description']) ?></td>
                            </tr>
                        <?php endforeach; ?>
```

```
                        <?php if (empty($logs)): ?>
                            <tr>
                                <td colspan="4" class="text-center">Δεν
υπάρχουν καταγραφές.</td>
                            </tr>
                        <?php endif; ?>
                    </tbody>
                </table>
            </div>
        </div>
    </div>

</div>
<!-- /.container-fluid -->

<?php require_once BASE_PATH . '/views/includes/footer.inc.php'; ?>

<!-- Page level plugins for DataTables -->
<script src="<?=
asset('assets/vendor/datatables/jquery.dataTables.min.js')
?>"></script>
<script src="<?=
asset('assets/vendor/datatables/dataTables.bootstrap4.min.js')
?>"></script>

<!-- Page level custom script -->
<script>
$(document).ready(function() {
  $('#dataTable').DataTable({
      "order": [[ 0, "desc" ]],
      // Παραμετροποίηση μηνυμάτων στα Ελληνικά
      "language": {
            "lengthMenu": "Εμφάνιση _MENU_ εγγραφών ανά σελίδα",
            "zeroRecords": "Δεν βρέθηκαν εγγραφές",
            "info": "Εμφάνιση σελίδας _PAGE_ από _PAGES_",
            "infoEmpty": "Δεν υπάρχουν διαθέσιμες εγγραφές",
            "infoFiltered": "(φιλτραρισμένο από _MAX_ συνολικές
εγγραφές)",
            "search": "Αναζήτηση:",
            "paginate": {
                "first": "Πρώτη",
                "last": "Τελευταία",
                "next": "Επόμενη",
                "previous": "Προηγούμενη"
            }
        }
  });
});
```

```
</script>
```

Γ.9 Διαχείριση Ρυθμίσεων (Settings)

Αρχείο Μοντέλου (`app/models/Settings.php`)

```php
<?php

class Settings
{
    private $pdo;

    public function __construct(PDO $pdo)
    {
        $this->pdo = $pdo;
    }

    /**
     * Φέρνει όλες τις ρυθμίσεις από τη βάση και τις επιστρέφει
     * ως έναν associative array (π.χ., ['contact_phone' => '210...']).
     */
    public function getAllAsArray(): array
    {
        $settings = [];
        $stmt = $this->pdo->query("SELECT setting_key, setting_value
FROM settings");

        while ($row = $stmt->fetch()) {
            $settings[$row['setting_key']] = $row['setting_value'];
        }

        return $settings;
    }


    /**
     * Ενημερώνει πολλαπλές ρυθμίσεις ταυτόχρονα.
     * Δέχεται ένα array της μορφής ['setting_key' => 'new_value', ...]
     */
    public function update(array $settings): bool
    {
        $sql = "UPDATE settings SET setting_value = :value WHERE
setting_key = :key";
        $stmt = $this->pdo->prepare($sql);

        $this->pdo->beginTransaction();
        try {
            foreach ($settings as $key => $value) {
                $stmt->execute([':key' => $key, ':value' => $value]);
```

```
            }
            $this->pdo->commit();
            return true;
        } catch (PDOException $e) {
            $this->pdo->rollBack();
            error_log($e->getMessage());
            return false;
        }
    }
}
```

Αρχείο Ελεγκτή (`app/controllers/SettingsController.php`)

```
<?php

class SettingsController
{
    private $settingsModel;
    private $logModel;

    public function __construct(Settings $settingsModel, ActivityLog
$logModel)
    {
        $this->settingsModel = $settingsModel;
        $this->logModel = $logModel;
    }

    /**
     * Εμφανίζει τη σελίδα με τη φόρμα των ρυθμίσεων.
     */
    public function index()
    {
        if (!is_super_admin()) {
            http_response_code(403);
            require_once BASE_PATH . '/views/errors/403.php';
            exit();
        }

        $settings = $this->settingsModel->getAllAsArray();
        require_once BASE_PATH . '/views/settings/index.php';
    }

    /**
     * Αποθηκεύει τις αλλαγές στις ρυθμίσεις.
     */
    public function update()
    {
        if (!is_super_admin()) {
            http_response_code(403);
```

```
            require_once BASE_PATH . '/views/errors/403.php';
            exit();
        }

        // Το $_POST θα περιέχει όλες τις ρυθμίσεις από τη φόρμα
        $success = $this->settingsModel->update($_POST);

        if ($success) {
            flash('success', 'Οι ρυθμίσεις αποθηκεύτηκαν με
επιτυχία.');
            $adminUsername = $_SESSION['username'];
            $this->logModel->logAction(
                'SETTINGS_UPDATE',
                "Ο διαχειριστής '{$adminUsername}' ενημέρωσε τις
ρυθμίσεις της ιστοσελίδας.",
                $_SESSION['user_id']
            );
        } else {
            flash('danger', 'Παρουσιάστηκε σφάλμα κατά την αποθήκευση
των ρυθμίσεων.');
        }

        header('Location: ' . url('/settings'));
        exit();
    }
}
```

Αρχεία Προβολής

Προβολή Διαχειριστή (`app/views/settings/index.php`)

