ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΛΟΠΟΝΝΗΣΟΥ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΑΣ ΚΑΙ ΟΙΚΟΝΟΜΙΑΣ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ
Π.Μ.Σ. ΣΤΗΝ ΕΠΙΣΤΗΜΗ ΤΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Σχεδιασμός και υλοποίηση διαδικτυακού περιβάλλοντος προσομοίωσης κίνησης επισκεπτών σε μουσεία εσωτερικού χώρου

ΚΟΥΤΡΟΥΜΑΝΗΣ ΝΙΚΟΛΑΟΣ-ΣΠΥΡΙΔΩΝ
Α.Μ. 2022201902008

ΛΑΛΟΥΔΑΚΗΣ ΚΩΝΣΤΑΝΤΙΝΟΣ
Α.Μ. 2022201902011

Επιβλέπων:
Νικόλαος Τσελίκας

ΤΡΙΠΟΛΗ
Οκτώβριος 2020

Ευχαριστίες

Με την ολοκλήρωση αυτής της διπλωματικής εργασίας, στο πλαίσιο του μεταπτυχιακού προγράμματος σπουδών που παρακολουθήσαμε στο Πανεπιστήμιο Πελοποννήσου, στο τμήμα Πληροφορικής και Τηλεπικοινωνιών (Π.Μ.Σ. Επιστήμης Υπολογιστών), θα θέλαμε να ευχαριστήσουμε όλους όσοι βοήθησαν για να φτάσουμε στην ολοκλήρωση αυτού του κύκλου. Θα θέλαμε να ευχαριστήσουμε τον επιβλέποντα καθηγητή κο Νικόλαο Τσελίκα για την εμπιστοσύνη, που μας έδειξε αναθέτοντάς μας την παρούσα εργασία, καθώς και για την άμεση και πολύτιμη καθοδήγησή του, τόσο στην εκπόνηση της εργασίας όσο και καθ' όλη τη διάρκεια των μεταπτυχιακών μας σπουδών.

Σαν Νίκος θα ήθελα να ευχαριστήσω την οικογένειά μου που ήταν πάντα στο πλευρό μου τόσο στα εύκολα όσο και στα δύσκολα. Πιο συγκεκριμένα όμως, θα ήθελα να αφιερώσω την παρούσα εργασία στην αδερφή μου, την Ελευθερία, που μου έδειξε πως είναι να μάχεσαι για τους στόχους σου.

Σαν Κώστας θα ήθελα και εγώ με τη σειρά μου να ευχαριστήσω την οικογένειά μου για όλη τη στήριξή της όλα αυτά τα χρόνια των σπουδών μου.

Τέλος δεν θα μπορούσαμε να ξεχάσουμε τους φίλους μας Θάνο Βακ, Θανάση, Θάνο και Ηλία που έκαναν κατάληψη επί έξι χρόνια στο ΣΠΙΤΙ ΜΑΣ. Χωρίς αυτούς, αυτά τα έξι χρόνια δε θα περνούσαν το ίδιο ευχάριστα.

Περίληψη

Στις μέρες μας, η τεχνολογία έχει συμβάλλει τα μέγιστα στην ανάπτυξη διάφορων τομέων. Παρ' όλα αυτά, αρκετά μουσεία, επιλέγουν να μη χρησιμοποιούν τις νέες τεχνολογίες, με αποτέλεσμα να μένουν στάσιμα και να χάνουν μέρος της δημοτικότητάς τους. Μέσω αυτής της εφαρμογής προτείνουμε έναν νέο τρόπο οργάνωσης των μουσείων. Πιο συγκεκριμένα, προτείνουμε την εγκατάσταση access points σε κάθε έκθεμα, τα οποία σε συνάρτηση με την εφαρμογή μας, θα παρέχουν σημαντικά στατιστικά στοιχεία.

Αντικείμενο μελέτης της εργασίας αποτελεί η δημιουργία διαδικτυακής εφαρμογής με χρήση διαφόρων τεχνολογιών διαδικτύου. Αφενός εξετάζονται όλες οι σύγχρονες τεχνολογίες και αφετέρου δοκιμάζονται στην πράξη για την κατασκευή της εφαρμογής.

Η εφαρμογή υλοποιείται με χρήση των HTML5, CSS, Javascript, PHP και MySQL. Είναι σχεδιασμένη να καλύψει μία κατηγορία χρηστών, τους διαχειριστές μουσείων.

Κάθε διαχειριστής μουσείου έχει τη δυνατότητα δημιουργίας ενός λογαριασμού στην εφαρμογή μας, συμπληρώνοντας μερικά στοιχεία. Με αυτά, μπορεί να εισέλθει στο σύστημα και να εκτελέσει όλες τις δημιουργημένες από εμάς λειτουργίες. Συγκεκριμένα, υπάρχει η δυνατότητα δημιουργίας ενός αντιγράφου της κάτοψης του μουσείου του και αποθήκευσής του στο σύστημα. Μόλις το αποθηκεύσει θα μπορεί να το επαναφορτώσει για τυχόν διορθώσεις και αλλαγές, καθώς και να προσομοιώσει την κίνηση διαφόρων γκρουπ επισκεπτών σε αυτό. Μετά τη προσομοίωση, έχει την επιλογή να εμφανίσει στο μουσείο του ένα θερμογράφημα (heatmap) κίνησης, το οποίο εμφανίζει την επισκεψιμότητα κάθε σημείου του μουσείου κατά τη διάρκεια της προσομοίωσης. Μέσω αυτού, μπορεί να παρατηρήσει διάφορα στοιχεία για το μουσείο του, όπως επισκεψιμότητα εκθεμάτων, πιθανά σημεία συνωστισμού ή και απουσίας κόσμου.

Στο παρόν κείμενο γίνεται εκτενής αναφορά στη θεωρία που χρησιμοποιήθηκε, αλλά εμφανίζονται και πολλά πρακτικά παραδείγματα. Τέλος, η λογική ανάπτυξης της συγκεκριμένης εφαρμογής μπορεί να λειτουργήσει σαν αφετηρία στην περαιτέρω, εάν όχι καθολική επέκταση των παροχών ενός μουσείου προς το κοινό του, απολαμβάνοντας αντίστοιχα οφέλη.

Ο κώδικας, μέσω του οποίου υλοποιήθηκε η εφαρμογή μας, βρίσκεται στο GitHub. Μπορείτε να το μελετήσετε, ακολουθώντας το qr code, που βρίσκεται παρακάτω.

Abstract

Nowadays, technology has contributed greatly to the development of various fields. Nevertheless, many museums still insist not to use the new technologies, and as a result, they remain stagnant and lose part of their popularity. Through this application, we propose a new way of organising museums. More specifically, we propose the installation of access points at each exhibit, which, in combination with our application, will provide important statistics.

The objective of our work is the development of a web application using various Internet technologies. On one hand, all modern technologies are examined, while on the other, they are tested in practice during the development life cycle of our application.

The application is implemented using HTML5, CSS, Javascript, PHP and MySQL. It is designed to cover a category of users, and specifically museum administrators.

Every museum administrator has the ability to create an account in our application by filling in some trivial data. Using these, he/she can log into the system and perform all the provided functionality. Specifically, there is the possibility of creating a copy of his/her museum layout and saving it in the system. Once saved it can be reloaded for any corrections and changes, as well as for the simulation of the movement of various groups of visitors to it. After performing each simulation, the user has the option of displaying a motion heatmap in the museum layout, which displays the traffic of each point of the museum during the simulation. Through it, the user can observe various intresting parameters regarding the museum, such as popular exhibits, possible crowded areas or even the absence of people.

The present text makes extensive reference to the theory used, but also features many practical examples. Finally, the logic of developing this application can serve as a starting point for the further, if not the universal, extension of a museum's facilities to its public, enjoying corresponding benefits.

The implemented code of our application is available on GitHub. You can study it by following the qr code below.

***Keywords**: Javascript, JQuery, SVG, HTML5, Pathfinding.js, Heatmap.js, PHP, Bootstrap, XAMPP, Web Application*

Περιεχόμενα

Ευρετήριο εικόνων

Πίνακας πηγαίου κώδικα

Κεφάλαιο 1: Εισαγωγή

1.1 Σκοπός- Παρουσίαση του προβλήματος

Σκοπός αυτής της εργασίας ήταν η ανάπτυξη μιας διαδικτυακής εφαρμογής, που θα απευθύνεται σε υπεύθυνους μουσείων. Πριν εξηγήσουμε πώς λειτουργεί η εφαρμογή μας, ας επικεντρωθούμε στην αρχική ιδέα, που μας ώθησε να τη δημιουργήσουμε. Ας υποθέσουμε ότι υπάρχει ένα μουσείο με διάφορα εκθέματα. Οι υπεύθυνοι του μουσείου επιλέγουν να τοποθετήσουν σε κάθε έκθεμα ένα φορητό σημείο πρόσβασης wifi (access point). Κάθε επισκέπτης του μουσείου έχει μαζί του ένα smartphone, με δυνατότητα πρόσβασης στο διαδίκτυο. Αυτό σημαίνει ότι όση ώρα βρίσκεται στο μουσείο έχει ενεργοποιημένο το wifi του κινητού του και μπορεί να συνδέεται με όποιο access point βρίσκεται πιο κοντά. Με αυτό τον τρόπο, αφού συλλέξουμε τα κατάλληλα δεδομένα από κάθε access point, μπορούμε να καταλάβουμε περίπου τη διαδρομή του μέσα στο μουσείο.

Αφού εξηγήσαμε την αρχική ιδέα, ας αναφερθούμε τώρα στο πώς σχετίζεται η εφαρμογή μας με αυτό. Οι χρήστες θα είναι αποκλειστικά μία κατηγορία ατόμων (διαχειριστές μουσείων), οι οποίοι θα μπορούν να κατασκευάσουν ένα αντίγραφο της κάτοψης του μουσείου τους και να το αποθηκεύσουν. Στη συνέχεια, χρησιμοποιώντας τα δεδομένα, που έχουν συλλέξει από τα access points, θα μπορούν να πραγματοποιήσουν μία προσομοίωση της κίνησης των επισκεπτών. Τέλος, αφού έχουν εισάγει όλα τα δεδομένα, η εφαρμογή θα τους δίνει τη δυνατότητα δημιουργίας ενός heatmap κίνησης, μέσω του οποίου θα μπορούν να παρατηρήσουν την επισκεψιμότητα κάθε εκθέματος του μουσείου, καθώς και αν υπάρχουν κάποια σημεία που δημιουργείται συνωστισμός ή κάποια άλλα, που δεν ενδιαφέρουν το κοινό. Οπότε, μέσω αυτής της λειτουργίας, θα μπορεί κάθε διαχειριστής να παρατηρήσει αν υπάρχουν προβλήματα με τη διάταξη του μουσείου του, ώστε να μπορέσει να τα διορθώσει.

1.2 Δομή της εργασίας

Η δομή της εργασίας είναι δομημένη κλιμακωτά. Αρχικά, στο Κεφάλαιο 2 δίνονται κάποιες βασικές πληροφορίες για τις τεχνολογίες, που χρησιμοποιήθηκαν. Συγκεκριμένα, εξηγούμε κάποια στοιχεία, ώστε να μπορέσει κάποιος να κατανοήσει πώς δημιουργήθηκε και λειτουργεί η εφαρμογή. Στο Κεφάλαιο 3, αναφερόμαστε στη σχεδίαση της εφαρμογής, όπως την είχαμε σκεφτεί πριν ξεκινήσει η υλοποίησή της. Επίσης, γίνεται λεπτομερής αναφορά σε όλες τις λειτουργίες που θα περιέχει η εφαρμογή μας. Αμέσως μετά, στο κεφάλαιο 4 παρουσιάζεται αναλυτικά η υλοποίηση της βάσης δεδομένων της εφαρμογής μας, καθώς και των κυριότερων λειτουργιών της. Συγκεκριμένα, για κάθε σημαντική λειτουργία της εφαρμογής εξηγούμε τον τρόπο, με τον οποίο υλοποιήθηκε, δείχνοντας παράλληλα και τα ανάλογα κομμάτια κώδικα. Στο αμέσως επόμενο κεφάλαιο (κεφάλαιο 5) περιγράφονται πιθανές βελτιώσεις και επεκτάσεις, που μπορεί να λάβει η εφαρμογή στο μέλλον. Επίσης, αναγράφονται τα συμπεράσματα που προέκυψαν από την ανάπτυξη της εφαρμογής. Τέλος, στο κεφάλαιο 6, παρουσιάζεται ένας εκτενής οδηγός ορθής χρήσης της εφαρμογής, καθώς και το δέντρο αρχείων αυτής, επεξηγώντας κάθε αρχείο ξεχωριστά.

Κεφάλαιο 2: Τεχνολογίες που χρησιμοποιήσαμε

2.1 JavaScript

Η JavaScript είναι μία γλώσσα προγραμματισμού υπολογιστών, για σεναριοποίηση των διαδραστικών αποτελεσμάτων εντός των περιηγητών ιστού. Υποστηρίζεται από όλους τους δημοφιλείς περιηγητές, όπως οι Firefox, Safari, Opera, Google Chrome, κ.τ.λ.

Αποτελεί μία από τις πιο ευέλικτες και αποτελεσματικές γλώσσες που μπορεί να χρησιμοποιηθούν από προγραμματιστές. Σύμφωνα με μία έρευνα, η JavaScript χρησιμοποιείται από το 88% όλων των ιστοτόπων. Θα βρείτε τη JavaScript όχι μόνο σε κάθε ιστότοπο, αλλά επίσης και σε ιστότοπους κινητών, παιχνιδιών και εφαρμογών ιστού.

Ας ρίξουμε μια ματιά στα πλεονεκτήματα της JavaScript, τα οποία την καθιστούν τόσο δημοφιλή ανάμεσα στους προγραμματιστές:

- **Επεξεργασία από την Πλευρά του Πελάτη**

Αυτό σημαίνει ότι ο κώδικας εκτελείται από τον επεξεργαστή του χρήστη, αντί του εξυπηρετητή ιστού, εξοικονομώντας έτσι εύρος ζώνης και περιορίζοντας την υπερφόρτωση του εξυπηρετητή.

- **Η εκμάθησή της είναι απλή**

Η σύνταξη αυτής της γλώσσας είναι παρόμοια με τα απλά Αγγλικά, καθιστώντας την εκμάθησή της ευκολότερη για τους προγραμματιστές.

- **Εκτεταμένη Λειτουργικότητα για Ιστοσελίδες**

Οι προσθήκες τρίτων βοηθούν τους προγραμματιστές JavaScript να γράψουν αποσπάσματα κώδικα, τα οποία μπορεί να χρησιμοποιηθούν στις ιστοσελίδες, όπου χρειάζεται.

- **Η Υλοποίησή της είναι Απλή**

Η δυνατότητα χρήσης της ίδιας γλώσσας στην κεντρική σελίδα που βλέπει ο χρήστης και το διαχειριστικό τμήμα, καθιστά την εργασία των ομάδων προγραμματισμού ευκολότερη.

- **Οικονομική Γλώσσα**

Δεν απαιτεί κανέναν ειδικό μεταγλωττιστή ή συντάκτη. Το μόνο που χρειάζεστε είναι ένας προγραμματιστής, ένα πρόγραμμα επεξεργασίας κειμένου και έναν περιηγητή για να “τρέξει” τον κώδικα JavaScript.

- **Σχετικά γρήγορη για τον τελικό χρήστη**

Δεν χρειάζεται, πλέον, οι επισκέπτες να συμπληρώσουν μία ολόκληρη φόρμα και να την υποβάλλουν, για να μάθουν πως υπάρχει κάποιο τυπογραφικό λάθος στο πρώτο πεδίο και ότι θα πρέπει να συμπληρώσουν ολόκληρη τη φόρμα ξανά. Με τη JavaScript, κάθε πεδίο μπορεί να επαληθεύεται καθώς συμπληρώνεται από τους χρήστες, γεγονός που παρέχει άμεση ανατροφοδότηση, όταν αυτοί κάνουν κάποιο λάθος.

- **Περιηγητές με ενσωματωμένη JavaScript**

Οι χρήστες του ιστότοπου δεν χρειάζονται ειδικό λογισμικό και λήψεις προγραμμάτων για να δουν τη JavaScript, έτσι κάθε χρήστης έχει την ίδια εμπειρία.

Ο κώδικας Javascript μιας σελίδας περικλείεται από τις ετικέτες της HTML **<script type="text/javascript"> και </script>.**

Για παράδειγμα, ο ακόλουθος κώδικας Javascript εμφανίζει ένα πλαίσιο διαλόγου με το κείμενο "Γεια σου, κόσμε!":

```
1. <script type="text/javascript">
2. alert('Γεια σου, κόσμε!');
3. </script>
```

Αν ο κώδικας Javascript περιέχει περισσότερες από μία εντολές, αυτές θα πρέπει να διαχωριστούν μεταξύ τους με το χαρακτήρα του ελληνικού ερωτηματικού ';' (δηλαδή της λατινικής άνω τελείας). Η χρήση του χαρακτήρα αυτού για την τελευταία εντολή δεν είναι απαραίτητη. Η διαχώριση των εντολών στους νεότερους φυλλομετρητές (browsers) δεν είναι απαραίτητη.

Μια άλλη βασική εντολή, η window.prompt ("μήνυμα προς το χρήστη"), ζητάει από το χρήστη να συμπληρώσει ένα κομμάτι μιας αίτησης απευθείας ώστε τα δεδομένα να χρησιμοποιηθούν σαν κείμενο:

```
1. <script>
2. var FIRSTvariable = window.prompt("PLEASE FILL IN YOUR NAME")
3. alert("Your name is " + FIRSTvariable + ".")
4. </script>
```

2.2 HTML

Η HTML είναι ένα σύνολο κανόνων για την διαμόρφωση της εμφάνισης και του περιεχομένου μιας ιστοσελίδας. Ουσιαστικά, δεν είναι γλώσσα προγραμματισμού, αλλά γλώσσα περιγραφής ιδιοτήτων των στοιχείων που αποτελούν μία ιστοσελίδα. Χρησιμοποιώντας ειδικές ετικέτες (tags) δίνονται εντολές, για το πώς να διαχειριστεί ένα έγγραφο html το περιεχόμενό του και πώς να το εμφανίσει στον χρήστη ένας web browser, το πρόγραμμα δηλαδή που χρησιμοποιούμε για να προβάλουμε τις διάφορες σελίδες στο διαδίκτυο.