```
<?php require_once BASE_PATH . '/views/includes/header.inc.php'; ?>

<!-- Begin Page Content -->
<div class="container-fluid">

    <h1 class="h3 mb-4 text-gray-800">Ρυθμίσεις Ιστοσελίδας</h1>

    <?php flash(); ?>

    <form action="<?= url('/settings') ?>" method="POST">
        <div class="row">

            <!-- ==== ΣΤΗΛΗ 1: ΣΤΟΙΧΕΙΑ & ΥΠΕΥΘΥΝΟΙ ==== -->
            <div class="col-lg-6">
                <!-- Card for Contact Info -->
                <div class="card shadow mb-4">
                    <div class="card-header py-3">
```

```
                        <h6 class="m-0 font-weight-bold text-
primary">Γενικά Στοιχεία Επικοινωνίας & Διεύθυνση</h6>
                    </div>
                    <div class="card-body">
                        <div class="form-group">
                            <label for="contact_phone">Γενικό
Τηλέφωνο</label>
                            <input type="text" class="form-control"
id="contact_phone" name="contact_phone" value="<?=
htmlspecialchars($settings['contact_phone'] ?? '') ?>">
                        </div>
                        <div class="form-group">
                            <label for="contact_email">Γενικό
Email</label>
                            <input type="email" class="form-control"
id="contact_email" name="contact_email" value="<?=
htmlspecialchars($settings['contact_email'] ?? '') ?>">
                        </div>
                        <div class="form-group">
                            <label for="address_street">Οδός</label>
                            <input type="text" class="form-control"
id="address_street" name="address_street" value="<?=
htmlspecialchars($settings['address_street'] ?? '') ?>">
                        </div>
                        <div class="form-row">
                            <div class="form-group col-md-8">
                                <label for="address_city">Πόλη</label>
                                <input type="text" class="form-control"
id="address_city" name="address_city" value="<?=
htmlspecialchars($settings['address_city'] ?? '') ?>">
                            </div>
                            <div class="form-group col-md-4">
                                <label
for="address_postal">Τ.Κ.</label>
                                <input type="text" class="form-control"
id="address_postal" name="address_postal" value="<?=
htmlspecialchars($settings['address_postal'] ?? '') ?>">
                            </div>
                            <div class="form-group">
                                <label for="google_maps_url">URL
Ενσωμάτωσης Google Maps</label>
                                <textarea class="form-control"
id="google_maps_url" name="google_maps_url" rows="4"><?=
htmlspecialchars($settings['google_maps_url'] ?? '') ?></textarea>
                                <small class="form-text text-
muted">Αντιγράψτε τον κώδικα "Ενσωμάτωση χάρτη" από το Google Maps και
επικολλήστε εδώ μόνο το περιεχόμενο του src="...".</small>
                            </div>
```

```
                        </div>
                    </div>
                </div>

                <!-- Card for Manager Info -->
                <div class="card shadow mb-4">
                    <div class="card-header py-3">
                        <h6 class="m-0 font-weight-bold text-
primary">Στοιχεία Υπευθύνων</h6>
                    </div>
                    <div class="card-body">
                        <div class="form-group">
                            <label for="venue_manager_phone">Τηλέφωνο
Υπεύθυνου Αιθουσών</label>
                            <input type="text" class="form-control"
id="venue_manager_phone" name="venue_manager_phone" value="<?=
htmlspecialchars($settings['venue_manager_phone'] ?? '') ?>">
                        </div>
                         <div class="form-group">
                            <label for="venue_manager_email">Email
Υπεύθυνου Αιθουσών</label>
                            <input type="email" class="form-control"
id="venue_manager_email" name="venue_manager_email" value="<?=
htmlspecialchars($settings['venue_manager_email'] ?? '') ?>">
                        </div>
                        <hr>
                        <div class="form-group">
                            <label for="tour_manager_phone">Τηλέφωνο
Υπεύθυνου Σχ. Επισκέψεων</label>
                            <input type="text" class="form-control"
id="tour_manager_phone" name="tour_manager_phone" value="<?=
htmlspecialchars($settings['tour_manager_phone'] ?? '') ?>">
                        </div>
                         <div class="form-group">
                            <label for="tour_manager_email">Email
Υπεύθυνου Σχ. Επισκέψεων</label>
                            <input type="email" class="form-control"
id="tour_manager_email" name="tour_manager_email" value="<?=
htmlspecialchars($settings['tour_manager_email'] ?? '') ?>">
                        </div>
                         <hr>
                        <div class="form-group">
                            <label for="admin_phone">Τηλέφωνο
Διαχειριστή (για επαναφορά κωδικού)</label>
                            <input type="text" class="form-control"
id="admin_phone" name="admin_phone" value="<?=
htmlspecialchars($settings['admin_phone'] ?? '') ?>">
                        </div>
```

```
                        <div class="form-group">
                            <label for="admin_email">Email Διαχειριστή
(για επαναφορά κωδικού)</label>
                            <input type="email" class="form-control"
id="admin_email" name="admin_email" value="<?=
htmlspecialchars($settings['admin_email'] ?? '') ?>">
                        </div>
                    </div>
                </div>
            </div>

            <!-- ==== ΣΤΗΛΗ 2: ΩΡΑΡΙΑ ==== -->
            <div class="col-lg-6">
                 <!-- Card for Museum Hours -->
                <div class="card shadow mb-4">
                    <div class="card-header py-3">
                        <h6 class="m-0 font-weight-bold text-
primary">Ωράριο Λειτουργίας Μουσείου (για το κοινό)</h6>
                    </div>
                    <div class="card-body">
                        <?php $days = ['monday', 'tuesday',
'wednesday', 'thursday', 'friday', 'saturday', 'sunday'];
                              $days_gr = ['Δευτέρα', 'Τρίτη',
'Τετάρτη', 'Πέμπτη', 'Παρασκευή', 'Σάββατο', 'Κυριακή']; ?>
                        <?php foreach($days as $index => $day): ?>
                        <div class="form-group">
                            <label for="<?= $day ?>"><?=
$days_gr[$index] ?></label>
                            <input type="text" class="form-control"
id="<?= $day ?>" name="<?= $day ?>" value="<?=
htmlspecialchars($settings[$day] ?? '') ?>">
                        </div>
                        <?php endforeach; ?>
                    </div>
                </div>

                 <!-- Card for Staff Hours -->
                <div class="card shadow mb-4">
                    <div class="card-header py-3">
                        <h6 class="m-0 font-weight-bold text-
primary">Ωράριο Προσωπικού (για τηλ. επικοινωνία)</h6>
                    </div>
                    <div class="card-body">
                         <?php $staff_days = ['staff_monday',
'staff_tuesday', 'staff_wednesday', 'staff_thursday', 'staff_friday',
'staff_saturday', 'staff_sunday']; ?>
                        <?php foreach($staff_days as $index => $day):
?>
```

```
                        <div class="form-group">
                            <label for="<?= $day ?>"><?=
$days_gr[$index] ?></label>
                            <input type="text" class="form-control"
id="<?= $day ?>" name="<?= $day ?>" value="<?=
htmlspecialchars($settings[$day] ?? '') ?>">
                        </div>
                        <?php endforeach; ?>
                    </div>
                </div>
            </div>
        </div>

        <div class="card shadow mb-4">
            <div class="card-body text-center">
                 <button type="submit" class="btn btn-primary btn-lg">
                    <i class="fas fa-save"></i> Αποθήκευση Ρυθμίσεων
                </button>
            </div>
        </div>
    </form>
</div>
<!-- /.container-fluid -->

<?php require_once BASE_PATH . '/views/includes/footer.inc.php'; ?>
```

Δημόσια Προβολή (`app/views/pages/contact.php`)

```
<?php require_once BASE_PATH . '/views/includes/header.inc.php'; ?>

<style>
.map-responsive {
    overflow: hidden;
    padding-bottom: 112.5%;
    position: relative;
    height: 0;
}
.map-responsive iframe {
    left: 0;
    top: 0;
    height: 100%;
    width: 100%;
    position: absolute;
}
</style>


<!-- Begin Page Content -->
<div class="container-fluid">
```

```
    <!-- Page Heading -->
    <h1 class="h3 mb-4 text-gray-800">Επικοινωνία</h1>

    <div class="row">

        <!-- Contact Info Column -->
        <div class="col-lg-8 mb-4">
            <div class="card shadow">
                <div class="card-header py-3">
                    <h6 class="m-0 font-weight-bold text-
primary">Στοιχεία Επικοινωνίας</h6>
                </div>
                <div class="card-body">
                    <p>Για οποιαδήποτε πληροφορία σχετικά με τις
κρατήσεις αιθουσών, τις σχολικές επισκέψεις ή για γενικές ερωτήσεις, μη
διστάσετε να επικοινωνήσετε μαζί μας.</p>
                    <hr>
                    <p>
                        <i class="fas fa-map-marker-alt fa-fw mr-2
text-gray-500"></i>
                        <strong>Διεύθυνση:</strong>
                        <?=
htmlspecialchars($settings['address_street']) ?>, <?=
htmlspecialchars($settings['address_city']) ?>, <?=
htmlspecialchars($settings['address_postal']) ?>
                    </p>
                    <p>
                        <i class="fas fa-phone fa-fw mr-2 text-gray-
500"></i>
                        <strong>Τηλέφωνο:</strong>
                        <a href="tel:<?=
htmlspecialchars($settings['contact_phone']) ?>"><?=
htmlspecialchars($settings['contact_phone']) ?></a>
                    </p>
                    <p>
                        <i class="fas fa-envelope fa-fw mr-2 text-gray-
500"></i>
                        <strong>Email:</strong>
                        <a href="mailto:<?=
htmlspecialchars($settings['contact_email']) ?>"><?=
htmlspecialchars($settings['contact_email']) ?></a>
                    </p>
                    <hr>
                    <h6 class="font-weight-bold"><u>Ωράριο
Λειτουργίας</u></h6>
                    <ul class="list-unstyled">
```