Ενδεικτικά, οι εντολές / tags της HTML, μπορούν:

1. Να εισάγουν σε μία ιστοσελίδα links (συνδέσμους)
2. Να εισάγουν σε μία ιστοσελίδα εικόνες
3. Να διαμορφώσουν το κείμενο με έντονα ή πλάγια γράμματα κλπ.

Για να δημιουργηθεί ένα αρχείο html, αρκεί ένα αρχείο απλού κειμένου, το οποίο θα έχει κατάληξη **.html** ή .**htm** και το αρχείο αυτό να περιέχει τις επιθυμητές εντολές με τις ανάλογες παραμέτρους τους.

Η HTML μπορεί να γραφτεί απευθείας ως κώδικας (πηγαίος κώδικας) ή μπορεί να παραχθεί αυτόματα από κάποιο πρόγραμμα κατασκευής ιστοσελίδων (υπάρχουν πολλά διαθέσιμα open source προγράμματα που μπορεί να κατεβάσει κανείς και να εγκαταστήσει στο μηχάνημά του) την

στιγμή που ο δημιουργός, σε γραφικό περιβάλλον, απλά χρησιμοποιεί τα διάφορα εργαλεία που του παρέχει το πρόγραμμα για το σχεδιασμό της σελίδας του.

Για την σωστή συγγραφή της HTML υπάρχουν web standards (https://www.w3.org/), τα οποία άλλοι τα λαμβάνουν υπόψη τους και άλλοι όχι. Ένας εύκολος τρόπος για τον έλεγχο των web standards σε μία ιστοσελίδα, είναι μέσω του εξής validator: https://validator.w3.org/

Η σήμανση HTML αποτελείται από μερικά βασικά συστατικά, συμπεριλαμβανομένων των στοιχείων (και των ιδιοτήτων τους), τους βασισμένους σε χαρακτήρες τύπους δεδομένων, τις αναφορές χαρακτήρων και τις αναφορές οντοτήτων. Ένα ξεχωριστό σημαντικό συστατικό είναι η δήλωση τύπου εγγράφου (document type declaration), η οποία ορίζει στο πρόγραμμα περιήγησης (browser) τον τρόπο εμφάνισης της σελίδας.

Στην HTML, το πρόγραμμα Hello world, ένα συνηθισμένο πρόγραμμα υπολογιστή που χρησιμεύει για τη σύγκριση γλωσσών προγραμματισμού, γλωσσών σεναρίων και γλωσσών σήμανσης, φτιάχνεται με 9 γραμμές κώδικα, παρότι οι νέες γραμμές είναι προαιρετικές στην HTML:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Hello HTML</title>
  </head>
  <body>
    <p>Hello world</p>
  </body>
</html>
```

2.2.1 HTML5

Η HTML5 είναι το νέο standard πρότυπο για την HTML, την XHTML και την HTML DOM. Η ανάπτυξη της HTML5 έγινε με την συνεργασία της World Wide Web Consortium (W3C) και της Web Hypertext Application Technology Working Group (WHATWG). Η WHATWG εργαζόταν επάνω στις web φόρμες και τις web εφαρμογές, ενώ η W3C, η οποία δημιούργησε και διαχειρίζεται τα πρότυπα της HTML και της XHTML, ασχολήθηκε με την ανάπτυξη του νέου προτύπου XHTML 2.0. Το 2006 αποφάσισαν να συνεργαστούν για να δημιουργήσουν το νέο πρότυπο, την HTML5.

Σε αυτήν την έκδοση δίνεται μεγάλη έμφαση στη δημιουργία responsive ιστοσελίδων ,σελίδων που προσαμόζονται σε διαφορετικές συσκευές. Για τους χρήστες κινητών συσκευών (smartphones), που πιθανόν πλοηγούνται στην ιστοσελίδα με περιορισμένο bandwidth και είναι αυτοί στους οποίους απευθύνεται κυρίως η τεχνική της responsive σχεδίασης, θα πρέπει η ιστοσελίδα όχι μόνο να ανταποκρίνεται στον περιορισμένο «χώρο» της οθόνης τους, αλλά και να «φορτώνεται» όσο το δυνατόν ταχύτερα. Επίσης, προσφέρει πληθώρα απο νέες δυνατότητες που απλοποιούν και διευκολύνουν στην ανάπτυξη ενός ιστοτόπου. Προστίθενται νέες ετικέτες (tags) που προσφέρουν πιο ουσιαστικό κώδικα στις μηχανές αναζήτησης και νέες καινοτομίες που απλοποιούν την υλοποίηση διαφόρων εργασιών αναιρώντας τη χρήση κώδικα JavaScript και plug-ins όπως Adobe Flash κτλ. Σκοπός είναι η δημιουργία περισσότερο λιτού απλοποιημένου, και γρήγορου στη φόρτωση, κώδικα.

2.3 CSS

CSS σημαίνει Cascading Style Sheets και είναι στυλ που μπορούμε να ορίσουμε για τις HTML σελίδες.

Γράφοντας τις σελίδες μας μόνο με HTML κώδικα, μπορούμε να ορίσουμε το χρώμα και το μέγεθος του κειμένου αλλά και άλλων στοιχείων της σελίδας (όπως πίνακες, links, λίστες κτλ). Για να αλλάξουμε το χρώμα κάποιου κειμένου ή το χρώμα ενός πίνακα, θα πρέπει να βρούμε το χρώμα αυτό μέσα στον κώδικα και να το αλλάξουμε. Η διαδικασία αυτή μπορεί να φαντάζει εύκολη όταν έχουμε να διαχειριστούμε μια μόνο σελίδα, αλλά ένα site αποτελείται από δεκάδες σελίδες τις οποίες χρειάζεται να διαχειριζόμαστε εύκολα και γρήγορα.

Φανταστείτε, για παράδειγμα, πόσο χρονοβόρο θα είναι αν θελήσουμε κάποια στιγμή να αλλάξουμε τα χρώματα στο κύριο μενού του site μας, το οποίο επαναλαμβάνεται σε όλες τις σελίδες. Σε μια τέτοια περίπτωση θα χρειαζόταν να ανοίγουμε κάθε σελίδα του site και να αλλάζουμε τα χρώματα του φόντου και των links του μενού, διαδικασία που, εκτός από χρονοβόρα, είναι και κουραστική.

Με τη χρήση CSS μπορούμε να ορίζουμε χρώματα και μεγέθη οργανωμένα σε στυλ και έπειτα να εφαρμόζουμε τα στυλ αυτά στα στοιχεία των σελίδων του site μας. Με αυτόν τον τρόπο, κάθε φορά που αλλάζουμε το χρώμα ενός στυλ, αλλάζει το χρώμα όλων των στοιχείων που έχουν αναφορά στο στυλ αυτό. Έτσι, αν έχουμε ορίσει ένα στυλ για το κύριο μενού του site, τότε θα χρειάζεται να αλλάξουμε το χρώμα του στυλ αυτού και αυτόματα θα εφαρμοστεί σε όλες τις σελίδες.

Εκτός από την ευκολία στη διαχείριση ενός site, ένα άλλο σημαντικό πλεονέκτημα της χρήσης CSS στις σελίδες είναι ο "καθαρότερος" κώδικας, χωρίς πολλές ιδιότητες στις ετικέτες, οι οποίες τον κάνουν δυσανάγνωστο. Επιπλέον, κάνει γρηγορότερη την πλοήγηση καθώς το αρχείο, μέσα στο οποίο ορίζονται τα στυλ, "διαβάζεται" από τον browser μόνο μια φορά και έπειτα αποθηκεύεται στην cache memory, μειώνοντας έτσι το μέγεθος της πληροφορίας που γίνεται download από τους browsers.

2.4 PHP

Η γλώσσα PHP (είναι ανοικτό-ελεύθερο λογισμικό) μπορεί να εγκατασταθεί σχεδόν σε όλα τα λειτουργικά συστήματα όπως Windows, Linux, Mac OS X, Risc OS κλπ αλλά και υποστηρίζεται και από τα περισσότερους εξυπηρετητές ιστοσελίδων όπως ο Apache ή ο IIS. Η PHP μπορεί να λειτουργήσει είτε ως εγκατεστημένη μονάδα (module) στον εξυπηρετητή ιστοσελίδων είτε μέσω ενός επεξεργαστή CGI σεναρίων. Η PHP μπορεί να χρησιμοποιηθεί για εκτέλεση σεναρίων (scripts) από την πλευρά του απομακρυσμένου εξυπηρετητή ιστοσελίδων όπως γίνεται και με τα σενάρια CGI. Επίσης η php μπορεί να χρησιμοποιηθεί για είσοδο/έξοδο δεδομένων από τον χρήστη ή για τη δυναμική δημιουργία σελίδων.

Παρακάτω μπορούμε να δούμε ένα απλό παράδειγμα:

```
1.  <html>
2.   <head>
3.   <title>Example</title>
4.   </head>
```

```
5.   <body>
6.   <?php
7.   echo "Hi, I'm a PHP script!";
8.   ?>
9.   </body>
10. </html>
```

Όπως βλέπουμε, αυτό το script είναι διαφορετικό από διάφορα άλλα σε γλώσσες όπως C ή Perl. Στην προκειμένη περίπτωση, αντί να γράφουμε ένα πρόγραμμα με πολλές εντολές για να εξάγουμε HTML, γράφουμε ένα HTML script με κάποιο ενσωματωμένο κώδικα για να εκτελέσει κάτι (σε αυτή την περίπτωση, να εμφανίζει κάποιο κείμενο). Ο κώδικας PHP είναι εσώκλειστος σε ειδικά tags (ετικέτες) αρχής και τέλους.

Η PHP εργάζεται μ' έναν τελείως διαφορετικό τρόπο. Μια ιστοσελίδα που περιέχει κάποιον κώδικα σε PHP υφίσταται προεπεξεργασία από τη μηχανή της PHP, που αποκαλείται διερμηνευτής (interpreter), και τα αποτελέσματα αυτής της επεξεργασίας στέλνονται πίσω στον web server και από εκεί στον φυλλομετρητή του χρήστη της ιστοσελίδας. Ο κώδικας PHP μπορεί να θέσει ερωτήματα σε βάσεις δεδομένων, να δημιουργήσει εικόνες, να διαβάσει και να γράψει αρχεία, να συνδεθεί με απομακρυσμένους υπολογιστές, κ.ο.κ. Σε γενικές γραμμές οι δυνατότητες που μας δίνει είναι απεριόριστες. Τα αποτελέσματα της επεξεργασίας του PHP κώδικα είναι τα μόνα που στέλνονται στον φυλλομετρητή, ο κώδικας που τα δημιούργησε παραμένει κρυφός και συνεπώς πολύ πιο ασφαλής. Αυτό το είδος της προεπεξεργασίας αποκαλείται server-side scripting και ενώ δεν παρέχει το ίδιο είδος δυναμικών εφέ όπως η JavaScript, οι PHP σελίδες αποκαλούνται δυναμικές.

2.5 MySQL

Η MySQL είναι ένα σύστημα διαχείρισης σχεσιακών βάσεων ανοικτού κώδικα (relational database management system – RDBMS), που χρησιμοποιεί την Structured Query Language (SQL), την πιο γνωστή γλώσσα για την προσθήκη, την πρόσβαση και την επεξεργασία δεδομένων σε μία Βάση Δεδομένων.

Επειδή είναι ανοικτού κώδικα (open source), οποιοσδήποτε μπορεί να κατεβάσει τη MySQL και να την διαμορφώσει με βάση τις ανάγκες του, σύμφωνα πάντα με την γενική άδεια χρήσης.

Η MySQL είναι γνωστή κυρίως για την ταχύτητα, την αξιοπιστία, και την ευελιξία που παρέχει. Οι περισσότεροι συμφωνούν ωστόσο ότι δουλεύει καλύτερα όταν διαχειρίζεται περιεχόμενο και όχι όταν εκτελεί συναλλαγές. Η MySQL αυτή τη στιγμή μπορεί να λειτουργήσει σε περιβάλλον Linux, Unix, και Windows.

Ένας MySQL server αποτελείται από ένα σύνολο βάσεων δεδομένων (databases). Κάθε βάση δεδομένων είναι ένα σύνολο πινάκων (tables) παρόμοιων με αυτών που μπορείτε να φτιάξετε σε ένα πρόγραμμα spreadsheet όπως το Excel. Οι γραμμές του πίνακα ονομάζονται εγγραφές (records) και οι στήλες του πίνακα ονομάζονται πεδία (fields). Δεν υπάρχει ουσιαστικό όριο στην ποσότητα βάσεων που περιέχονται στον server, στην ποσότητα πινάκων που περιέχονται στις βάσεις μα ούτε και στην ποσότητα πεδίων και εγγραφών που περιέχονται στους πίνακες. Τα πεδία των πινάκων μπορούν να περιέχουν μόνο ένα συγκεκριμένο τύπο πληροφορίας ο οποίος ορίζεται κατά τη δημιουργία του πίνακα. Για παράδειγμα, ένα πεδίο που είναι ορισμένο ως ακέραιος αριθμός δεν είναι κατάλληλο για αποθήκευση δεκαδικού αριθμού ή κειμένου. Η δομή αυτή χρησιμοποιείται από όλα τα σύγχρονα συστήματα βάσεων δεδομένων και έχει αποδειχτεί πολύ πρακτική για την αποθήκευση τεράστιων όγκων δομημένης πληροφορίας. Σε κάθε πίνακα βάσης δεδομένων είναι

σημαντικό να υπάρχει μια στήλη (πεδίο) σε κάθε σειρά της οποίας να υπάρχει διαφορετική τιμή έτσι ώστε κάθε εγγραφή να αποκτήσει ένα μοναδικό χαρακτηριστικό, έναν «αριθμό ταυτότητας» με τον οποίο να μπορούμε να αναφερόμαστε σε αυτήν. Το ιδιαίτερο αυτό πεδίο το ορίζουμε κατά την κατασκευή του πίνακα δίνοντάς του την ιδιότητα του πρωτεύοντος κλειδιού (primary key)

2.6 AJAX

Το AJAX (Asynchronous JavaScript and XML) είναι ένα σύνολο τεχνικών ανάπτυξης ασύγχρονων εφαρμογών στο Διαδίκτυο που χρησιμοποιεί πολλές τεχνολογίες μαζί από την πλευρά του προγράμματος πελάτη (client). Με το AJAX οι εφαρμογές ιστού μπορούν να στέλνουν και να λαμβάνουν δεδομένα στο παρασκήνιο χωρίς να τους απασχολεί τι προβάλλεται στην σελίδα. Ο διαχωρισμός της ανταλλαγής δεδομένων με την προβολή τους επιτρέπει στην ιστοσελίδα να αλλάζει το περιεχόμενό της δυναμικά, χωρίς να χρειάζεται ολοκληρωτική ανανέωσή της.

Η χρήση του AJAX έχει συνεισφέρει στην ραγδαία εξέλιξη των διαδραστικών και δυναμικών εφαρμογών σε ιστοσελίδες. Αξίζει να σημειωθεί ότι η τεχνολογία AJAX δεν είναι μια τεχνολογία από μόνη της, αλλά ένας συνδυασμός τεχνολογιών. Η AJAX χρησιμοποιεί HTML και CSS για τη σήμανση της δομής και της εμφάνισης. Η χρήση της JavaScript σε συνδυασμό με το αντικείμενο XMLHttpRequest έρχεται να καλύψει τον χρόνο που κάνει μια σελίδα για να φορτώσει (page loading). Δηλαδή, με τη χρήση της τεχνολογίας αυτής, δεν υφίσταται page loading, παρά μόνο φόρτωση συγκεκριμένης πληροφορίας (partial loading). Το παραπάνω, μας προσφέρει μεγαλύτερη ταχύτητα και λιγότερο bandwidth – traffic, αφού πλέον δεν φορτώνεται ολόκληρη η σελίδα, αλλά μόνο το κομμάτι που θέλουμε να ανανεώσουμε.

2.7 XAMPP

Το XAMPP είναι ακρωνύμιο και αναφέρεται στα παρακάτω αρχικά:

- X (αναφέρεται στο «cross-platform» που σημαίνει λογισμικό ανεξάρτητο πλατφόρμας)
- Apache HTTP εξυπηρετητής
- MySQL
- PHP
- Perl

Το XAMPP είναι ελεύθερο λογισμικό το οποίο περιέχει έναν εξυπηρετητή ιστοσελίδων, το οποίο μπορεί να εξυπηρετεί και δυναμικές ιστοσελίδες τεχνολογίας PHP/MySQL. Είναι ανεξάρτητο πλατφόρμας και τρέχει σε Microsoft Windows, Linux, Solaris, and Mac OS X και χρησιμοποιείται ως πλατφόρμα για τη σχεδίαση και ανάπτυξη ιστοσελίδων με τις τεχνολογίες όπως PHP, JSP και Servlets.

Όταν το XAMPP εγκατασταθεί στον τοπικό υπολογιστή διαχειρίζεται τον localhost ως ένα απομακρυσμένο κόμβο, ο οποίος συνδέεται με το πρωτόκολλο μεταφοράς αρχείων FTP. Η σύνδεση στον localhost μέσω του FTP μπορεί να γίνει με το όνομα χρήστη «newuser» και το κωδικό «wampp». Για τη βάση δεδομένων MySQL υπάρχει ο χρήστης «root» χωρίς κωδικό πρόσβασης.

2.8 Jquery

Η jQuery είναι μια βιβλιοθήκη JavaScript σχεδιασμένη να απλοποιήσει την υλοποίηση σεναρίων (scripting) στην πλευρά του πελάτη (client-side) της HTML και υποστηρίζει πολλαπλούς φυλλομετρητές Ιστού. Κυκλοφόρησε τον Ιανουάριο του 2006 από τον Τζον Ρέριγκ (John Resig). Χρησιμοποιείται σε πάνω από το 65% των 10.000 ιστοτόπων με τη μεγαλύτερη επισκεψιμότητα.

Η jQuery είναι ελεύθερο λογισμικό, με άδεια MIT.

2.8.1 Χαρακτηριστικά

- DOM element επιλογές χρησιμοποιώντας την ανοιχτού κώδικα μηχανή επιλογής πολλαπλών φυλλομετρητών Sizzle
- DOM διάσχιση και τροποποίηση (υποστηρίζοντας CSS 1-3)
- Χειρισμός DOM βασισμένος σε CSS επιλογείς που χρησιμοποιεί τα id και class σαν κριτήρια για να κατασκευάσει επιλογείς
- Events
- Εφέ και κινητά στοιχεία
- AJAX
- Επεκτασιμότητα μέσω plug-ins
- Εργαλεία όπως πληροφορίες user-agent, ανίχνευση χαρακτηριστικών
- Μεθόδοι συμβατότητας που είναι εγγενώς διαθέσιμες σε σύγχρονα προγράμματα περιήγησης
- Υποστήριξη πολλαπλών φυλλομετρητών.

2.9 Bootstrap

Το Bootstrap είναι ένα front-end framework που δίνει τη δυνατότητα στους σχεδιαστές, να ελαχιστοποιήσουν τον χρόνο δημιουργίας μιας ιστοσελίδας από την αρχή. Αυτό επιτυγχάνεται με τη χρήση των ενσωματωμένων στοιχείων που παρέχει και είναι απαραίτητα για τον σχεδιασμό μιας σύγχρονης ιστοσελίδας. Περιέχει αρχεία html, Css, JavaScript καθώς και εικόνες Τι περιέχει το boostrap:

1) Εικονίδια
2) Αναπτυσσόμενα μενού
3) Κουμπιά διαφόρων χρωμάτων και μεγεθών
4) Φόρμες αποστολής
5) Μενού περιήγησης
6) Σελιδοποίηση
7) Ετικέτες
8) Έτοιμη δομή σελίδων
9) Κεφαλίδες
10) Προεπισκόπηση εικόνων
11) Ειδοποιήσεις

12) Μπάρες προόδου
13) Λίστες
14) Καρτέλες
15) Εφέ JavaScript

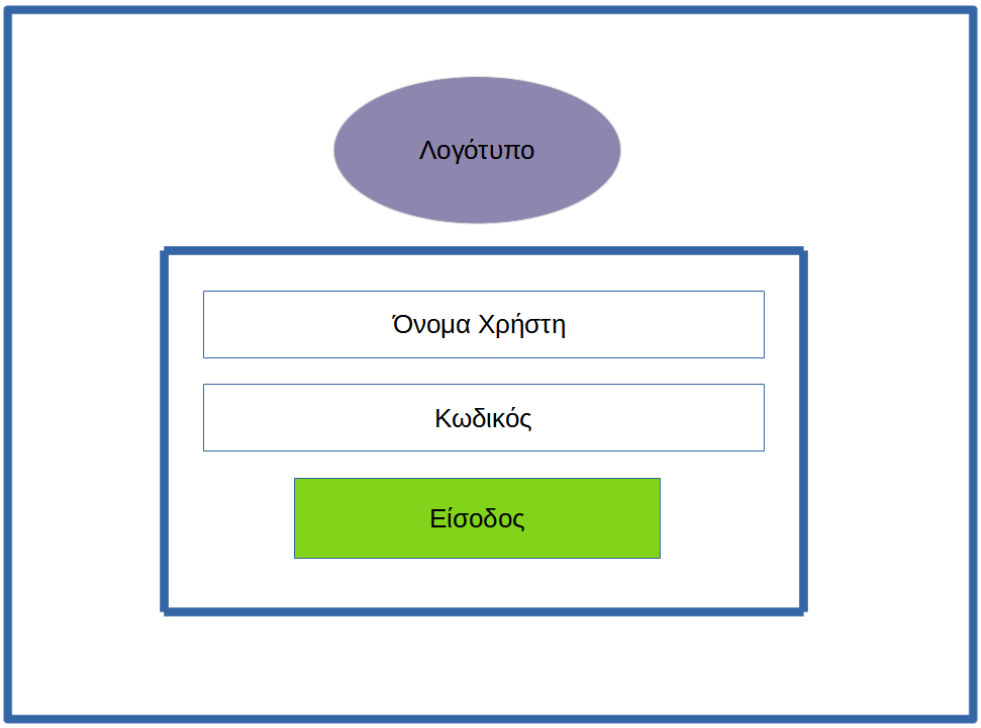
Κεφάλαιο 3: Λειτουργικές προδιαγραφές και Σχεδίαση της εφαρμογής

3.1 Σχεδίαση εφαρμογής

3.1.1 Είσοδος χρήστη

Η ιδέα για την οπτική απεικόνιση της σελίδας εισόδου ήταν ως εξής:

Στο κέντρο της σελίδας θα υπάρχει μία φόρμα με 2 κουτιά κειμένου, ένα για το όνομα χρήστη (username) και ένα για τον κωδικό. Κάτω από αυτά θα υπάρχει ένα κουμπί για την αποστολή αυτών των δεδομένων. Επίσης, πάνω από τη φόρμα, θα υπάρχει ένα λογότυπο μουσείου. Στην παρακάτω εικόνα φαίνεται μια γραφική αναπαράσταση αυτών που περιγράψαμε.



Εικόνα 1: Είσοδος Χρήστη

3.1.2 Εγγραφή νεου χρήστη

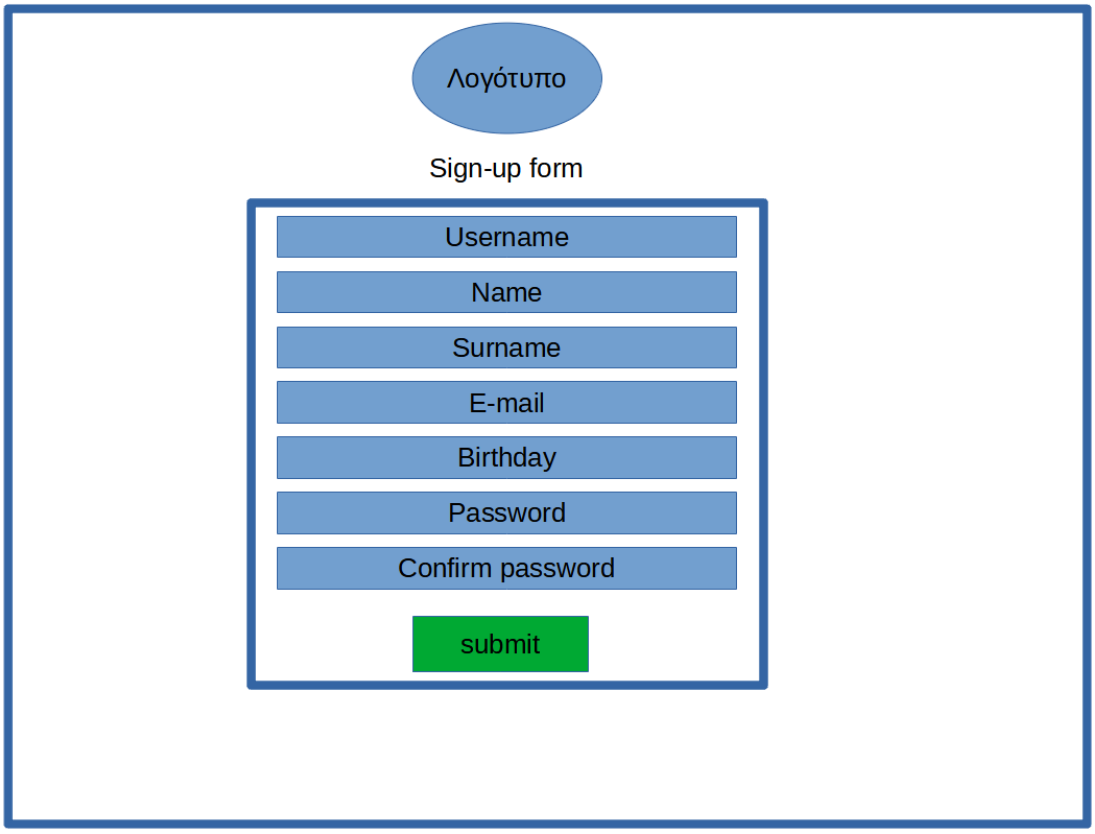
Η ιδέα για την οπτική απεικόνιση της σελίδας εγγραφής ήταν ως εξής:

Ο χρήστης πρέπει να συμπληρώσει τη φόρμα, που εμφανίζεται στο κέντρο της σελίδας. Αυτή περιέχει κάποια απαραίτητα στοιχεία, τα οποία θα αποθηκεύσουμε στη βάση μας. Αυτά είναι:

- Username
- Όνομα
- Επώνυμο

- Διεύθυνση ηλεκτρονικού ταχυδρομείου (e-mail)
- Ημερομηνία γέννησης
- Κωδικός (2 φορές για επιβεβαίωση)

Μόλις συμπληρώσει όλα τα υποχρεωτικά στοιχεία, πρέπει να πατήσει το πράσινο κουμπί (submit), ώστε να ολοκληρωθεί η διαδικασία εγγραφής του.



Εικόνα 2: Φόρμα εγγραφής νέου χρήστη

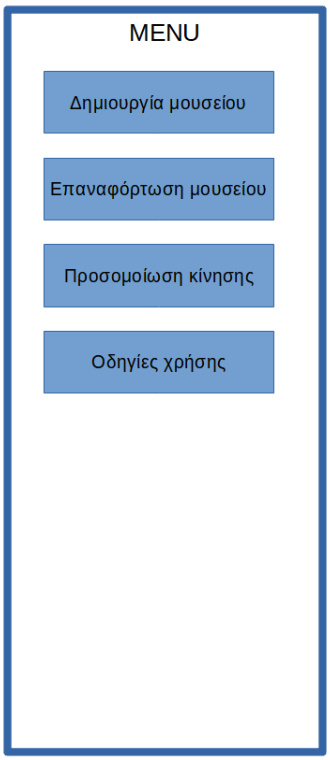
3.1.3 Αρχική σελίδα εφαρμογής

Η ιδέα για την οπτική απεικόνιση της αρχικής σελίδας της εφαρμογής ήταν ως εξής:

Αριστερά το μενού, μέσω του οποίου θα μπορεί ο χρήστης να περιηγηθεί στην εφαρμογή και δεξιά του ένας πίνακας με τα ονόματα των μουσείων, που έχει δημιουργήσει ο χρήστης.

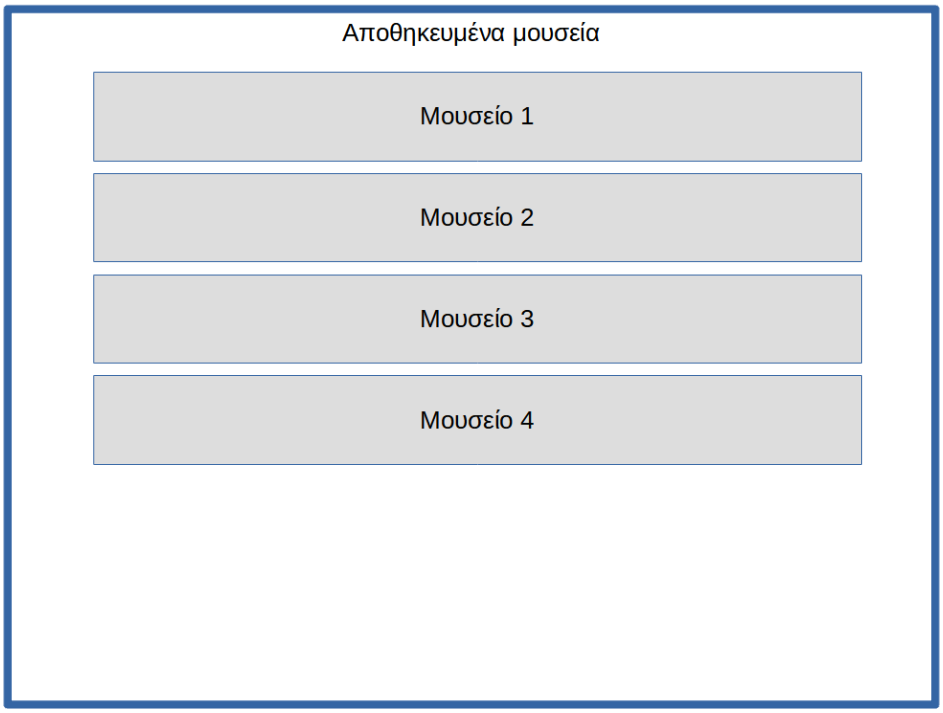
Εικόνα 3: Αρχική σελίδα εφαρμογής

Το menu, όπως είπαμε και παραπάνω θα περιλαμβάνει διάφορα κουμπιά, μέσω των οποίων ο χρήστης θα μπορεί να μεταβεί σε κάθε λειτουργία της εφαρμογής. Πιο συγκεκριμένα, θα υπάρχει ένα κουμπί για τη δημιουργία ενός καινούριου μουσείου, ένα για την επαναφόρτωση ενός ήδη υπάρχοντος και ένα για την προσομοίωση κίνησης επισκεπτών. Τέλος, θα υπάρχει και ένα κουμπί, το οποίο θα περιέχει ένα εγχειρίδιο χρήσης της εφαρμογής, ώστε να μπορεί ο χρήστης να τη χρησιμοποιήσει.



Εικόνα 4: Μενού επιλογών αρχικής σελίδας

Από την άλλη πλευρά, ο πίνακας μουσείων θα περιέχει όλα τα ονόματα μουσείων, που έχει δημιουργήσει ο χρήστης μέχρι αυτή τη στιγμή. Τα ονόματα των μουσείων θα εμφανίζονται με χρονολογική σειρά, δηλαδή όσο πιο πάνω βρισκόμαστε στη λίστα, τόσο πιο παλιά έχει δημιουργηθεί το μουσείο.

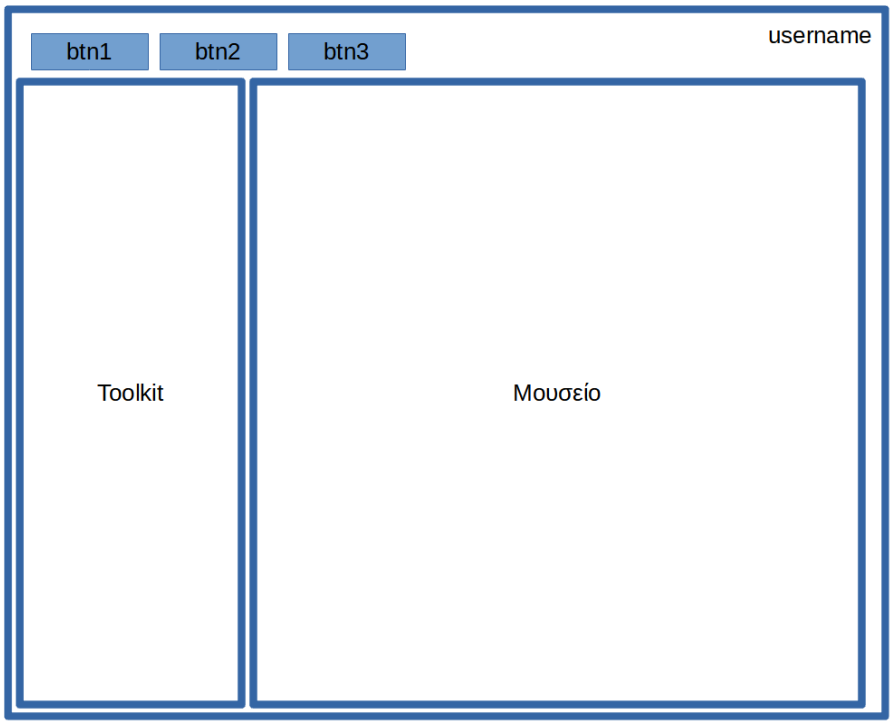


Εικόνα 5: Λίστα αποθηκευμένων μουσείων

3.1.4 Δημιουργία/Επεξεργασία μουσείου

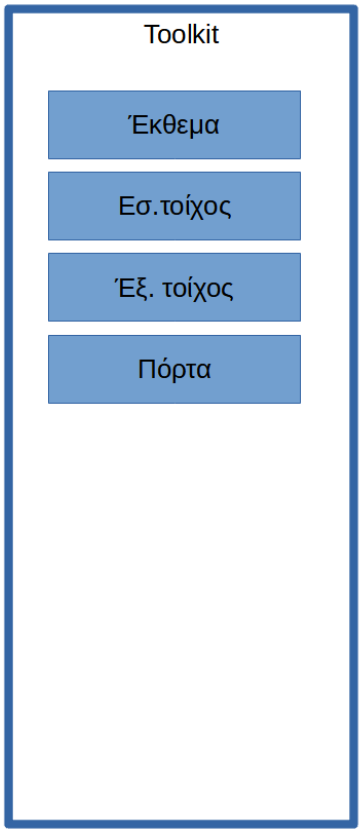
Η ιδέα για την οπτική απεικόνιση της σελίδας δημιουργίας- επεξεργασίας μουσείου της εφαρμογής ήταν ως εξής:

Αριστερά μία εργαλειοθήκη, η οποία περιέχει αντικείμενα για εισαγωγή στο μουσείο, και δεξιά ένας καμβάς, ο οποίος αναπαριστά το μουσείο. Επίσης πάνω αριστερά θα υπάρχει μία μπάρα με κάποια κουμπιά για επιπλέον λειτουργίες της εφαρμογής, όπως αποθήκευση, διαγραφή αντικειμένου κ.ά. Τέλος, στην αριστερή πλευρά της σελίδας και πάνω από το μουσείο θα εμφανίζεται το username του χρήστη, που χρησιμοποιεί την εφαρμογή.



Εικόνα 6: Σέλιδα δημιουργίας μουσείου

Η εργαλειοθήκη (toolkit), θα περιέχει διάφορα κουμπιά, μέσω των οποίων θα μπορεί ο χρήστης να εισάγει διάφορα αντικείμενα στο μουσείο του. Πιο συγκεκριμένα, θα υπάρχουν κουμπιά για την εισαγωγή εκθεμάτων, εσωτερικών και εξωτερικών τοίχων, πορτών κ.ά.



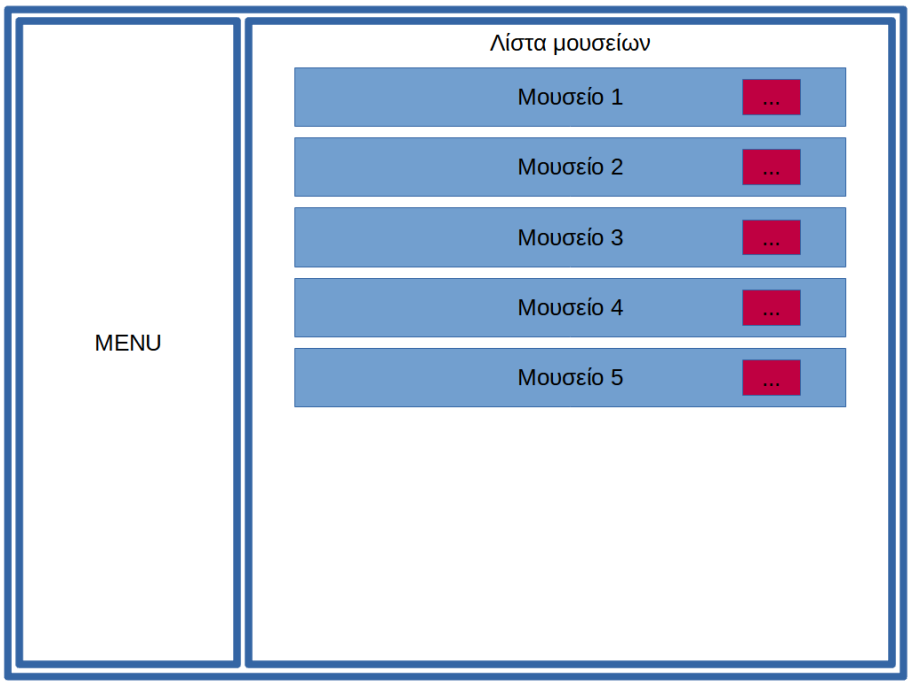
Εικόνα 7: Εργαλειοθήκη για την δημιουργία του μουσείου

Για να μπορέσει ο χρήστης να εισάγει κάποιο καινούριο αντικείμενο στο μουσείο του, θα πρέπει αρχικά να επιλέξει ένα από τα κουμπιά της εργαλειοθήκης (toolkit). Στη συνέχεια, αν θέλει να προσθέσει κάποιο έκθεμα πρέπει να κλικάρει σε ένα σημείο του μουσείου, που επιθυμεί να το τοποθετήσει. Από την άλλη πλευρά, αν επιλέξει το κουμπί, με όνομα εξωτερικός τοίχος, τότε θα δημιουργηθούν όλοι οι εξωτερικοί τοίχοι του μουσείου σε συγκεκριμένες θέσεις. Όσον αφορά το κουμπί πόρτα, ο χρήστης θα πρέπει να κλικάρει σε κάποιο σημείο, όπου υπάρχει εξωτερικός τοίχος, ώστε να μπορέσει να προστεθεί μία στο μουσείο. Τέλος, για τον εσωτερικό τοίχο, ο χρήστης θα πρέπει να επιλέξει ένα σημείο έναρξης του τοίχου και στη συνέχεια να σύρει το ποντίκι του προς οποιαδήποτε κατεύθυνση θέλει, ώστε να τον δημιουργήσει.

3.1.5 Επαναφόρτωση μουσείου

Η ιδέα για την οπτική απεικόνιση της σελίδας επαναφόρτωσης μουσείου ήταν ως εξής:

Αριστερά θα υπάρχει το menu περιήγησης στην εφαρμογή, ενώ δεξιά θα υπάρχει μία λίστα με τα μουσεία που έχει δημιουργήσει ο χρήστης. Σε κάθε γραμμή αυτής της λίστας θα υπάρχει το όνομα του μουσείου, καθώς και ένας υπερσύνδεσμος για την επαναφόρτωση του συγκεκριμένου.

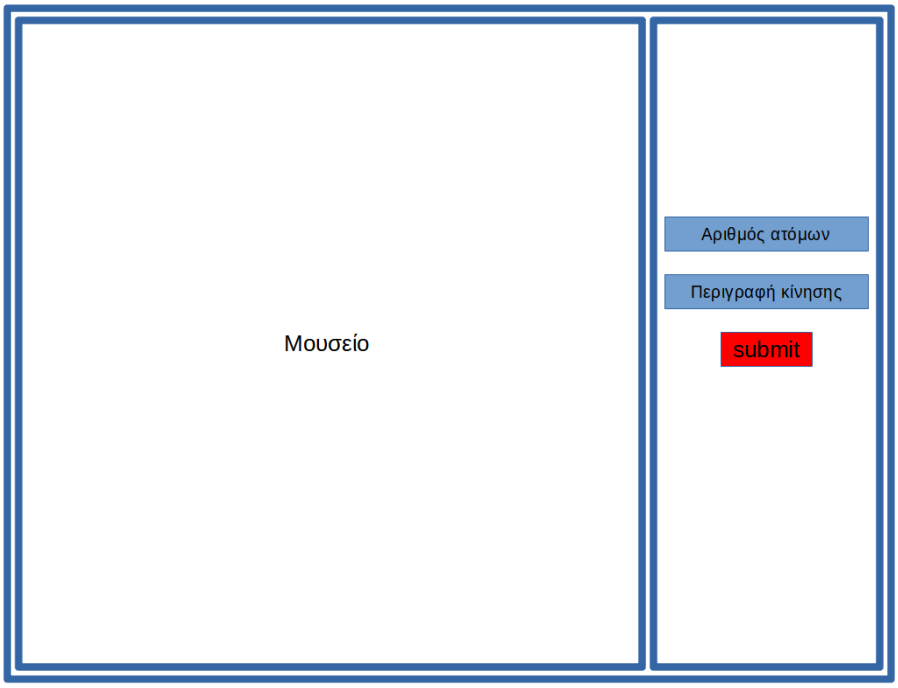


Εικόνα 8: Σελίδα επαναφόρτωσης μουσείου

3.1.6 Προσομοίωση κίνησης

Η ιδέα για την οπτική απεικόνιση της σελίδας προσομοίωσης κίνησης της εφαρμογής ήταν ως εξής:

Αριστερά το μουσείο, που επιλέγει ο χρήστης για προσομοίωση και δεξιά μία φόρμα με 2 κουτιά, ένα για τον αριθμό των ατόμων, που θα εισέλθουν στο μουσείο, και ένα για την περιγραφή της κίνησης κάθε ομάδας επισκεπτών.



Εικόνα 9: Σελίδα προσομοίωσης κίνησης

Μόλις ο χρήστης συμπληρώσει τα δύο παραπάνω πεδία και πατήσει το κουμπί submit ξεκινάει η προσομοίωση κίνησης. Όλα τα άτομα του γκρουπ επισκεπτών (αριθμός ατόμων) εισέρχονται στο μουσείο από την πόρτα εισόδου και πηγαίνουν από το ένα έκθεμα στο άλλο, με βάση τα στοιχεία, που έδωσε ο χρήστης στη περιγραφή κίνησης. Αυτή η κίνηση θα πραγματοποιείται με τη χρήση αλγορίθμων συντομότερης διαδρομής. Μόλις οι επισκέπτες εκτελέσουν τη συνολική κίνησή τους, κατευθύνονται προς την έξοδο του μουσείου, ώστε να αποχωρήσουν από αυτό. Κάθε επισκέπτης θα απεικονίζεται με τη μορφή SVG εικονιδίου (person-άτομο). Τέλος, θα δίνεται η δυνατότητα στον χρήστη να εισάγει πολλά διαφορετικά γκρουπ επισκεπτών στο μουσείο την ίδια χρονική στιγμή. Με αυτό τον τρόπο, η προσομοίωση κίνησης θα φαίνεται πιο αληθοφανής, διότι πολλά γκρουπ ατόμων βρίσκονται ταυτόχρονα σε ένα μουσείο, χωρίς να ακολουθούν την ίδια κίνηση.

3.1.7 Περιβάλλον λειτουργίας

Θα προσπαθήσουμε να δημιουγήσουμε μια εφαρμογή η οποία θα δουλέυει στην πλειοψηφία των ηλεκτρονικών υπολογιστών μέσω κάποιου εξυπηρετητή διαδικτύου. Οι δοκιμές της εφαρμογής μας θα γίνουν σε chrome based browsers, όπως Google Chrome, Brave browser και Opera Browser. Σκοπός μας είναι να λειτουργεί άριστα σε αυτούς τους τρεις εξυπηρετητές διαδικτύου.

3.2 Λειτουργίες συστήματος

Παρακάτω παρουσιάζονται οι βασικές λειτουργίες της εφαρμογής μας:

3.2.1 Εγγραφή νέου χρήστη

Για να μπορέσει ο χρήστης να εισέλθει στην εφαρμογή, πρέπει να δημιουργήσει έναν νέο λογαριασμό. Τα στοιχεία, που πρέπει να καταχωρήσει είναι τα εξής:

- Όνομα Χρήστη (username)
- Όνομα
- Επώνυμο
- E-mail

- Ημερομηνία γέννησης
- Κωδικός πρόσβασης (2 φορές για επαλήθευση)

Ο χρήστης μετά τη συμπλήρωση των στοιχείων του στο αντίστοιχο πεδίο της φόρμας επιλέγει 'Sign Up'. Σε περίπτωση, που θέλει να ακυρώσει τη διαδικασία επιλέγει 'Cancel'.

i. Ανάλυση σε λειτουργικές απαιτήσεις

REQ-1: Όλα τα πεδία θα πρέπει να έχουν πληροφορίες, δηλαδή δεν θα πρέπει να υπάρχουν κενά κελιά.

REQ-2: Το πεδίο e-mail, πρέπει να ακολουθεί ένα συγκεκριμένο πρότυπο, ώστε να θεωρείται σωστό. ({name}@{example}.{com})

REQ-3: Το πεδίο username δεν πρέπει να είναι ίδιο με κάποιο ήδη υπάρχον στη βάση μας.

REQ-4: Τα πεδία κωδικός και επιβεβαίωση κωδικού ελέγχονται για την ομοιότητά τους. Αν δεν είναι όμοια, δε μπορεί να προχωρήσει η εγγραφή.

ii. Ανάλυση σε μη λειτουργικές απαιτήσεις

Δεν υπάρχουν μη λειτουργικές απαιτήσεις.

3.2.2 Είσοδος χρήστη στο σύστημα (login)

Εμφανίζεται μία φόρμα και ο χρήστης πρέπει να συμπληρώσει τα παρακάτω πεδία:

- Όνομα χρήστη
- Κωδικός πρόσβασης

Ο χρήστης μετά τη συμπλήρωση των στοιχείων του στο αντίστοιχο πεδίο της φόρμας επιλέγει 'Sign in'. Σε περίπτωση που δεν επιβεβαιωθούν τα στοιχεία χρήστη, εμφανίζεται μήνυμα λάθους.

i. Ανάλυση σε λειτουργικές απαιτήσεις

REQ-1: Όλα τα πεδία θα πρέπει να είναι συμπληρωμένα, δηλαδή δε θα πρέπει να υπάρχουν κενά κελιά.

ii. Ανάλυση σε μη λειτουργικές απαιτήσεις

Δεν υπάρχουν μη λειτουργικές απαιτήσεις.

3.2.3 Αποσύνδεση χρήστη από το σύστημα (sign out)

Ο χρήστης επιλέγει 'Sign out ', ώστε να μπορέσει με επιτυχία να αποσυνδεθεί από το σύστημα.

i. Ανάλυση σε λειτουργικές απαιτήσεις

Δεν υπάρχουν λειτουργικές απαιτήσεις

ii. Ανάλυση σε μη λειτουργικές απαιτήσεις

NFREQ-1: Ο χρήστης θα μπορεί να επιβεβαιώσει την επιλογή του για έξοδο από το σύστημα.

3.2.4 Παρουσίαση λίστας μουσείων

Η εφαρμογή προβάλλει στον χρήστη το σύνολο των μουσείων, που έχει αποθηκεύσει μέχρι στιγμής στο σύστημα. Αυτή η λειτουργία εμφανίζεται σε 2 διαφορετικές σελίδες της εφαρμογής, στην αρχική

σελίδα κάθε χρήστη και σελίδα φόρτωσης των μουσείων. Στη δεύτερη περίπτωση εμφανίζονται για κάθε μουσείο δύο υπερσύνδεσμοι, κάθε ένας από τους οποίους εκτελεί και διαφορετική λειτουργία. Ο πρώτος είναι για την επαναφόρτωση του μουσείου και ο δεύτερος για την προσομοίωση κίνησης σε αυτό.

i. Ανάλυση σε λειτουργικές απαιτήσεις

REQ-1: Εμφάνιση σε μορφή πίνακα

ii. Ανάλυση σε μη λειτουργικές απαιτήσεις

NFREQ-1: Στη λίστα θα φαίνεται η ημερομηνία τελευταίας τροποποίησης

NFREQ-2: Εμφάνιση είδους μουσείου (π.χ. αρχαιολογικό, λαογραφικό κλπ.)

NFREQ-3: Προεπισκόπηση (preview) του μουσείου

3.2.5 Δημιουργία μουσείου

Ο χρήστης μπορεί να δημιουργήσει ένα καινούριο μουσείο από την αρχή. Μπορεί να επιλέξει από μια λίστα με κουμπιά, τι αντικείμενα θέλει να προσθέσει μέσα σε αυτό. Αυτή η λίστα περιέχει τα εξής κουμπιά:

- Έκθεμα
- Εξωτερικός τοίχος
- Εσωτερικός τοίχος
- Πόρτα

Έτσι, μπορεί να δημιουργήσει κάθε φορά ένα μοναδικό μουσείο, ανάλογα με τις επιλογές του.

i. Ανάλυση σε λειτουργικές απαιτήσεις

REQ-1: Για να μπορέσει ο χρήστης να ξεκινήσει τη δημιουργία του μουσείου, πρέπει αρχικά να προσθέσει εξωτερικούς τοίχους.

REQ-2: Μόλις ο χρήστης επιλέξει κάποιο κουμπί δεν μπορεί να επιλέξει κάποιο άλλο μέχρι να προσθέσει στο μουσείο το αντικείμενο που αναφέρεται σε αυτό.

ii. Ανάλυση σε μη λειτουργικές απαιτήσεις

Δεν υπάρχουν μη λειτουργικές απαιτήσεις.

3.2.6 Επεξεργασία μουσείου

Ο χρήστης μπορεί να επαναφορτώσει ένα ήδη αποθηκευμένο μουσείο όσες φορές επιθυμεί, με σκοπό να πραγματοποιήσει διάφορες αλλαγές σε αυτό.

i. Ανάλυση σε λειτουργικές απαιτήσεις

Οι λειτουργικές απαιτήσεις είναι ίδιες με τις λειτουργικές απαιτήσεις του 5.

ii. Ανάλυση σε μη λειτουργικές απαιτήσεις

Δεν υπάρχουν μη λειτουργικές απαιτήσεις.

3.2.7 Διαγραφή αντικειμένου

Ο χρήστης μπορεί να επιλέξει κάποιο αντικείμενο που υπάρχει στο μουσείο του, ώστε να το διαγράψει από αυτό.

i. Ανάλυση σε λειτουργικές απαιτήσεις

REQ-1: Είναι δυνατή η διαγραφή μόνο ενός αντικειμένου τη φορά.

REQ-2: Μόλις ο χρήστης επιλέξει το κουμπί της διαγραφής, δε μπορεί να επιλέξει κάποιο άλλο κουμπί μέχρι να διαγράψει κάποιο αντικείμενο.

ii. Ανάλυση σε μη λειτουργικές απαιτήσεις

Δεν υπάρχουν μη λειτουργικές απαιτήσεις.

3.2.8 Αποθήκευση μουσείου

Ο χρήστης μπορεί να επιλέξει να αποθηκεύσει ένα μουσείο, που μόλις έχει δημιουργήσει, μέσα στη βάση δεδομένων μας. Αφού πατήσει το κουμπί της αποθήκευσης, εμφανίζεται ένα παράθυρο διαλόγου, μέσω του οποίου επιλέγει το όνομα, που θέλει να έχει το αποθηκευμένο μουσείο. Μόλις ο χρήστης πληκτρολογήσει το όνομα, που θέλει, επιλέγει 'Submit'. Σε περίπτωση, που θέλει να ακυρώσει τη διαδικασία επιλέγει 'Cancel'.

Μέσω αυτής της λειτουργίας, είναι δυνατό να πραγματοποιηθούν επίσης και οι λειτουργίες επαναφόρτωσης μουσείου και προσομοίωσης κίνησης.

i. Ανάλυση σε λειτουργικές απαιτήσεις

REQ-1: Το πεδίο, που υπάρχει στο παράθυρο διαλόγου δεν πρέπει να είναι κενό. Σε αυτή την περίπτωση, ο χρήστης ενημερώνεται με κατάλληλο μήνυμα λάθους ότι το όνομα που πληκτρολόγησε δεν είναι έγκυρο και καλείται να επαναλάβει τη διαδικασία επιλογής ονόματος.

REQ-2: Το όνομα του μουσείου δεν πρέπει να είναι ίδιο με κάποιο ήδη αποθηκευμένο μουσείο στο λογαριασμό του χρήστη. Αν δεν ισχύει αυτό, τότε η αποθήκευση ακυρώνεται.

REQ-3: Για να πραγματοποιηθεί η αποθήκευση πρέπει να ισχύουν κάποιες προϋποθέσεις. Αυτές είναι:

- Ύπαρξη εξωτερικών τοίχων
- Ύπαρξη τουλάχιστον ενός εκθέματος
- Ύπαρξη τουλάχιστον μίας πόρτας

ii. Ανάλυση σε μη λειτουργικές απαιτήσεις

Δεν υπάρχουν μη λειτουργικές απαιτήσεις.

3.2.9 Προσομοίωση κίνησης

Ο χρήστης μπορεί να προσομοιώσει την κίνηση γκρουπ επισκεπτών στο μουσείο του. Για να συμβεί αυτό πρέπει να συμπληρώσει μία φόρμα με τα ακόλουθα πεδία:

- Πλήθος ατόμων
- Μονοπάτι κίνησης
- Κατηγορία γκρουπ ατόμων

Ο χρήστης μετά τη συμπλήρωση όλων των πεδίων της φόρμας επιλέγει ‘Submit’. Υπάρχει η δυνατότητα ύπαρξης πολλών διαφορετικών γκρουπ ατόμων μέσα στο μουσείο την ίδια στιγμή.

i. Ανάλυση σε λειτουργικές απαιτήσεις

REQ-1: Όλα τα πεδία θα πρέπει να έχουν πληροφορίες, δηλαδή δεν θα πρέπει να υπάρχουν κενά κελιά.

REQ-2: Το πεδίο μονοπάτι κίνησης, πρέπει να ακολουθεί ένα συγκεκριμένο πρότυπο, ώστε να θεωρείται σωστό. ({Εκθεμα},{Εκθεμα},…)

REQ-3: Το πεδίο πλήθος ατόμων πρέπει περιέχει μία τιμή από 1 μέχρι 100.

REQ-4: Το πεδίο μονοπάτι κίνησης δεν πρέπει να περιέχει κάποιο έκθεμα που δεν υπάρχει στο μουσείο. Σε αυτή τη περίπτωση, ο χρήστης ενημερώνεται με κατάλληλο μήνυμα λάθους και η προσομοίωση ακυρώνεται.

REQ-5: Το πεδίο κατηγορία γκρουπ ατόμων έχει ως προκαθορισμένη τιμή την πρώτη επιλογή (‘Οικογένεια’).

ii. Ανάλυση σε μη λειτουργικές απαιτήσεις

Δεν υπάρχουν μη λειτουργικές απαιτήσεις.

3.2.10 Δημιουργία heatmap

Αφού ο χρήστης εκτελέσει έναν ικανοποιητικό αριθμό κινήσεων για διάφορα γκρουπ επισκεπτών, η εφαρμογή του δίνει τη δυνατότητα παραγωγής ενός heatmap επισκεψιμότητας για όλο το μουσείο. Μέσω αυτού, μπορεί να παρατηρήσει την επισκεψιμότητα κάθε σημείου του μουσείου του, με βάση τις παραπάνω κινήσεις.

i. Ανάλυση σε λειτουργικές απαιτήσεις

REQ-1: Αν δεν έχει πραγματοποιηθεί κάποια προσομοίωση κίνησης, τότε η εφαρμογή ενημερώνει τον χρήστη με κατάλληλο μήνυμα και η δημιουργία του heatmap ακυρώνεται.

ii. Ανάλυση σε μη λειτουργικές απαιτήσεις

NFREQ-1: Δημιουργία heatmap για κάθε κατηγορία γκρουπ ατόμων ξεχωριστά.

3.2.11 Αποθήκευση heatmap

Ο χρήστης μπορεί να αποθηκεύσει το heatmap, που δημιουργήθηκε, μαζί με το μουσείο του τοπικά στον υπολογιστή του. Μόλις επιλέξει το αντίστοιχο κουμπί, εμφανίζεται ένα παράθυρο διαλόγου, όπου του ζητάει ένα όνομα, με το οποίο θα αποθηκεύσει τα παραπάνω. Η μορφή του αρχείου αποθήκευσης είναι **PNG** (αρχείο εικόνας).

i. Ανάλυση σε λειτουργικές απαιτήσεις

REQ-1: Το πεδίο, που υπάρχει στο παράθυρο διαλόγου, δεν πρέπει να είναι κενό. Σε αυτή τη περίπτωση, η εφαρμογή ενημερώνει το χρήστη με κατάλληλο μήνυμα λάθους, ότι το όνομα που πληκτρολόγησε δεν είναι έγκυρο και καλείται να επαναλάβει τη διαδικασία.

REQ-2: Αν δεν έχει δημιουργηθεί κάποιο heatmap επισκεψιμότητας για το μουσείο, τότε η εφαρμογή ενημερώνει κατάλληλα τον χρήστη και η αποθήκευση του heatmap ακυρώνεται.

ii. Ανάλυση σε μη λειτουργικές απαιτήσεις

Δεν υπάρχουν μη λειτουργικές απαιτήσεις.

Κεφάλαιο 4: Υλοποίηση της εφαρμογής

4.1 Υλοποίηση βάσης δεδομένων

Η βάση δεδομένων της εφαρμογής μας αποτελείται από τους παρακάτω βασικούς πίνακες:

- Users
- User_rooms
- Visitors

Ο πίνακας users περιέχει πληροφορίες που αφορούν τους χρήστες της εφαρμογής. Αποτελείται από τα εξής πεδία:

1. user_id: είναι το πρωτεύον κλειδί και είναι ένας αύξων αριθμός, που αντιστοιχεί στο μοναδικό id του κάθε χρήστη
2. name: είναι το όνομα του χρήστη
3. surname: είναι το επώνυμο του χρήστη
4. email: είναι το email του χρήστη
5. password: είναι το πεδίο, που αποθηκεύει τον κωδικό, τον οποίο έχει επιλέξει ο χρήστης για τον λογαριασμό του
6. birthday: είναι η ημερομηνία γέννησης του χρήστη
7. username: είναι το username του χρήστη

Ο πίνακας user_rooms περιέχει πληροφορίες, που αφορούν τα δωμάτια, τα οποία έχουν κατασκευάσει οι χρήστες. Αποτελείται από τα εξής πεδία:

1. user_rooms_id: είναι το πρωτεύον κλειδί και είναι ένας αύξων αριθμός, που αντιστοιχεί στο μοναδικό id του κάθε δωματίου, που δημιουργούν οι χρήστες
2. name: είναι το όνομα του δωματίου, που επέλεξε ο χρήστης
3. user_id (foreign key): είναι ο κωδικός αριθμός του χρήστη, που δημιούργησε και αποθήκευσε το μουσείο. Οπότε το πεδίο αυτό είναι ξένο κλειδί στον πίνακα **users**

Ο πίνακας visitors περιέχει πληροφορίες, που αφορούν τις κινήσεις των επισκεπτών μέσα στα μουσεία που έχουν δημιουργήσει οι χρήστες. Αποτελείται από τα εξής πεδία:

1. visitors_id: είναι το πρωτεύον κλειδί και είναι ένας αύξων αριθμός, που αντιστοιχεί στο μοναδικό id της κάθε κίνησης ενός γκρουπ επισκεπτών
2. path: είναι ο αριθμός κάθε εκθέματος που θα επισκεφθεί το γκρουπ στην κίνησή του, χωρισμένα με κόμμα (,)
3. number_visitors: Ο ακριβής αριθμός του γκρουπ επισκεπτών (η τιμή του κυμαίνεται από 1 μέχρι 100)
4. room_id (foreign key): είναι ο κωδικός αριθμός του δωματίου, μέσα στο οποίο πραγματοποιείται η κίνηση. Οπότε το πεδίο αυτό είναι ξένο κλειδί στον πίνακα user_rooms.

4.2 Παρουσίαση κυριότερων λειτουργιών της εφαρμογής

4.2.1 Κοινά σημεία μεταξύ αρχείων

Κοινά κομμάτια κώδικα μπορούν να βρεθούν σε όλα τα php αρχεία. Αυτά τα κομμάτια αφορούν ελέγχους, που γίνονται για να διαπιστώσουμε αν ο χρήστης έχει συνδεθεί στο σύστημα. Αν αυτό δεν ισχύει τότε η εφαρμογή επιστρέφει τον χρήστη στη σελίδα εισόδου. Αυτό το κομμάτι μπορούμε να το παρατηρήσουμε στην εικόνα