```
                        <li><strong>Δευτέρα:</strong> <?=
htmlspecialchars($settings['monday']) ?></li>
                        <li><strong>Τρίτη:</strong> <?=
htmlspecialchars($settings['tuesday']) ?></li>
                        <li><strong>Τετάρτη:</strong> <?=
htmlspecialchars($settings['wednesday']) ?></li>
                        <li><strong>Πέμπτη:</strong> <?=
htmlspecialchars($settings['thursday']) ?></li>
                        <li><strong>Παρασκευή:</strong> <?=
htmlspecialchars($settings['friday']) ?></li>
                        <li><strong>Σάββατο:</strong> <?=
htmlspecialchars($settings['saturday']) ?></li>
                        <li><strong>Κυριακή:</strong> <?=
htmlspecialchars($settings['sunday']) ?></li>
                    </ul>
                    <hr>
                    <h6 class="font-weight-bold"><u>Τμήματα</u></h6>
                    <p>
                        <i class="fas fa-fw fa-podcast"></i>
                        <strong>Τμήμα Αιθουσών Εκδηλώσεων</strong>
                        <br>
                        <i class="fas fa-phone fa-fw mr-2 text-gray-
500"></i>
                        <strong>Τηλέφωνο:</strong>
                        <a href="tel:<?=
htmlspecialchars($settings['venue_manager_phone'] ?? '') ?>"><?=
htmlspecialchars($settings['venue_manager_phone'] ?? '') ?></a>
                        <br>
                        <i class="fas fa-envelope fa-fw mr-2 text-gray-
500"></i>
                        <strong>Email:</strong>
                        <a href="mailto:<?=
htmlspecialchars($settings['venue_manager_email'] ?? '') ?>"><?=
htmlspecialchars($settings['venue_manager_email'] ?? '') ?></a>
                    </p>
                    <p>
                        <i class="fas fa-fw fa-school"></i>
                        <strong>Τμήμα Σχολικών Επισκέψεων</strong>
                        <br>
                        <i class="fas fa-phone fa-fw mr-2 text-gray-
500"></i>
                        <strong>Τηλέφωνο:</strong>
                        <a href="tel:<?=
htmlspecialchars($settings['tour_manager_phone'] ?? '') ?>"><?=
htmlspecialchars($settings['tour_manager_phone'] ?? '') ?></a>
                        <br>
                        <i class="fas fa-envelope fa-fw mr-2 text-gray-
500"></i>
```

```
                        <strong>Email:</strong>
                        <a href="mailto:<?=
htmlspecialchars($settings['tour_manager_email'] ?? '') ?>"><?=
htmlspecialchars($settings['tour_manager_email'] ?? '') ?></a>
                    </p>
                    <p>
                        <i class="fas fa-fw fa-cog"></i>
                        <strong>Διαχείριση εφαρμογής</strong>
                        <br>
                        <i class="fas fa-phone fa-fw mr-2 text-gray-
500"></i>
                        <strong>Τηλέφωνο:</strong>
                        <a href="tel:<?=
htmlspecialchars($settings['admin_phone'] ?? '') ?>"><?=
htmlspecialchars($settings['admin_phone'] ?? '') ?></a>
                        <br>
                        <i class="fas fa-envelope fa-fw mr-2 text-gray-
500"></i>
                        <strong>Email:</strong>
                        <a href="mailto:<?=
htmlspecialchars($settings['admin_email'] ?? '') ?>"><?=
htmlspecialchars($settings['admin_email'] ?? '') ?></a>
                    </p>
                    <hr>
                     <h6 class="font-weight-bold"><u>Ωράριο Προσωπικού
για Τηλεφωνική Επικοινωνία</u></h6>
                     <ul class="list-unstyled">
                        <li><strong>Δευτέρα:</strong> <?=
htmlspecialchars($settings['staff_monday']) ?></li>
                        <li><strong>Τρίτη:</strong> <?=
htmlspecialchars($settings['staff_tuesday']) ?></li>
                        <li><strong>Τετάρτη:</strong> <?=
htmlspecialchars($settings['staff_wednesday']) ?></li>
                        <li><strong>Πέμπτη:</strong> <?=
htmlspecialchars($settings['staff_thursday']) ?></li>
                        <li><strong>Παρασκευή:</strong> <?=
htmlspecialchars($settings['staff_friday']) ?></li>
                        <li><strong>Σάββατο:</strong> <?=
htmlspecialchars($settings['staff_saturday']) ?></li>
                        <li><strong>Κυριακή:</strong> <?=
htmlspecialchars($settings['staff_sunday']) ?></li>
                    </ul>
                </div>
            </div>
        </div>

        <!-- Map Column -->
        <div class="col-lg-4 mb-4">
```

```
            <div class="card shadow">
                <div class="card-header py-3">
                    <h6 class="m-0 font-weight-bold text-
primary">Βρείτε μας στον Χάρτη</h6>
                </div>
                <div class="map-responsive">
                    <iframe src="<?=
htmlspecialchars($settings['google_maps_url'] ?? '') ?>"
style="border:0;" allowfullscreen="" loading="lazy" referrerpolicy="no-
referrer-when-downgrade"></iframe>
                </div>
            </div>
        </div>
    </div>
</div>
<!-- /.container-fluid -->