```
1.  <?php
2.      session_start();
3.      include "database.php";
4.  if (!(isset($_SESSION['name'])))
5.  {
6.  header ("location: ../index.html");
7.  die;
8.  }
9.  ?>
```

source code 1

Για να πραγματοποιηθεί αυτός ο έλεγχος, χρησιμοποιούμε ένα SESSION (σύνοδος). Το SESSION είναι μια μεταβλητή, που αποθηκεύεται στον server και περιέχει πληροφορίες για τον χρήστη, που έχει συνδεθεί. Η μεταβλητή παραμένει αποθηκευμένη μέχρι ο χρήστης να κλείσει τον browser, ή να αποσυνδεθεί από την εφαρμογή, οπότε η μεταβλητή θα γίνει unset. Το όνομα της συνόδου στην εφαρμογή είναι name, με το οποίο αποθηκεύουμε το username του χρήστη. Με την εντολή isset($_SESSION['name']) ελέγχουμε αν υπάρχει κάποια αποθηκευμένη σύνοδος. Σε περίπτωση που δεν υπάρχει, τότε η εφαρμογή ανακατευθύνει τον χρήστη στη σελίδα index.html.

4.2.2 Σύνδεση με την βάση δεδομένων-Conn.php

Το αρχείο περιέχει στοιχεία σύνδεσης με τη βάση δεδομένων.

```
1.  <?php
2.   $con=mysqli_connect("localhost", "root", "");
3.      if(!$con)
4.           die('Connection Failed'.mysql_error());
5.
6.       mysqli_select_db($con,"museumdb")
7.  ?>
```

source code 2

Σε αυτό το αρχείο δημιουργείται μία νέα σύνδεση στον MySQL server και στη συνέχεια, επιλέγουμε τη βάση στην οποία θέλουμε να εκτελέσουμε τα ερωτήματα (queries).

4.2.3 Είσοδος χρήστη – login.php

Σε αυτό το αρχείο πραγματοποιείται η είσοδος των χρηστών στην εφαρμογή. Κάθε φορά, που ο χρήστης προσπαθεί να συνδεθεί στην εφαρμογή, χρησιμοποιώντας τα credentials του, καλείται αυτό το αρχείο.

```
1.  <?php
2.  require 'conn.php';
3.  session_start();
4.  include "database.php";
5.  $username = $_POST["username"];
6.  $password = trim($_POST["pass"]);
```

```
7.
8.
9.
10. $result = mysqli_query($con,"SELECT * FROM users WHERE username = \"$username\" AND
    password =\"$password\"");
11. $count=mysqli_num_rows($result);
12. if($count==1){
13.
14.         $_SESSION['isAuthenticated'] = true;
15.         $_SESSION['name']        = $username;
16.         header('Location: '.$login_redir.'');
17.
18. }
19. else{
20.     mysqli_close($con);
21.     header('.Location: '.$logout_redirect.'');
22. }
23.
24. ?>
```

source code 3

Στον παραπάνω κώδικα βλέπουμε τη διαδικασία με την οποία γίνεται η σύνδεση ενός χρήστη στην εφαρμογή. Οι μεταβλητές username και password έχουν τα δεδομένα, που έδωσε ο χρήστης κατά την είσοδό του. Στη γραμμή 10 ελέγχουμε αν υπάρχει κάποιος χρήστης με αυτά τα στοιχεία στη βάση. Αν υπάρχει ακριβώς ένας χρήστης με αυτά τα στοιχεία, τότε αρχικοποιείται το SESSION στις γραμμές 14 και 15 και στη συνέχεια ο χρήστης μεταφέρεται στην κεντρική σελίδα της εφαρμογής μας.

Σε περίπτωση, που το SQL ερώτημα δεν επιστρέψει κάποιο αποτέλεσμα, τότε η σύνδεση με τη βάση κλείνει (γραμμή 20) και ο χρήστης μεταφέρεται εκ νέου στην σελίδα index.html.

4.2.4 Έξοδος χρήστη – logout.php

Στη συνέχεια φαίνεται η σελίδα logout.php. Σε αυτή διαγράφονται από τον server όλες οι μεταβλητές τύπου SESSION, που δημιουργήθηκαν κατά την είσοδο του χρήστη στην εφαρμογή. Επίσης η σύνδεση στη βάση δεδομένων κλείνει. Τέλος, η εφαρμογή ανακατευθύνει τον χρήστη στην σελίδα index.html.

```
1.  <?php
2.      session_start();
3.      require 'conn.php';
4.      session_destroy();
5.      unset($_SESSION['isAuthenticated']);
6.      unset ($_SESSION['name']);
7.      mysqli_close($con);
8.      include "database.php";
9.      header('Location:'.$logout_redirect.'');
10. ?>
```

source code 4

4.2.5 Μετακίνηση αντικειμένου

Μία από τις κυριότερες λειτουργίες της εφαρμογής είναι η δυνατότητα μετακίνησης εκθεμάτων και εσωτερικών τοίχων του μουσείου. Όταν ο χρήστης εισάγει ένα καινούριο αντικείμενο στο μουσείο του, πρέπει να επιλέξει τη θέση στην οποία επιθυμεί να το τοποθετήσει. Όμως ο χρήστης πρέπει να έχει τη δυνατότητα αλλαγής θέσης αυτού του αντικειμένου καθ'όλη τη διάρκεια δημιουργίας του μουσείου. Αυτό επιτυγχάνεται με τον παρακάτω κώδικα.

```
1.  //makes all objects inside the viebox draggable
2.  function makeDraggable(evt) {
```

```
3.    var svg = evt.target;
4.    var dragx;
5.    var dragy;
6.    svg.addEventListener('mousedown', startDrag);
7.    svg.addEventListener('mousemove', drag);
8.    svg.addEventListener('mouseup', endDrag);
9.    svg.addEventListener('mouseleave', endDrag);
```

source code 5

Η παραπάνω λειτουργία αποτελείται από 3 διαφορετικά τμήματα:

- Πάτημα του ποντικιού, όπου πρέπει να βρούμε ποιο αντικείμενο κλίκαρε ο χρήστης
- Μετακίνηση του ποντικιού, όπου πρέπει να κουνήσουμε το αντικείμενο
- Απελευθέρωση του ποντικιού, όπου πρέπει να αφήσουμε το αντικείμενο, ώστε να μην κουνιέται όταν μετακινούμε το ποντίκι

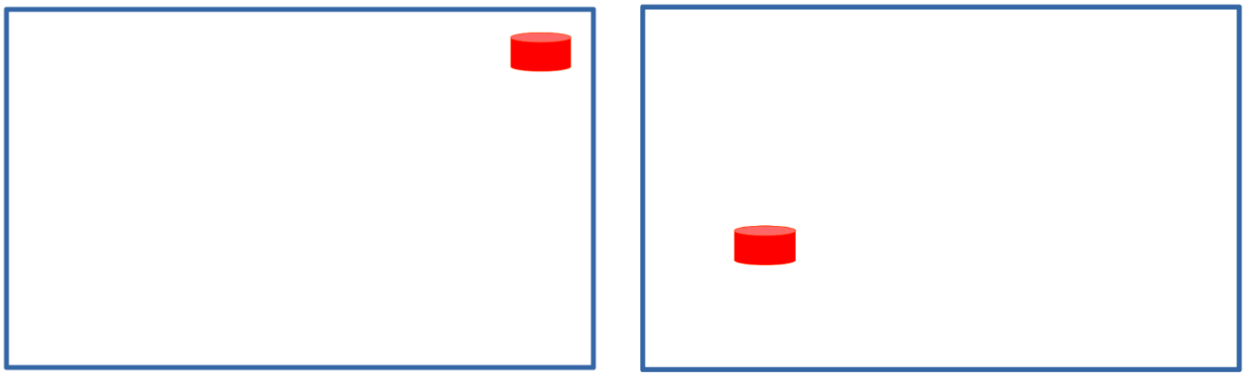
Αρχικά δημιουργήσαμε τη συνάρτηση makeDraggable, η οποία δέχεται το αντικείμενο, που θέλει ο χρήστης να μετακινήσει, ως όρισμα. Έπειτα προσθέσαμε 4 event listeners, τους mousedown, mousemove, mouseup και mouseleave, όπου ο καθένας αντιπροσωπεύει μία συγκεκριμένη λειτουργία του αντικειμένου.

```
10.   function startDrag(evt) {
11.       //if there is a button pressed don't drag the object
12.     if(!button2Clicked && !button4Clicked){
13.
14.         //constrain the movement of the objects inside the viewbox
15.
16.         if (evt.target.classList.contains('draggable')) {
17.             selectedElement = evt.target;
18.             offset = getMousePosition(evt);
19.
20.             var transforms = selectedElement.transform.baseVal;
21.             console.log(transforms);
22.             // Ensure the first transform is a translate transform
23.             if (transforms.length === 0  ||transforms.getItem(0).type !== SVGTransf
    orm.SVG_TRANSFORM_TRANSLATE) {
24.                 // Create a transform that translates by (0, 0)
25.                 var translate = svg.createSVGTransform();
26.                 translate.setTranslate(0, 0);
27.                 // Add the translation to the front of the transforms list
28.                 selectedElement.transform.baseVal.insertItemBefore(translate, 0);
29.             }
30.             // Get initial translation amount
31.             transform = transforms.getItem(0);
32.             offset.x -= transform.matrix.e;
33.             offset.y -= transform.matrix.f;
34.
35.             confined = evt.target.classList.contains('confine');
36.         if (confined) {
37.             bbox = evt.target.getBBox();
38.             minX = boundaryX1 - bbox.x;
39.             maxX = boundaryX2 - bbox.x - bbox.width;
40.             minY = boundaryY1 - bbox.y;
41.             maxY = boundaryY2 - bbox.y - bbox.height;
42.         }
43.
44.         }
45.     }
46.   }
```

source code 6

Στη συνέχεια προσθέσαμε τη συνάρτηση startDrag μέσα στην makeDraggable. Σε αυτή ελέγχουμε αν το αντικείμενο που κλίκαρε ο χρήστης περιέχει την κλάση draggable. Αν αυτό ισχύει, τότε η μεταβλητή selectedElement γίνεται ίση με το αντικείμενο. Το πρόβλημα, που αντιμετωπίσαμε σε αυτό το σημείο είναι ότι οι περισσότερες συναρτήσεις μετακίνησης λειτουργούσαν ενημερώνοντας απλά τα χαρακτηριστικά x και y του επιλεγμένου αντικειμένου. Όμως υπήρχαν αντικείμενα στην εφαρμογή, τα οποία δεν είχαν αυτά τα χαρακτηριστικά. Για να μπορεί, λοιπόν, η συνάρτηση να λειτουργεί για όλα τα αντικείμενα, χρησιμοποιήσαμε transforms (μετατροπές). Η λογική είναι ότι οι συντεταγμένες, που περιέχει η κλάση κάθε αντικειμένου (π.χ. x,y για εικόνες ή x1,y1,x2,y2 για γραμμές) αποθηκεύουν την αρχική θέση του αντικειμένου και δεν αλλάζουν ποτέ. Μόλις κάθε αντικείμενο μετακινηθεί για πρώτη φορά, δημιουργείται το χαρακτηριστικό transform. Αυτό υποδηλώνει το πόσο μετακινήθηκε το αντικείμενο σε σχέση με την αρχική του θέση. Οπότε για να μπορέσουμε να υπολογίσουμε αργότερα την πραγματική θέση ενός αντικειμένου προσθέτουμε στην αρχική του θέση αυτό το χαρακτηριστικό. Για να γίνει περισσότερο κατανοητό ας δούμε ένα παράδειγμα.

Παρακάτω παρατηρούμε ένα χώρο 100 x 50, καθώς και ένα αντικείμενο μέσα σε αυτό. Η αρχική θέση του αντικειμένου φαίνεται στην εικόνα στα αριστερά, ενώ στα δεξιά παρατηρούμε την τελική του θέση, μετά από μετακίνηση με τον κέρσορα.



Τα χαρακτηριστικά x και y του αντικειμένου είναι ίδια και στις δύο εικόνες. Αυτό που αλλάζει είναι το χαρακτηριστικό transform. Στην πρώτη εικόνα, το χαρακτηριστικό δεν έχει δημιουργηθεί ακόμα, διότι δεν έχει υπάρξει κάποια μεταβολή της θέσης του. Από την άλλη πλευρά, στη δεύτερη εικόνα το χαρακτηριστικό transform έχει δημιουργηθεί και έχει τιμή ανάλογη της μετατόπισης του στον κάθε άξονα. Π.χ. αν υποθέσουμε ότι η αρχική θέση ήταν (90,10), και το αντικείμενο μετά τη μετακίνηση με τον κέρσορα βρίσκεται στη θέση (20,35), τότε το χαρακτηριστικό transform θα έχει τιμή (-70,25). Έτσι για να υπολογίσουμε την τελική θέση στην εφαρμογή μας, εφαρμόζουμε την ακόλουθη πράξη:

Τελική_θέση(x,y)= (αρχική_θέση_x + transform_x , αρχική_θέση_y + transform_y)

Τέλος, χρησιμοποιώντας τις παρακάτω συντεταγμένες δημιουργούμε τα όρια του μουσείου, για τα αντικείμενα, που περιέχουν την κλάση confine, ώστε να μην χτυπούν στους τοίχους ή να βγαίνουν εκτός μουσείου.