<?php require_once BASE_PATH . '/views/includes/footer.inc.php'; ?>
```

Γ.10 Δημόσιες Σελίδες και Στατιστικά

Αρχεία Ελεγκτών

`app/controllers/PageController.php`

```
<?php
class PageController
{

    private $tourModel;
    private $venueModel;
    private $photoModel;
    private $settingsModel;

    public function __construct(Tour $tourModel, Venue $venueModel,
Photo $photoModel, Settings $settingsModel)
    {
        $this->tourModel = $tourModel;
        $this->venueModel = $venueModel;
        $this->photoModel = $photoModel;
        $this->settingsModel = $settingsModel;
    }

    public function home()
    {
        require_once BASE_PATH . '/views/pages/home.php';
    }

    /**
     * Εμφανίζει τη σελίδα για τις σχολικές επισκέψεις.
```

```
     */
    public function schoolTours()
    {
        // 1. Ζητάμε από το Model να μας δώσει όλες τις περιόδους
        $tours = $this->tourModel->getAll(true);

        // 2. Φορτώνουμε το View και του "δίνουμε" τα δεδομένα ($tours)
        require_once BASE_PATH . '/views/pages/school-tours.php';
    }


    /**
     * Εμφανίζει τη σελίδα επικοινωνίας.
     */
    public function contact()
    {
        // 1. Παίρνουμε όλες τις ρυθμίσεις από το model
        $settings = $this->settingsModel->getAllAsArray();

        // 2. Φορτώνουμε τη view, δίνοντάς της τον πίνακα με τις
ρυθμίσεις
        require_once BASE_PATH . '/views/pages/contact.php';
    }

    // Μέθοδος για την προβολή των αιθουσών
    public function venues()
    {
        // 1. Παίρνουμε όλες τις αίθουσες
        $venues = $this->venueModel->getAll(true);

        // 2. Για κάθε αίθουσα, βρίσκουμε τις φωτογραφίες της
        foreach ($venues as $key => $venue) {
            $venues[$key]['photos'] = $this->photoModel-
>findByVenueId($venue['idVenue']);
        }

        // 3. Φορτώνουμε τη view, δίνοντάς της τον πίνακα με τις
αίθουσες και τις φωτογραφίες τους
        require_once BASE_PATH . '/views/pages/venues-list.php';
    }

    /**
     * Εμφανίζει τη σελίδα με τις λεπτομέρειες για μία συγκεκριμένη
αίθουσα.
     */
    public function venueDetail(int $id)
    {
        // 1. Ζητάμε από το Model να βρει την αίθουσα με το
συγκεκριμένο ID.
```

```
        $venue = $this->venueModel->findById($id);

        // 2. Έλεγχος Ασφαλείας:
        // Αν η αίθουσα δεν βρέθηκε (π.χ. λάθος ID) ή αν δεν είναι
ενεργή,
        // τότε δείχνουμε σφάλμα 404 (Δεν Βρέθηκε)
        if (!$venue || !$venue['active']) {
            http_response_code(404);
            require_once BASE_PATH . '/views/errors/404.php';
            exit();
        }

        // 3. Αν η αίθουσα βρέθηκε, ζητάμε και όλες τις φωτογραφίες
της.
        $photos = $this->photoModel->findByVenueId($id);

        // 4. Περνάμε και τα δύο σύνολα δεδομένων ($venue και $photos)
στο view.
        require_once BASE_PATH . '/views/pages/venue-detail.php';
    }

    /**
     * Εμφανίζει τη σελίδα του ημερολογίου για τις σχολικές επισκέψεις.
     */
    public function toursCalendar()
    {
        // Ελέγχουμε αν ο χρήστης είναι συνδεδεμένος και αν είναι είτε
user, είτε tour manager, είτε admin.
        if (!is_logged_in()) {
            header('Location: ' . url('/login'));
            exit();
        }

        require_once BASE_PATH . '/views/calendars/tours.php';
    }

    /**
     * Εμφανίζει τη σελίδα του ημερολογίου για τις αίθουσες εκδηλώσεων.
     */
    public function venuesCalendar()
    {
        if (!is_logged_in()) {
            header('Location: ' . url('/login'));
            exit();
        }

        require_once BASE_PATH . '/views/calendars/venues.php';
    }
```

```
}
```

app/controllers/StatsController.php

```php
<?php
class StatsController
{
    private $bookTourModel;
    private $bookVenueModel;

    public function __construct(BookTour $bookTourModel, BookVenue
$bookVenueModel)
    {
        $this->bookTourModel = $bookTourModel;
        $this->bookVenueModel = $bookVenueModel;
    }

    public function index()
    {
        $tourStats = null;
        $venueStats = null;

        if (is_super_admin() || is_tour_manager()) {
            $tourStats = $this->bookTourModel->getStats();
        }

        if (is_super_admin() || is_venue_manager()) {
            $venueStats = $this->bookVenueModel->getStats();
        }

        require_once BASE_PATH . '/views/stats/index.php';
    }
}
```

Αρχεία Προβολής

app/views/pages/home.php