```
1.  //boundaries are used for the confine class, in order to prevent exhibits of
    leaving the museum or hit the walls
2.  var boundaryX1 = 7;
3.  var boundaryX2 = 95;
4.  var boundaryY1 = 7;
5.  var boundaryY2 = 45;
```

source code 7

```
47.
48.   function drag(evt) {
49.     if (selectedElement) {
50.         evt.preventDefault();
51.         var coord = getMousePosition(evt);
52.         var dx=coord.x - offset.x;
53.         var dy=coord.y - offset.y;
54.         if (confined) {
55.             if (dx < minX) { dx = minX; }
56.             else if (dx > maxX) { dx = maxX; }
57.             if (dy < minY) { dy = minY; }
58.             else if (dy > maxY) { dy = maxY; }
59.         }
60.         transform.setTranslate(dx, dy);
61.
62.
63.
64.
65.     }
66.   }
```

source code 8

Η επόμενη συνάρτηση, που εισαγάγαμε ήταν η drag. Αφού ελέγξουμε αν υπάρχει κάποιο επιλεγμένο αντικείμενο, χρησιμοποιείται η συνάρτηση preventDefault(), η οποία αποκλείει οποιαδήποτε άλλη ενέργεια μετακίνησης. Για παράδειγμα, αν ο χρήστης μετακινήσει ένα αντικείμενο πάνω από κάποιο άλλο, τότε το δεύτερο αντικείμενο δεν θα μετακινηθεί. Στη συνέχεια, αφού βρούμε τις συντεταγμένες του ποντικιού ελέγχουμε αν η νέα θέση του αντικειμένου βρίσκεται εντός του μουσείου. Αν ναι, τότε ενημερώνουμε τις συντεταγμένες μετατροπής με την καινούρια θέση.

```
67.   function endDrag(evt) {
68.       if(selectedElement){
69.             selectedElement = false;
70.
71.         }
72.   }
73.     function getMousePosition(evt) {
74.       var CTM = svg.getScreenCTM();
75.       return {
76.         x: (evt.clientX - CTM.e) / CTM.a,
77.         y: (evt.clientY - CTM.f) / CTM.d
78.       };
79.     }
80. }
```

source code 9

Επόμενη συνάρτηση, που πρέπει να αναλύσουμε είναι η getMousePosition(). Το πρόβλημα, που προκύπτει είναι ότι τα χαρακτηριστικά clientX και clientY του αντικειμένου evt υποδηλώνουν τις συντεταγμένες του ποντικιού χρησιμοποιώντας το σύστημα συντεταγμένων της οθόνης. Πρέπει λοιπόν να γνωρίζουμε τις συντεταγμένες του μουσείου μας (SVG space), οι οποίες ορίζονται από το χαρακτηριστικό viewBox. Για να γίνει αυτό, μπορούμε να χρησιμοποιήσουμε τη μέθοδο getScreenCTM του SVG αντικειμένου. Αυτή επιστρέφει το Current Transformation Matrix της οθόνης, το οποίο είναι ένα αντικείμενο με έξι κλειδιά, τα a,b,c,d,e,f.

Αυτό που πρέπει να γνωρίζουμε είναι ότι αν ένα αντικείμενο έχει χαρακτηριστικά της μορφής (x,y), τότε οι συντεταγμένες του στην οθόνη θα είναι της μορφής (ax+e, dy+f). Οπότε, για να βρούμε τη θέση του ποντικιού στο SVG space, υπολογίζουμε απλά το αντίστροφο.

Τέλος, υπάρχει και η συνάρτηση endDrag, η οποία απλά δίνει την τιμή false στο selectedElement, ώστε να μην συνεχίζει ο χρήστης να το μετακινεί αφού έχει ήδη αφήσει το ποντίκι.

4.2.6 Εισαγωγή εκθέματος

Παρακάτω βλέπουμε τον κώδικα βάσει του οποίου επιτυγχάνεται η εισαγωγή ενός εκθέματος στο μουσείο. Μόλις ο χρήστης πατήσει το κουμπί **ΕΚΘΕΜΑ**, εμφανίζεται ένα καινούριο μενού με 3 επιλογές (μικρό, μεσαίο, μεγάλο). Αυτή η επιλογή του μεταφέρεται στη συνάρτηση createImage μέσω της μεταβλητής type.

```
1.  //onclick function of the 1st button (image)
2.  //type determines the size of the image
3.  function createImage(type){
4.      if(polyline_flag){
5.          museum.addEventListener("click",function _listener(event){
6.              var myImage = document.createElementNS(svgNS,"image");
7.              var start= museumPoint(museum,event.clientX,event.clientY);
8.              if(type=="small"){
9.                  myImage.setAttributeNS(null,"height","5");
10.                 myImage.setAttributeNS(null,"width","3");
11.                 myImage.setAttributeNS(xlink,"href","../images/small.png");
12.             }
13.             else if(type=="medium"){
14.                 myImage.setAttributeNS(null,"height","8");
15.                 myImage.setAttributeNS(null,"width","4");
16.                 myImage.setAttributeNS(xlink,"href","../images/sparta.png");
17.             }
18.             else if(type=="large"){
19.                 myImage.setAttributeNS(null,"height","10");
20.                 myImage.setAttributeNS(null,"width","10");
21.                 myImage.setAttributeNS(xlink,"href","../images/europi.jpeg");
22.             }
23.             else{
24.                 alert("Something wrong happened");
25.             }
26.             numImages++;
```

source code 10

Για να προσθέσει κάποιο έκθεμα στο μουσείο του, ο χρήστης πρέπει να έχει εισάγει εξωτερικούς τοίχους πρώτα. Γι' αυτό το λόγο ελέγχουμε το polyline_flag, ώστε να ξέρουμε αν μπορεί ο χρήστης να εισάγει κάποιο έκθεμα. Στη συνέχεια, πρέπει να δοθεί στον χρήστη η δυνατότητα επιλογής τοποθέτησης του εκθέματος σε συγκεκριμένο σημείο, που επιθυμεί, με τη χρήση του ποντικιού. Αυτό επιτυγχάνεται με τη χρήση ενός eventListener στο viewbox, ο οποίος περιμένει τον χρήστη να

κλικάρει σε κάποιο σημείο του SVG space. Αφού γίνει αυτό, δημιουργούμε ένα στοιχείο SVG τύπου εικόνας και αποθηκεύουμε σε μία μεταβλητή, με όνομα start, τις συντεταγμένες που κλίκαρε ο χρήστης. Έπειτα, ανάλογα με τον τύπο του εκθέματος που επέλεξε νωρίτερα, προσθέτουμε στο SVG αντικείμενο τα αντίστοιχα χαρακτηριστικά για το ύψος και το πλάτος του, καθώς και την αντίστοιχη εικόνα.

```
27.             var title=document.createElementNS(svgNS,"title");
28.             var titletext  = document.createTextNode("Exhibit "+numImages);
29.             title.appendChild(titletext);
30.             myImage.appendChild(title);
31.             myImage.setAttributeNS(null,"id","img"+numImages);
32.             myImage.setAttributeNS(null,"x",start.x-
    (((parseInt(myImage.getAttributeNS(null,"width"))/100)*100)/2));
33.             myImage.setAttributeNS(null,"y",start.y-
    (((parseInt(myImage.getAttributeNS(null,"height"))/100)*50)/2));
34.             myImage.setAttribute("class","draggable confine");
35.             document.getElementById("museum").appendChild(myImage);
36.             museum.removeEventListener("click",_listener,true);
37.         },true);
38.     }
39.     else{
40.         alert("You haven't create a museum yet. Press the big box button in order t
    o continue.");
41.     }
42. }
```

source code 11

Αμέσως μετά αφού εισάγουμε και ένα τίτλο για το έκθεμα (π.χ. Exhibit 1), προσθέτουμε τα x και y χαρακτηριστικά του. Σε αυτό το σημείο χρησιμοποιούμε τις συντεταγμένες του ποντικιού και το ύψος, πλάτος του SVG αντικειμένου, ώστε το σημείο, που κλίκαρε ο χρήστης να αντιστοιχεί στο κέντρο της εικόνας. Στη συνέχεια, προσθέτουμε στο SVG αντικείμενο τις κλάσεις draggable και confine. Η draggable μας επιτρέπει τη μετακίνηση του αντικειμένου, ενώ τη confine τη χρησιμοποιούμε για να μη μετακινείται το έκθεμα πέρα από τους εξωτερικούς τοίχους του μουσείου. Τέλος, προσθέτουμε το SVG αντικείμενο στο μουσείο μας, δηλαδή στο SVG space.

4.2.7 Εξωτερικοί τοίχοι

Παρακάτω παρατηρούμε τον κώδικα, που πραγματοποιεί την εισαγωγή των εξωτερικών τοίχων του μουσείου μας. Ιδιαίτερη προσοχή πρέπει να δοθεί στις μεταβλητές box και polyline_flag. Η box έχει την τιμή true, μόλις ξεκινάει η εφαρμογή, και όταν ο χρήστης εισάγει τους εξωτερικούς τοίχους παίρνει την τιμή false. Αυτό μας βοηθάει ώστε να μπορεί ο χρήστης να εισάγει μόνο μία φορά τους εξωτερικούς τοίχους. Επίσης η μεταβλητή polyline_flag, χρησιμοποιείται για να ενημερώσει άλλες συναρτήσεις, όπως την createImage, που είδαμε παραπάνω, ότι έχουν προστεθεί οι εξωτερικοί τοίχοι του μουσείου.

```
1.  //onclick function of the 3rd button(big box)
2.  document.getElementById("btn3").addEventListener("click", function(){
3.      if(box){
4.          box=false;
5.          polyline_flag=true;
6.          var myPolyline = document.createElementNS(svgNS,"polyline");
7.          myPolyline.setAttributeNS(null,"points","3,3 3,47 97,47 97,3 3,3");
8.          myPolyline.setAttributeNS(null,"fill","transparent");
9.          myPolyline.setAttributeNS(null,"stroke","black");
10.         museum.appendChild(myPolyline);
```

```
    }
    else{
        alert("You can only add one box");
    }
});
```

source code 12

Για τη δημιουργία των εξωτερικών τοίχων χρησιμοποιούμε το SVG στοιχείο polyline. Αυτό το στοιχείο χρησιμοποιείται για να δημιουργήσουμε οποιοδήποτε σχήμα αποτελείται από μόνο ευθείες γραμμές. Στη συνέχεια προσθέτουμε το χαρακτηριστικό points, το οποίο υποδηλώνει μία λίστα από x και y συντεταγμένες, οι οποίες είναι απαραίτητες για τη δημιουργία του. Τέλος, αφού προσθέσουμε κάποια ακόμα χαρακτηριστικά, προσθέτουμε το polyline στο μουσείο (SVG space).

4.2.8 Εσωτερικοί τοίχοι

Παρακάτω παρουσιάζεται ο κώδικας με τον οποίο μπορεί ο χρήστης να εισάγει εσωτερικούς τοίχους στο μουσείο του. Μόλις ο χρήστης πατήσει το κουμπί ΤΟΙΧΟΣ από το μενού, εκτελείται η συνάρτηση createLine(), η οποία θα αναλυθεί παρακάτω.

```
//onclick function of the 4th button(wall)
document.getElementById("btn4").addEventListener("click", createLine);

function createLine(){
    if(polyline_flag){

        if(!buttonPressed){
            buttonPressed=true;
            button4Clicked=true;
            //on mousedown start creating the line
            museum.addEventListener("mousedown",(event) =>{

                if(button4Clicked && !button2Clicked){
                    button4Clicked=false;
                    var line =document.createElementNS(svgNS,"line");
                    var start= museumPoint(museum,event.clientX,event.clientY);
                    const drawLine= (e) =>{

                        let p= museumPoint(museum,e.clientX,e.clientY);
                        line.setAttributeNS(null, 'x1', start.x);
                        line.setAttributeNS(null, 'y1', start.y);
                        line.setAttributeNS(null, 'x2', p.x);
                        line.setAttributeNS(null, 'y2', p.y);
                        line.setAttributeNS(null,"stroke","black");
                        line.setAttributeNS(null,"id","wall"+numWalls);
                        line.setAttribute("class","draggable");
                        document.getElementById("museum").appendChild(line);

                    }
                    //stop drawing when the mouse is up
                    const endDrawLine = (e) => {
                        museum.removeEventListener('mousemove', drawLine);
                        museum.removeEventListener('mouseup', endDrawLine);

                    }
                    museum.addEventListener('mousemove', drawLine);
                    museum.addEventListener('mouseup', endDrawLine);
                    buttonPressed=false;
                }

            });
```

```
        }
        else{
            alert("You have already pressed a button");
        }
        numWalls++;
    }
    else{
            alert("You haven't create a museum yet. Press the big box button in ord
er to continue.");
    }

}
```

source code 13

Αφού η εφαρμογή ελέγξει αν ο χρήστης έχει εισάγει εξωτερικούς τοίχους στο μουσείο, προσθέτει έναν event Listener στο viewbox, ο οποίος περιμένει από το χρήστη να κλικάρει με το ποντίκι του κάπου μέσα σε αυτό. Στη συνέχεια, δημιουργούμε ένα SVG στοιχείο τύπου line και αποθηκεύουμε σε μία μεταβλητή, με όνομα start, τις συντεταγμένες, που κλίκαρε ο χρήστης. Έπειτα, για όσο χρόνο ο χρήστης μετακινεί το ποντίκι του μέσα στο viewbox, εκτελείται η συνάρτηση drawLine. Με αυτή τη συνάρτηση και χρησιμοποιώντας τη μεταβλητή start, καθώς και μία ακόμα μεταβλητή (p), στην οποία αποθηκεύονται οι τελικές συντεταγμένες του τοίχου, ενημερώνουμε τα αντίστοιχα χαρακτηριστικά του SVG αντικειμένου, δηλαδή τα (x1,y1), (x2,y2). Επίσης, προσθέτουμε στο αντικείμενο και την κλάση draggable, ώστε να είναι δυνατή η μετακίνησή του, αφού ολοκληρωθεί η κατασκευή του. Τέλος προσθέτουμε το SVG αντικείμενο μέσα στο μουσείο μας.

4.2.9 Δημιουργία πόρτας

Παρακάτω μπορούμε να παρατηρήσουμε τη συνάρτηση createDoor, η οποία επιτρέπει στον χρήστη να εισάγει κάποια πόρτα στο μουσείο του. Μόλις ο χρήστης πατήσει το κουμπί **ΠΟΡΤΑ** από το μενού, εμφανίζονται 2 ακόμα επιλογές. Έτσι μπορεί να αποφασίσει αν θέλει να εισάγει μία οριζόντια ή κάθετη πόρτα. Αυτή η επιλογή του, μεταφέρεται στη συνάρτηση με τη χρήση του ορίσματος type. Σε αυτό το σημείο, θα ήταν καλό να αναφέρουμε κάποιες σημαντικές πληροφορίες για τις πόρτες του μουσείου. Ο χρήστης μπορεί να εισάγει 1 ή 2 πόρτες στο μουσείο του. Η πρώτη πόρτα, που θα εισάγει θα είναι πάντα η είσοδος. Αν εισάγει και δεύτερη πόρτα τότε αυτή θα θεωρείται η έξοδος του μουσείου. Από την άλλη πλευρά, αν ο χρήστης αποθηκεύσει το μουσείο του ενώ υπάρχει μόνο μία πόρτα σε αυτό, τότε αυτή μετατρέπεται αυτόματα σε είσοδο/έξοδο. Επίσης, όπως εξηγήσαμε και παραπάνω, για να μπορέσει ο χρήστης να εισάγει κάποια πόρτα, πρέπει να έχει προσθέσει πρώτα τους εξωτερικούς τοίχους του μουσείου του.

```
//onclick function of the door button. Creates a red door type is used to determine
the direction of the door
function createDoor(type){
    if(polyline_flag){
        if(!buttonPressed){
            if(!door_flag_entrance || !door_flag_exit ){
            let acceptable_door=true;
            buttonPressed=true;
            museum.addEventListener("click",function _listener(event){

        var line =document.createElementNS(svgNS,"line");
                var start= museumPoint(museum,event.clientX,event.clientY);


                    if(checkCoords(start.x,start.y)){
```

```
16.
17.
18.                 if(type=="horizontal"){
19.                     if(checkCoords(start.x-
    5,start.y) &&  checkCoords(start.x+5,start.y)){
20.                     line.setAttributeNS(null, 'x1', start.x-5);
21.                     line.setAttributeNS(null, 'y1', start.y);
22.                     line.setAttributeNS(null, 'x2', start.x+5);
23.                     line.setAttributeNS(null, 'y2', start.y);
24.                     acceptable_door=false;
25.                     }
26.                     else{
27.                         alert("You can't create a horizontal door here");
28.
29.                     }
30.                 }
31.                 else if(type=="vertical"){
32.                     if(checkCoords(start.x,start.y-
    5) &&  checkCoords(start.x,start.y+5)){
33.                     line.setAttributeNS(null, 'x1', start.x);
34.                     line.setAttributeNS(null, 'y1', start.y-5);
35.                     line.setAttributeNS(null, 'x2', start.x);
36.                     line.setAttributeNS(null, 'y2', start.y+5);
37.                     acceptable_door=false;
38.                     }
39.                     else{
40.                         alert("You can't create a vertical door here");
41.                     }
42.                 }
43.                 else{
44.                     alert("Unexpected error! Door type is NOT established!");
45.                 }
```

source code 14

Αρχικά ελέγχουμε αν ο χρήστης έχει ήδη εισάγει 2 πόρτες, δηλαδή αν το μουσείο του περιέχει και πόρτα εισόδου και πόρτα εξόδου. Αν λοιπόν, δεν υπάρχει κάποια από τις 2, τότε προσθέτουμε στο viewbox έναν event listener, ο οποίος περιμένει από τον χρήστη να κλικάρει με το ποντίκι του σε κάποιο σημείο αυτού. Στη συνέχεια, όπως συνέβη και στην εισαγωγή εσωτερικού τοίχου (κεφ. 3.2.8), δημιουργούμε ένα SVG στοιχείο τύπου line και αποθηκεύουμε σε μια μεταβλητή, τις συντεταγμένες, στις οποίες κλίκαρε ο χρήστης. Στη συνέχεια, ανάλογα με τον τύπο της πόρτας, πραγματοποιούμε κάποιους ελέγχους, ώστε να είμαστε σίγουροι ότι μπορεί να τοποθετηθεί μία πόρτα στο σημείο, που κλίκαρε ο χρήστης.

Απαραίτητες προϋποθέσεις για την εισαγωγή μίας πόρτας είναι ο χρήστης να έχει κλικάρει σε κάποιο σημείο, όπου υπάρχει εξωτερικός τοίχος. Γι' αυτό το λόγο χρησιμοποιούμε τη συνάρτηση checkCoords(), η οποία δέχεται ως ορίσματα τις συντεταγμένες, που κλίκαρε ο χρήστης, και αν αυτές βρίσκονται πάνω στο SVG αντικείμενο polyline (εξωτερικοί τοίχοι), τότε επιστρέφει true. Σε αντίθετη περίπτωση επιστρέφει την τιμή false.

```
1.  //check if this set of coords belongs to a polyline(museum)
2.  function checkCoords(x,y){
3.      var returnVal=false;
4.      var element=museum.getElementsByTagNameNS(svgNS,"polyline");
5.      let point = museum.createSVGPoint();
6.      point.x=x;
7.      point.y=y;
8.
9.      returnVal=returnVal || element[0].isPointInStroke(point);
10.     return returnVal;
```

```
11.
12. }
```

source code 15

```
46.                 if(!acceptable_door){
47.                 var title=document.createElementNS(svgNS,"title");
48.                 if(!door_flag_entrance){
49.                     door_flag_entrance=true;
50.                     var titletext  = document.createTextNode("Entrance");
51.                 }
52.                 else{
53.                     door_flag_exit=true;
54.                     var titletext  = document.createTextNode("Exit");
55.
56.                 }
57.                 title.appendChild(titletext);
58.                 line.appendChild(title);
59.                 line.setAttributeNS(null,"stroke","red");
60.                 line.setAttributeNS(null,"id","door");
61.
62.                 document.getElementById("museum").appendChild(line);
63.                 console.log(museum);
64.                 museum.removeEventListener("click",_listener,true);
65.
66.
67.                 buttonPressed=false;
68.                 }
69.                 }
70.
71.
72.             },true);
73.             }
74.             else{
75.                 alert("2 doors are allowed at maximum");
76.             }
77.         }
78.         else{
79.             alert("You have already pressed a button");
80.         }
81.
82.     }
83.     else{
84.         alert("You haven't create a museum yet. Press the big box button in order t
    o continue.");
85.     }
86. }
```

source code 16

Η πόρτα, που θα δημιουργηθεί θα έχει συγκεκριμένες διαστάσεις. Το σημείο, που επέλεξε ο χρήστης θα είναι το κέντρο της πόρτας και ανάλογα με τον τύπο της θα εκτείνεται είτε στον x είτε στον y άξονα. Γι' αυτό το λόγο, πρέπει να ελέγξουμε εκτός από τις αρχικές συντεταγμένες και τις τελικές, ώστε να είμαστε σίγουροι ότι όλη η πόρτα βρίσκεται στον εξωτερικό τοίχο. Αν ισχύουν όλες αυτές οι προϋποθέσεις, τότε προσθέτουμε και έναν τίτλο, ο οποίος υποδηλώνει το αν η πόρτα είναι εισόδου ή εξόδου. Τέλος προσθέτουμε το SVG αντικείμενο (πόρτα) στο μουσείο μας, μέσω της μεθόδου appendChild.

4.2.10 Διαγραφή αντικειμένου

Παρακάτω μπορούμε να παρατηρήσουμε τη συνάρτηση removeObject(), η οποία είναι υπεύθυνη για τη διαγραφή ενός αντικειμένου, που υπάρχει στο μουσείο. Το σημαντικότερο ζήτημα σε αυτή τη συνάρτηση ήταν να καταλάβουμε τι είδους είναι το αντικείμενο, που θέλει να διαγράψει ο χρήστης, ώστε να πραγματοποιήσουμε τις κατάλληλες αλλαγές στις μεταβλητές, που σχετίζονται με αντικείμενα αυτού του είδους. Για παράδειγμα, αν ο χρήστης διαγράψει μια εικόνα, πρέπει να μειώσουμε την μεταβλητή numImages, η οποία αποθηκεύει τον συνολικό αριθμό εικόνων κατά 1.

```
1.  //remove an object of your choice
2.  document.getElementById("btn5").addEventListener("click",removeObject);
3.
4.  function removeObject(){
5.      if(!buttonPressed){
6.          buttonPressed=true;
7.          museum.addEventListener("click",function _listener(event){
8.              selectedElement=event.target;
9.              if(selectedElement!=document.getElementById("museum")){
10.                 if(selectedElement.tagName.toLowerCase()=="polyline"){
11.
12.                         return;
13.
14.           }
```

source code 17

Αρχικά, για να διαγράψει κάποιο αντικείμενο ο χρήστης πρέπει να κλικάρει πάνω σε αυτό. Γι' αυτό το λόγο προσθέσαμε έναν eventListener στο viewbox, το οποίο περιμένει αυτή την ενέργεια από τον χρήστη. Μόλις ο χρήστης κλικάρει κάποιο αντικείμενο, αποθηκεύουμε τα χαρακτηριστικά αυτού στη μεταβλητή selectedElement. Αμέσως μετά κάνουμε διάφορους ελέγχους για να διαπιστώσουμε το είδος του αντικειμένου. Αν ο χρήστης κλικάρει σε σημείο, που δεν υπάρχει κάποιο αντικείμενο, τότε το selectedElement γίνεται ίσο με το viewbox, οπότε τον ενημερώνουμε ότι η διαγραφή δεν είναι δυνατό να πραγματοποιηθεί. Επίσης, αν κλικάρει τους εξωτερικούς τοίχους του μουσείου του, τότε η διαγραφή δε θα πραγματοποιηθεί, διότι όπως έχουμε αναφέρει οι εξωτερικοί τοίχοι βρίσκονται σε συγκεκριμένες συντεταγμένες, οπότε δεν υπάρχει λόγος διαγραφής τους.

```
15.                 else if(selectedElement.getAttributeNS(null,"id")=="door"){
16.                     if(selectedElement.firstChild.textContent=="Entrance/Exit"){
17.                         door_flag_entrance=false;
18.                         door_flag_exit=false;
19.                     }
20.                     else if(selectedElement.firstChild.textContent=="Entrance"){
21.                         door_flag_entrance=false;
22.                     }
23.                     else if(selectedElement.firstChild.textContent=="Exit"){
24.                         door_flag_exit=false;
25.                     }
26.                 }
27.                 var object = selectedElement;
28.                 var parent = object.parentNode;
29.                 parent.removeChild(object);
30.
31.                 if(selectedElement.tagName.toLowerCase()=="image"){
32.                     var images=document.getElementsByTagNameNS(svgNS,"image");
33.                     numImages=images.length;
34.                     for(var i=0;i<images.length;i++){
35.                         images[i].firstChild.textContent="Exhibit"+(i+1);
36.                     }
37.                 }
38.             }
```

```
            else{
                alert("You can't delete this!");
            }
            museum.removeEventListener("click",_listener,true);

            buttonPressed=false;
        },true);
    }
    else{
        alert("You have already pressed a button");
    }
}
```

source code 18

Από την άλλη πλευρά, αν το αντικείμενο είναι πόρτα, τότε ελέγχουμε το είδος της, δηλαδή αν είναι είσοδος, έξοδος ή και τα δύο, και αλλάζουμε τις τιμές των ανάλογων μεταβλητών. Στη συνέχεια, πρέπει να πραγματοποιήσουμε τη διαγραφή του αντικείμενου, που επέλεξε ο χρήστης. Για να γίνει αυτό, πρέπει αρχικά να βρούμε τον γονικό κόμβο αυτού (parent). Έπειτα, η διαγραφή πραγματοποιείται χρησιμοποιώντας τη μέθοδο removeChild του parent. Τέλος, πρέπει να γίνει ακόμα ένας έλεγχος, σε περίπτωση που το αντικείμενο ήταν τύπου εικόνας. Αν ισχύει αυτό, τότε αλλάζουμε τον τίτλο κάθε εικόνας, ώστε να υπάρχει συνέχεια στην αρίθμηση των εκθεμάτων.

4.2.11 Αποθήκευση μουσείου

Παρακάτω παρατηρούμε τη συνάρτηση save_json, μέσω της οποίας αποθηκεύουμε ένα μουσείο, που έχει δημιουργήσει ο χρήστης, σε μορφή json αρχείου. Αυτό το αρχείο αποθηκεύεται στον προσωπικό φάκελο του χρήστη, με ένα συγκεκριμένο όνομα, που θα επιλέξει ο ίδιος. Η κλήση της συνάρτησης μπορεί να περιέχει από 0 έως 2 ορίσματα. Αν ο χρήστης δεν έχει αποθηκεύσει ξανά το μουσείο, τότε καλείται χωρίς ορίσματα. Αν όμως ο χρήστης επιλέξει να αποθηκεύσει ένα μουσείο, που έχει επαναφορτώσει (σελίδα load), τότε καλείται με 2 ορίσματα. Το museum_name αφορά το όνομα με το οποίο αποθηκεύτηκε το μουσείο, ενώ το loaded είναι ένα flag, το οποίο μας ενημερώνει αν η κλήση της συνάρτησης έχει γίνει από τη σελίδα load.

```
1.  //save all the elements inside the viewbox into a json file
2.  //loaded flag is used to determine if the user has already saved the museum once
3.  function save_json(museum_name=null,loaded=false){
4.
5.      if(numImages==0){
6.          alert("You must have at least 1 exhibit in order to save a museum");
7.          return;
8.      }
9.      if(!door_flag_entrance){
10.         alert("You must have at least 1 door in order to save the museum(Entrance/E
    xit Or 1 Entrance And One Exit)");
11.         return;
12.     }
13.     if(!polyline_flag){
14.         alert("You haven't created a museum yet. No external walls ");
15.     }
```

source code 19

Για να μπορέσει ο χρήστης να αποθηκεύσει ένα μουσείο, που έχει δημιουργήσει πρέπει να ισχύουν κάποιες προϋποθέσεις. Αυτές είναι:

- να υπάρχει τουλάχιστον 1 έκθεμα

- να υπάρχει τουλάχιστον η πόρτα εισόδου
- να υπάρχουν οι εξωτερικοί τοίχοι του μουσείου

```
16.     else{
17.         if(!door_flag_exit){
18.             var element=document.getElementById("door");
19.             console.log(element);
20.             element.firstChild.textContent="Entrance/Exit";
21.
22.
23.         }
24.     }
25.
26.     if(!museum_name){
27.         var name=prompt("Please enter the file name","museum");
28.         while(name==""){
29.             name=prompt("Please enter the file name","museum");
30.         }
31.     }
32.     else{
33.         var name=prompt("Please enter the file name",museum_name);
34.         while(name==""){
35.             name=prompt("Please enter the file name",museum_name);
36.         }
37.
38.     }
39.     if(name==null){
40.         return;
41.     }
```

source code 20

Έπειτα, ελέγχουμε αν υπάρχει πόρτα εξόδου στο μουσείο. Αν δεν υπάρχει, μετατρέπουμε την πόρτα εισόδου σε είσοδο/έξοδο. Στη συνέχεια η εφαρμογή ζητά από τον χρήστη ένα όνομα για το μουσείο του με τη χρήση ενός popup παράθυρου. Το default όνομα, που εμφανίζεται σε αυτό είναι είτε museum (αν το μουσείο δεν έχει αποθηκευτεί ξανά) είτε το ήδη υπάρχον όνομα του μουσείου, το οποίο είναι αποθηκευμένο στη μεταβλητή museum_name.

```
42.     if(!loaded){
43.   event.preventDefault(event);
44.   $.ajax({
45.       url: '../php_files/test.php',
46.       type: 'POST',
47.       data:{
48.           'name':name,
49.       },
50.       success: function (response) {
51.
52.       //get response from your php page (what you echo or print)
53.         if(response =='True'){
54.             alert(name +" already exists");
55.             return;
56.         }
```

source code 21

Στη συνέχεια, αν το μουσείο αποθηκεύεται για πρώτη φορά, τότε πρέπει να ελέγξουμε αν υπάρχει ήδη κάποιο άλλο μουσείο στο λογαριασμό του χρήστη, το οποίο έχει το ίδιο όνομα. Αυτό γίνεται με τη χρήση AJAX. Ουσιαστικά, η εφαρμογή στέλνει το όνομα του μουσείου, στο php αρχείο test.php,

το οποίο ελέγχει αν υπάρχει κάποιο αποθηκευμένο μουσείο από τον ίδιο χρήστη με αυτό το όνομα. Αν υπάρχει, εμφανίζεται κατάλληλο μήνυμα και η αποθήκευση ακυρώνεται.

```
57.         else{
58.             var museum = document.getElementById("museum");
59.             tempDiv = document.createElement("div");
60.             tempDiv=museum.cloneNode(true);
61.             var svgText = tempDiv.innerHTML;
62.             jsonString = JSON.stringify(svgText);
63.             console.log(museum);
64.             var xhr = new XMLHttpRequest();
65.
66.             xhr.open("POST","save_json.php",true);
67.             xhr.setRequestHeader("Content-type","application/x-www-form-
    urlencoded");
68.             var data=''+ "json=" + jsonString + "&name="+name +"&loaded=" + loaded;

69.             xhr.send(data);
70.             alert(name + " saved successfully!!")
71.
72.         }
73.       }
74.     });
75.     }
```

source code 22

Αν όμως δεν υπάρχει, τότε δημιουργούμε ένα αντίγραφο του μουσείου (SVG space) και το αποθηκεύουμε σε μία μεταβλητή με όνομα tempDiv. Έπειτα, χρησιμοποιώντας τη Javascript συνάρτηση JSON.stringify() μετατρέπουμε σε string όλο το HTML περιεχόμενο της παραπάνω μεταβλητής. Αμέσως μετά δημιουργούμε ένα καινούριο XMLHttpRequest, το οποίο είναι ένα αντικείμενο που μας επιτρέπει να ζητήσουμε δεδομένα από έναν server. Τα κυριότερα πλεονεκτήματά του είναι ότι μπορεί να ενημερώσει τη σελίδα, χωρίς να γίνει επαναφόρτωσή της και να στέλνει δεδομένα σε ένα server, στο παρασκήνιο. Αυτό το αντικείμενο στέλνει στην php σελίδα, save_json.php, τα ακόλουθα δεδομένα:

- Το string σε μορφή JSON, που δημιουργήσαμε παραπάνω
- Το όνομα του μουσείου
- Το flag loaded

Σε αυτό το php αρχείο, εισαγάγουμε το δωμάτιο μέσα στη βάση δεδομένων μας, στον πίνακα user_rooms και στη συνέχεια δημιουργούμε ένα νέο json αρχείο στον προσωπικό φάκελο του χρήστη, μέσα στο οποίο αποθηκεύουμε το string, που στείλαμε παραπάνω. Τέλος, ενημερώνουμε το χρήστη ότι το μουσείο του αποθηκεύτηκε επιτυχώς.

```
76.     else{
77.         if(museum_name!=name){
78.             event.preventDefault(event);
79.             $.ajax({
80.               url: '../php_files/test.php',
81.               type: 'POST',
82.               data:{
83.                   'name':name,
84.               },
85.               success: function (response) {
86.               //get response from your php page (what you echo or print)
87.                 if(response =='True'){
88.                     alert(name +" already exists");
89.                     return;
```

```
                }
                else{
                    var museum = document.getElementById("museum");
                    tempDiv = document.createElement("div");
                    tempDiv=museum.cloneNode(true);
                    var svgText = tempDiv.innerHTML;
                    jsonString = JSON.stringify(svgText);
                    console.log(museum);
                    var xhr = new XMLHttpRequest();

                            xhr.open("POST","save_json.php",true);
                            xhr.setRequestHeader("Content-type","application/x-www-form-urlencoded");
                            var data=''+ "json=" + jsonString + "&name="+name +"&loaded=" + loaded;
                            xhr.send(data);
                            alert(name + " saved successfully!!")
                        }

                    }
                });
            }
            else{
                var museum = document.getElementById("museum");
                tempDiv = document.createElement("div");
                tempDiv=museum.cloneNode(true);
                var svgText = tempDiv.innerHTML;
                jsonString = JSON.stringify(svgText);
                console.log(museum);
                var xhr = new XMLHttpRequest();

                xhr.open("POST","save_json.php",true);
                xhr.setRequestHeader("Content-type","application/x-www-form-urlencoded");
                var data=''+ "json=" + jsonString + "&name="+name +"&loaded=" + loaded;
                xhr.send(data);

                alert(name + " saved successfully!!")
            }


        }


        }

```

source code 23

Σε περίπτωση, που το μουσείο έχει αποθηκευθεί τουλάχιστον μία φορά, τότε πρέπει να ελέγξουμε αν ο χρήστης επέλεξε να το αποθηκεύσει με το ίδιο όνομα ή με διαφορετικό. Αν επιλέξει διαφορετικό όνομα, τότε εφαρμόζουμε την ίδια διαδικασία, που αναλύσαμε και παραπάνω. Σε αντίθετη περίπτωση, δεν εκτελούμε τον έλεγχο για την ύπαρξη κάποιου μουσείου στον λογαριασμό του χρήστη, με το ίδιο όνομα, αλλά μόνο την υπόλοιπη διαδικασία, δηλαδή μετατροπή του SVG space σε string μορφής JSON και αποθήκευση του μουσείου μέσω του XMLHttpRequest.

4.2.12 Επαναφόρτωση μουσείου

Παρακάτω μπορούμε να παρατηρήσουμε τη συνάρτηση load, μέσω της οποίας η εφαρμογή μπορεί να επαναφορτώσει ένα συγκεκριμένο μουσείο από αυτά, που έχει δημιουργήσει ήδη στο λογαριασμό του. Η κλήση της συνάρτησης πρέπει να περιέχει δύο ορίσματα. Το πρώτο όρισμα (name), υποδηλώνει το username του χρήστη, που θέλει να επαναφορτώσει ένα μουσείο του. Από την άλλη πλευρά,το δεύτερο όρισμα (filename), αφορά το όνομα του μουσείου που επιλέγει ο χρήστης για επαναφόρτωση.