```php
<?php require_once BASE_PATH . '/views/includes/header.inc.php'; ?>

<!-- Begin Page Content -->
<div class="container-fluid">

    <!-- Hero Section -->
    <div class="text-center my-5">
        <img src="<?= asset('assets/img/logo.png') ?>" alt="MuseApp
Logo" style="max-width: 250px;" class="mb-4">
        <h1 class="display-4 text-gray-800">Καλώς Ήρθατε στο
MuseApp</h1>
```

```
        <p class="lead">Η πλατφόρμα για την οργάνωση των εκδηλώσεων και των σχολικών σας επισκέψεων στο μουσείο μας.</p>
    </div>

    <!-- Main Content Cards -->
    <div class="row">

        <!-- Card: Αίθουσες Εκδηλώσεων -->
        <div class="col-lg-6 mb-4">
            <div class="card shadow h-100">
                <div class="card-header py-3">
                    <h6 class="m-0 font-weight-bold text-primary"><i class="fas fa-podcast mr-2"></i>Αίθουσες Εκδηλώσεων</h6>
                </div>
                <div class="card-body">
                    <p>Ανακαλύψτε τις σύγχρονες και πλήρως εξοπλισμένες αίθουσές μας, ιδανικές για συνέδρια, ημερίδες, σεμινάρια και εταιρικές εκδηλώσεις. Δείτε τους χώρους και κάντε το αίτημα κράτησής σας online.</p>
                </div>
                <div class="card-footer text-center">
                    <a href="<?= url('/venues-list') ?>" class="btn btn-primary">Προβολή Αιθουσών</a>
                </div>
            </div>
        </div>

        <!-- Card: Σχολικές Επισκέψεις -->
        <div class="col-lg-6 mb-4">
            <div class="card shadow h-100">
                <div class="card-header py-3">
                    <h6 class="m-0 font-weight-bold text-success"><i class="fas fa-school mr-2"></i>Σχολικές Επισκέψεις</h6>
                </div>
                <div class="card-body">
                    <p>Οργανώστε την εκπαιδευτική επίσκεψη της τάξης σας στο μουσείο μας. Δηλώστε συμμετοχή εύκολα και γρήγορα μέσα από την πλατφόρμα μας.</p>
                </div>
                <div class="card-footer text-center">
                    <a href="<?= url('/school-tours') ?>" class="btn btn-success">Πληροφορίες Επισκέψεων</a>
                </div>
            </div>
        </div>

    </div>
```

```
    <!-- Call to Action for Guests -->
    <?php if (!is_logged_in()): ?>
    <div class="row justify-content-center mt-4">
        <div class="col-lg-8">
            <div class="card border-left-info shadow py-2">
                <div class="card-body text-center">
                    <h5 class="text-info">Έτοιμοι να ξεκινήσετε;</h5>
                    <p>Δημιουργήστε έναν λογαριασμό για να υποβάλετε τα
αιτήματά σας ή συνδεθείτε αν είστε ήδη μέλος.</p>
                    <a href="<?= url('/login') ?>" class="btn btn-
success mr-2">
                        <i class="fas fa-arrow-right fa-fw mr-
2"></i>Είσοδος
                    </a>
                    <a href="<?= url('/register') ?>" class="btn btn-
info">
                        <i class="fas fa-user-plus fa-fw mr-
2"></i>Εγγραφή
                    </a>
                </div>
            </div>
        </div>
    </div>
    <?php endif; ?>

</div>
<!-- /.container-fluid -->

<?php require_once BASE_PATH . '/views/includes/footer.inc.php'; ?>
```

app/views/stats/index.php

```
<?php require_once BASE_PATH . '/views/includes/header.inc.php'; ?>

<!-- Begin Page Content -->
<div class="container-fluid">

    <h1 class="h3 mb-4 text-gray-800">Στατιστικά & Αναφορές</h1>

    <!--           ΕΝΟΤΗΤΑ VENUES (ΑΙΘΟΥΣΕΣ)            -->
    <?php if ($venueStats): ?>
    <div class="mb-5">
        <h2 class="h4 text-gray-800 mb-3"><i class="fas fa-podcast mr-
2"></i>Αίθουσες Εκδηλώσεων</h2>
        <!-- Summary Cards -->
        <div class="row">
            <div class="col-xl-3 col-md-6 mb-4">
                <div class="card border-left-warning shadow h-100 py-
2">
```

```
                    <div class="card-body">
                        <div class="row no-gutters align-items-center">
                            <div class="col mr-2">
                                <div class="text-xs font-weight-bold
text-warning text-uppercase mb-1">Αιτήματα σε Εκκρεμότητα</div>
                                <div class="h5 mb-0 font-weight-bold
text-gray-800"><?= $venueStats['pending_requests'] ?></div>
                            </div>
                            <div class="col-auto"><i class="fas fa-
comments fa-2x text-gray-300"></i></div>
                        </div>
                    </div>
                </div>
            </div>
        </div>

        <!-- Charts -->
        <div class="row">
            <div class="col-lg-6">
                <div class="card shadow mb-4">
                    <div class="card-header py-3"><h6 class="m-0 font-
weight-bold text-primary">Κρατήσεις ανά Αίθουσα (Top 5)</h6></div>
                    <div class="card-body"><canvas
id="venuePieChart"></canvas></div>
                </div>
            </div>
            <div class="col-lg-6">
                <div class="card shadow mb-4">
                    <div class="card-header py-3"><h6 class="m-0 font-
weight-bold text-primary">Κρατήσεις ανά Μήνα (Τελευταίοι 12)</h6></div>
                    <div class="card-body"><canvas
id="venueBarChart"></canvas></div>
                </div>
            </div>
        </div>
    </div>
    <?php endif; ?>


    <!--       ΕΝΟΤΗΤΑ TOURS (ΣΧ. ΕΠΙΣΚΕΨΕΙΣ)          -->
    <?php if ($tourStats): ?>
    <div class="mb-4">
        <h2 class="h4 text-gray-800 mb-3"><i class="fas fa-school mr-
2"></i>Σχολικές Επισκέψεις</h2>
        <!-- Summary Cards -->
        <div class="row">
            <div class="col-xl-3 col-md-6 mb-4">
                <div class="card border-left-success shadow h-100 py-
2">
```

```
                    <div class="card-body">
                        <div class="row no-gutters align-items-center">
                            <div class="col mr-2">
                                <div class="text-xs font-weight-bold
text-success text-uppercase mb-1">Σύνολο Επιβεβαιωμένων</div>
                                <div class="h5 mb-0 font-weight-bold
text-gray-800"><?= $tourStats['total_bookings'] ?></div>
                            </div>
                            <div class="col-auto"><i class="fas fa-
check-circle fa-2x text-gray-300"></i></div>
                        </div>
                    </div>
                </div>
            </div>
        </div>

        <!-- Charts -->
        <div class="row">
            <div class="col-lg-6">
                <div class="card shadow mb-4">
                    <div class="card-header py-3"><h6 class="m-0 font-
weight-bold text-primary">Κρατήσεις ανά Πρόγραμμα (Top 5)</h6></div>
                    <div class="card-body"><canvas
id="tourPieChart"></canvas></div>
                </div>
            </div>
            <div class="col-lg-6">
                <div class="card shadow mb-4">
                    <div class="card-header py-3"><h6 class="m-0 font-
weight-bold text-primary">Κρατήσεις ανά Μήνα (Τελευταίοι 12)</h6></div>
                    <div class="card-body"><canvas
id="tourBarChart"></canvas></div>
                </div>
            </div>
        </div>
    </div>
    <?php endif; ?>

</div>
<!-- /.container-fluid -->

<?php require_once BASE_PATH . '/views/includes/footer.inc.php'; ?>

<!--            JAVASCRIPT ΓΙΑ ΤΑ CHARTS            -->
<script>
document.addEventListener('DOMContentLoaded', function() {

    // --- VENUE CHARTS ---
```

```
    <?php if ($venueStats): ?>
    // Data from PHP
    const venuePieData = <?= json_encode($venueStats['top_venues']) ?>;
    const venueBarData = <?=
json_encode($venueStats['bookings_by_month']) ?>;

    // Pie Chart
    new Chart(document.getElementById('venuePieChart'), {
        type: 'doughnut',
        data: {
            labels: venuePieData.map(item => item.venueName),
            datasets: [{
                data: venuePieData.map(item => item.count),
                backgroundColor: ['#4e73df', '#1cc88a', '#36b9cc',
'#f6c23e', '#e74a3b'],
            }]
        },
        options: { responsive: true, maintainAspectRatio: false }
    });

    // Line Chart
    new Chart(document.getElementById('venueBarChart'), {
        type: 'line',
        data: {
            labels: venueBarData.map(item => item.month),
            datasets: [{
                label: 'Αριθμός Κρατήσεων',
                data: venueBarData.map(item => item.count),
                borderColor: '#4e73df',
                backgroundColor: 'rgba(78, 115, 223, 0.05)',
                fill: true,
                tension: 0.3
            }]
        },
        options: {
            responsive: true,
            maintainAspectRatio: false,
            scales: {
                yAxes: [{
                    beginAtZero: true,
                    ticks: {
                        stepSize: 1,
                        callback: function(value) {if
(Math.floor(value) === value) {return value;}}
                    }
                }]
            }
        }
```

```
    });


    <?php endif; ?>


    // --- TOUR CHARTS ---
    <?php if ($tourStats): ?>
    // Data from PHP
    const tourPieData = <?= json_encode($tourStats['top_tours']) ?>;
    const tourBarData = <?=
json_encode($tourStats['bookings_by_month']) ?>;

    // Pie Chart
    new Chart(document.getElementById('tourPieChart'), {
        type: 'doughnut',
        data: {
            labels: tourPieData.map(item => item.tourName),
            datasets: [{
                data: tourPieData.map(item => item.count),
                backgroundColor: ['#4e73df', '#1cc88a', '#36b9cc',
'#f6c23e', '#e74a3b'],
            }]
        },
        options: { responsive: true, maintainAspectRatio: false }
    });


    // Line Chart
    new Chart(document.getElementById('tourBarChart'), {
        type: 'line',
        data: {
            labels: tourBarData.map(item => item.month),
            datasets: [{
                label: 'Αριθμός Κρατήσεων',
                data: tourBarData.map(item => item.count),
                borderColor: '#1cc88a',
                backgroundColor: 'rgba(28, 200, 138, 0.05)',
                fill: true,
                tension: 0.3
            }]
        },
        options: {
            responsive: true,
            maintainAspectRatio: false,
            scales: {
                yAxes: [{
                    beginAtZero: true,
                    ticks: {
```

```
                        stepSize: 1,
                        callback: function(value) {if
(Math.floor(value) === value) {return value;}}
                    }
                }]
            }
        }
    });


    <?php endif; ?>

});
</script>
```

ΠΑΡΑΡΤΗΜΑ Δ: Οδηγίες Εγκατάστασης και Εκτέλεσης

Το παρόν παράρτημα παρέχει αναλυτικές οδηγίες για την εγκατάσταση και την εκτέλεση της εφαρμογής «MuseApp» σε τοπικό περιβάλλον. Η διαδικασία έχει απλοποιηθεί χάρη στη χρήση της τεχνολογίας Docker, η οποία εξαλείφει την ανάγκη χειροκίνητης εγκατάστασης διακομιστών (Apache, MySQL, PHP).

Δ.1 Προαπαιτούμενα Συστήματος

Πριν ξεκινήσετε την εγκατάσταση, βεβαιωθείτε ότι στον υπολογιστή σας είναι εγκατεστημένα τα ακόλουθα λογισμικά:

- Docker Desktop (για Windows/Mac) ή Docker Engine & Docker Compose (για Linux).
- Ένας σύγχρονος φυλλομετρητής ιστοσελίδων (Google Chrome, Mozilla Firefox).

Δ.2 Λήψη και Προετοιμασία Αρχείων

- *Λήψη Κώδικα:* Αντιγράψτε τον φάκελο του έργου (MUSEAPP) στον τοπικό σας δίσκο. Η δομή του φακέλου πρέπει να είναι η εξής:

```
MUSEAPP/
├── mysql/
│   └── schema-data.sql
├── php/
│   ├── app/
│   ├── public/
│   ├── Dockerfile
│   └── apache.conf
├── .env
└── compose.yml
```

- Ρύθμιση Περιβάλλοντος (`.env`): Σιγουρευτείτε ότι υπάρχει το αρχείο .env στον ριζικό φάκελο. Αυτό το αρχείο περιέχει τις ευαίσθητες παραμέτρους σύνδεσης. Ένα τυπικό παράδειγμα περιεχομένου είναι:

```
MYSQL_ROOT_PASSWORD=rootpass
MYSQL_DATABASE=musedb
```

```
MYSQL_USER=appuser
MYSQL_PASSWORD=apppass
```

Δ.3 Εκκίνηση της Εφαρμογής

1. Ανοίξτε ένα τερματικό (Command Prompt, PowerShell ή Terminal).
2. Πλοηγηθείτε στον ριζικό φάκελο της εφαρμογής:

```
cd /path/to/MUSEAPP
```

3. Εκτελέστε την εντολή για το χτίσιμο (build) και την εκκίνηση των containers:

```
docker-compose up -d –build
```

- Η παράμετρος `-d` (detached) τρέχει τα containers στο παρασκήνιο.
- Η παράμετρος `--build` διασφαλίζει ότι θα ξαναχτιστεί η εικόνα της PHP αν έχουν γίνει αλλαγές στο Dockerfile.

Κατά την πρώτη εκτέλεση, το Docker θα κατεβάσει τις απαραίτητες εικόνες (images) για την MySQL και την PHP και θα δημιουργήσει τη βάση δεδομένων, εισάγοντας αυτόματα την βάση και τα αρχικά δεδομένα από το αρχείο `mysql/schema-data.sql`.

Δ.4 Πρόσβαση στην Εφαρμογή

Μόλις ολοκληρωθεί η εκκίνηση των containers, ανοίξτε τον φυλλομετρητή σας και πληκτρολογήστε τη διεύθυνση:

```
http://localhost:8080
```

Θα πρέπει να δείτε την αρχική σελίδα (Landing Page) της εφαρμογής MuseApp.

Δ.5 Στοιχεία Εισόδου

Για την είσοδο στο σύστημα ως Διαχειριστής (Administrator), χρησιμοποιήστε τα παρακάτω προκαθορισμένα στοιχεία:

```
username: admin
```

```
password: Museadmin1!
```

Σημείωση: Συνιστάται η αλλαγή του κωδικού πρόσβασης αμέσως μετά την πρώτη είσοδο μέσω της σελίδας «Επεξεργασία Προφίλ».

Δ.6 Τερματισμός Εφαρμογής

Για να σταματήσετε την εφαρμογή και να αποδεσμεύσετε τους πόρους του συστήματος, εκτελέστε στο τερματικό (μέσα στον φάκελο της εφαρμογής):

```
docker-compose down
```

Αν θέλετε να διαγράψετε και τα αποθηκευμένα δεδομένα της βάσης (volumes), προσθέστε την παράμετρο -v:

```
docker-compose down -v
```