```
1.  function load(name,filename){
2.      var success = false;
3.
4.      $.getJSON("../json/"+name+"/"+ filename+".json",function(data){
5.          success=true;
6.          museum.innerHTML=data;
7.          FindTotalImages();
8.
9.      });
10.
11.     setTimeout(function() {
12.         if (!success)
13.         {
14.             // Handle error accordingly
15.             alert("There isn't any saved template");
16.             window.location.href="../php_files/welcome.php";
17.         }
18.     },1000);
19.
20. }
```

source code 24

Αρχικά χρησιμοποιούμε τη μέθοδο της JQuery getJSON, η οποία μας επιτρέπει να πάρουμε JSON δεδομένα με τη χρήση ενός AJAX HTTP GET request. Στη συγκεκριμένη περίπτωση, δίνουμε στη μέθοδο, το ακριβές μονοπάτι στο οποίο βρίσκονται τα JSON δεδομένα του μουσείου, που επέλεξε ο χρήστης. Αυτό επιτυγχάνεται μέσω των 2 ορισμάτων, που αναλύσαμε παραπάνω. Αυτά τα δεδομένα δίνονται στο χαρακτηριστικό innerHTML του viewbox, δηλαδή στο HTML περιεχόμενο αυτού. Επίσης για την περαιτέρω επεξεργασία του μουσείου από το χρήστη είναι απαραίτητο η εφαρμογή να γνωρίζει τον συνολικό αριθμό εκθεμάτων, που υπάρχουν στο μουσείο. Αυτό επιτυγχάνεται με τη συνάρτηση FindTotalImages(). Τέλος, αν η μέθοδος getJSON δεν εντοπίσει το συγκεκριμένο αρχείο, τότε μετά από ένα χρονικό περιθώριο, η εφαρμογή εμφανίζει στον χρήστη ανάλογο μήνυμα, ότι δηλαδή δεν υπάρχει αποθηκευμένο δωμάτιο με τα δοθέντα στοιχεία και επιστρέφει το χρήστη στην αρχική σελίδα.

4.2.13 Animation

Παρακάτω μπορούμε να παρατηρήσουμε τη συνάρτηση createAnimation, μέσω της οποίας πραγματοποιείται η κίνηση των επισκεπτών στο μουσείο. Η συγκεκριμένη συνάρτηση δέχεται σαν ορίσματα 3 μεταβλητές:

- array → ένας πίνακας, που αντιπροσωπεύει το μονοπάτι, που εκτελεί κάθε χρήστης
- peopleNum → ο αριθμός των ατόμων, που εισέρχονται στο μουσείο την ίδια χρονική στιγμή
- visitor_category → η ιδιότητα του επισκέπτη (οικογένεια, σχολείο, άλλοι επισκέπτες)

```
1.  //creates a route based on parameters given by the user
```

```
2.  //array represents the path that each user takes
3.  //peopleNum represents the number of people that enter the museum at the same time
    and have the same path
4.  function createAnimation(array,peopleNum,visitor_category){
5.      var point=museum.createSVGPoint();
6.      var path="";
7.      var time;
8.      point=findDoor();
9.      var myImage=[];
10.     let svg = document.getElementById("museum");
11.     if(first_time){
12.         if(svg.getCurrentTime() > 0)
13.             time=svg.getCurrentTime();
14.
15.     }
16.
17.     else{
18.         svg.setCurrentTime(0);
19.         time=0;
20.     }
21.     first_time=1;
```

source code 25

Αρχικά, δημιουργούμε ένα SVG point, στο οποίο θα δώσουμε ως τιμή τις συντεταγμένες της πόρτας εισόδου, μέσω της συνάρτησης findDoor(), η οποία θα αναλυθεί αργότερα . Στη συνέχεια, για να υπάρχει η δυνατότητα επαναλαμβανόμενης προσομοίωσης, χρησιμοποιούμε το χαρακτηριστικό time του SVG, καθώς και τις μεθόδους getCurrentTime και setCurrentTime. Αν η προσομοίωση εκτελείται για πρώτη φορά, τότε θέτουμε σε μία καινούρια μεταβλητή time, η οποία θα μας χρησιμεύσει αργότερα, την τιμή 0. Σε αντίθετη περίπτωση, θέτουμε ως τιμή την τρέχουσα τιμή του χαρακτηριστικού time.

```
22.         //create people objects(shown as circles)
23.     for( var i=0;i<peopleNum;i++){
24.             myImage[i] = document.createElementNS(svgNS,"circle");
25.
26.             myImage[i].setAttributeNS(null,"cx",point.x);
27.             myImage[i].setAttributeNS(null,"cy",point.y);
28.             myImage[i].setAttributeNS(null,"r","0.3");
29.             myImage[i].setAttributeNS(null,"opacity","1");
30.             myImage[i].setAttribute("class","confine");
31.             myImage[i].setAttributeNS(null,"fill","none");
32.             switch(parseInt(visitor_category)){
33.                 case 1:
34.                         myImage[i].setAttributeNS(null,"stroke","green");
35.                         break;
36.
37.                 case 2:
38.                         myImage[i].setAttributeNS(null,"stroke","red");
39.                         break;
40.
41.                 case 3:
42.                         myImage[i].setAttributeNS(null,"stroke","blue");
43.                         break;
44.
45.             }
46.
47.             myImage[i].setAttributeNS(null,"stroke-width","1");
48.             var title=document.createElementNS(svgNS,"title");
49.             var titletext  = document.createTextNode("Visitor "+i);
50.             title.appendChild(titletext);
51.             myImage[i].appendChild(title);
```

source code 26

Στη συνέχεια, πρέπει να δημιουργήσουμε τα αντικείμενα που θα αντιπροσωπεύουν τους επισκέπτες του μουσείου, τα οποία θα έχουν τη μορφή κύκλου. Έτσι εκτελούμε έναν επαναληπτικό βρόχο ανάλογα με την τιμή του ορίσματος peopleNum. Έτσι δημιουργούμε SVG στοιχεία τύπου circle με αρχική θέση τις συντεταγμένες της πόρτας εισόδου, που βρήκαμε μέσω της συνάρτησης findDoor(). Επίσης, επιλέξαμε να έχουμε διαφορετικό χρώμα για κάθε τύπο επισκέπτη, ώστε να είναι δυνατός ο διαχωρισμός τους. Τέλος, προσθέτουμε και έναν τίτλο για κάθε επισκέπτη, στο οποίο φαίνεται ο αύξων αριθμός του, με βάση την ομάδα, που εισήλθε στο μουσείο.

```
52.             if(!first_time_visitor){
53.                 initialize_obstacles();
54.                 first_time_visitor=1;
55.             }
56.             //create new grid for pathfinding.js algorithms
57.             grid=new PF.Grid(100,50,obstacles);
58.             if(path==""){
59.                 path="M ";
60.                 var temp_path=Createpath(array,parseInt(myImage[i].getAttributeNS(n
    ull,"cx")),parseInt(myImage[i].getAttributeNS(null,"cy")));
61.                 if(temp_path==false){
62.                     break;
63.                 }
64.                 path+=temp_path;
65.                 var end=findDoor(true);
66.
67.             }
68.             var end=findDoor(true);
69.
```

source code 27

Έπειτα, αν πρόκειται για την πρώτη προσομοίωση, τότε πρέπει να αρχικοποιήσουμε τον πίνακα obstacles, ο οποίος χρησιμοποιείται ώστε να γνωρίζουμε σε ποια σημεία του μουσείου υπάρχουν εμπόδια (π.χ. εκθέματα, τοίχοι). Αυτό συμβαίνει μέσω της συνάρτησης initialize_obstacles(), που θα αναλυθεί σε επόμενο κεφάλαιο. Σε αυτό το σημείο, πρέπει να αναφερθεί ότι για τη δημιουργία του μονοπατιού κάθε επισκέπτη από έκθεμα σε έκθεμα χρησιμοποιήθηκε η βιβλιοθήκη της javascript PathFinding.js. Απαραίτητη για τη χρήση αυτής της βιβλιοθήκης είναι η δημιουργία ενός grid, το οποίο θα έχει ίδιες διαστάσεις με το μουσείο (100X50). Επίσης χρησιμοποιώντας τον πίνακα obstacles, ενημερώνουμε το grid για το πού ακριβώς υπάρχουν εμπόδια. Στη συνέχεια, πρέπει να δημιουργήσουμε το path, το οποίο θα προσθέσουμε στο SVG αντικείμενο, ώστε να γίνει το animation. Για να θεωρείται ένα path σωστό, πρέπει να ξεκινάει με το χαρακτήρα M και στη συνέχεια να περιέχει διάφορα ζευγάρια συντεταγμένων, τα οποία θα πρέπει να επισκεφτεί. Η δημιουργία αυτού του path γίνεται με τη χρήση της συνάρτησης CreatePath, η οποία θα αναλυθεί σε επόμενο κεφάλαιο. Αφού ο επισκέπτης περάσει από όλα τα εκθέματα που θέλει, πρέπει να κατευθυνθεί προς την έξοδο του μουσείου, ώστε να φύγει από αυτό. Γι' αυτό τον λόγο καλούμε ξανά τη συνάρτηση findDoor(), όμως αυτή τη φορά ψάχνουμε να βρούμε τις συντεταγμένες της πόρτας εξόδου.

```
70.             //create path of movement animation
71.             var mpath=document.createElementNS(svgNS,"path");
72.             mpath.setAttributeNS(null,"d",path);
73.             mpath.setAttributeNS(null,"fill","none");
74.             mpath.setAttributeNS(null,"id","theMotionPath"+path_id+ visitor_categor
    y);
75.
76.             //create movement animation
77.             var ani = document.createElementNS(svgNS,"animateMotion");
```

```
78.             ani.setAttributeNS(null,"dur", "30s");
79.             ani.setAttributeNS(null,"repeatCount", "1");
80.             ani.setAttributeNS(null,"begin",(i+time)+'s');
81.
82.             //create the fade out animation, when the user exits the museum, after
    he finishes his movement
83.             var anim_end=document.createElementNS(svgNS,"animate");
84.             anim_end.setAttributeNS(null,"attributeName","opacity");
85.             anim_end.setAttributeNS(null,"dur","1");
86.             anim_end.setAttributeNS(null,"begin",(i+time+29)+'s');
87.             anim_end.setAttributeNS(null,"fill","freeze");
88.             anim_end.setAttributeNS(null,"from","1");
89.             anim_end.setAttributeNS(null,"to","0");
90.             anim_end.setAttributeNS(null,"repeatCount","0");
```

source code 28

Αμέσως μετά, πρέπει να κατασκευάσουμε τα απαραίτητα SVG στοιχεία, για να πραγματοποιηθεί η κίνηση του επισκέπτη. Αυτά είναι ένα στοιχείο, τύπου path, στο οποίο θα αποθηκευτεί το path, που δημιουργήθηκε παραπάνω και ένα στοιχείο τύπου animateMotion. Αυτό επιλέγουμε να έχει μία συγκεκριμένη διάρκεια (30 δευτερόλεπτα) και να συμβεί μόνο μία φορά. Τέλος ιδιαίτερη προσοχή πρέπει να δοθεί στο χαρακτηριστικό begin, το οποίο υποδηλώνει σε ποιο χρονικό σημείο θα ξεκινήσει η κίνηση. Έχουμε επιλέξει οι επισκέπτες μίας ομάδας, να εισέρχονται στο μουσείο με διαφορά ενός δευτερολέπτου.

Εκτός από αυτά, μπορούμε να παρατηρήσουμε και άλλο ένα SVG στοιχείο, τύπου animate. Αυτό χρησιμοποιείται για να πραγματοποιηθεί ένα εφέ, μέσω του οποίου ο επισκέπτης κάνει fade-out, μόλις φτάσει κοντά στην πόρτα εξόδου του μουσείου. Για να συμβεί αυτό, αλλάζουμε το χαρακτηριστικό opacity του SVG αντικειμένου, από 1 σε 0, ένα δευτερόλεπτο πριν τελειώσει η κίνησή του.

```
91.
92.             var mpathObj=document.createElementNS(svgNS,"mpath");
93.             mpathObj.setAttribute("href","#theMotionPath"+path_id+ visitor_category
    );
94.             ani.appendChild(mpathObj);
95.
96.             myImage[i].appendChild(ani);
97.             myImage[i].appendChild(anim_end);
98.
99.             document.getElementById("museum").appendChild(mpath);
100.           }
101.           //if there wasn't an error append objects in viewbox
102.           if(!stop_movement_error){
103.           for(var i=0;i<peopleNum;i++){
104.               document.getElementById("museum").appendChild(myImage[i]);
105.           }
106.           }
107.           previous_peopleNum=peopleNum;
108.           path_id++;
109.
110.
111.       }
```

source code 29

Στη συνέχεια δημιουργούμε ένα ακόμα SVG στοιχείο τύπου mpath, το οποίο πρέπει να προστεθεί στο animation. Σε αυτό δίνουμε ως αναφορά το χαρακτηριστικό id του path, που δημιουργήσαμε παραπάνω. Έπειτα εισάγουμε όλα τα στοιχεία, στο SVG αντικείμενο τύπου circle, που

αντιπροσωπεύει τον επισκέπτη. Τέλος, ελέγχουμε αν υπήρχε κάποιο λάθος κατά τη διάρκεια δημιουργίας του path και προσθέτουμε τα αντικείμενα μέσα στο μουσείο (SVG space).

4.2.14 Εμπόδια

Παρακάτω μπορούμε να παρατηρήσουμε τη συνάρτηση add_obstacle, μέσω της οποίας πραγματοποιείται η εισαγωγή εμποδίων στο μουσείο. Η συγκεκριμένη συνάρτηση δημιουργεί εμπόδια μόνο για τους εσωτερικούς και εξωτερικούς τοίχους του μουσείου και όχι για τα εκθέματα. Δέχεται σαν ορίσματα 6 μεταβλητές:

- (x1,y1),(x2,y2) → πρόκειται για τις συντεταγμένες αρχής και τέλους ενός τοίχου
- door → ένας πίνακας, που περιέχει όλα τα αντικείμενα τύπου πόρτας
- polyline → ένα flag, το οποίο ενημερώνει τη συνάρτηση, αν οι παραπάνω συντεταγμένες αφορούν εσωτερικό (false) ή εξωτερικό (true) τοίχο

Προτού εξηγήσουμε τη συνάρτηση, πρέπει να αναλύσουμε τον τρόπο με τον οποίο η βιβλιοθήκη PathFinding.js αναγνωρίζει τα εμπόδια. Κάθε στοιχείο του πίνακα obstacles, που δημιουργούμε, μπορεί να έχει 2 πιθανές τιμές. Την τιμή 0 αν στο συγκεκριμένο σημείο δεν υπάρχει κάποιο εμπόδιο ή 1 αν υπάρχει.

```
1.  //make a specific line as obstacle.
2.  //If it's a polyline find where the door is located and mark it as walkable space(i
    n obstacles array)
3.  function add_obstacle(x1,y1,x2,y2,door,polyline=false){
4.      var temp;
5.      if(x1>x2){
6.          temp=x1;
7.          x1=x2;
8.          x2=temp;
9.      }
10.     if(y1>y2){
11.         temp=y1;
12.         y1=y2;
13.         y2=temp;
14.     }
15.
16.     for(k=y1-1;k<=y2+1;k++){
17.         for(l=x1-1;l<=x2+1;l++){
18.             obstacles[k][l]=1;
19.         }
20.
21.     }
```

source code 30

Αρχικά, ελέγχουμε τις συντεταγμένες κάθε τοίχου, ώστε να ξέρουμε ποιο ζευγάρι (x,y) έχει την μεγαλύτερη τιμή. Αυτό συμβαίνει, για να είμαστε σίγουροι ότι τα εμπόδια θα προστεθούν από αριστερά προς τα δεξιά και από πάνω προς τα κάτω. Μετά από αυτή τη διαδικασία εισάγουμε στον πίνακα τα εμπόδια, στα σημεία που υπάρχουν. Να σημειωθεί εδώ ότι σε κάθε τοίχο εισάγουμε και ένα επιπλέον περιθώριο, ώστε να σιγουρέψουμε ότι οι επισκέπτες δεν θα ¨χτυπήσουν¨ πάνω του.

```
22.     if(polyline){
23.         console.log(door);
24.         for(var dor=0;dor<door.length;dor++){
25.         console.log(door[dor]);
26.         var x1=parseInt(door[dor].getAttributeNS(null,"x1"));
27.         var x2=parseInt(door[dor].getAttributeNS(null,"x2"));
```

```
28.         var y1=parseInt(door[dor].getAttributeNS(null,"y1"));
29.         var y2=parseInt(door[dor].getAttributeNS(null,"y2"));
30.         if(x1==x2){
31.             if(x1<10){
32.                 for(var j=y1;j<=y2;j++){
33.                 for(i=0;i<5;i++){
34.                         obstacles[j][x1+i]=0;
35.                 }
36.                 }
37.             }
38.             else{
39.                 for(var j=y1;j<=y2;j++){
40.                 for(i=0;i<5;i++){
41.                         obstacles[j][x1-i]=0;
42.                 }
43.                 }
44.             }
45.         }
46.         else if(y1==y2){
47.             if(y1<10){
48.                 for(var j=x1;j<=x2;j++){
49.                 for(i=0;i<5;i++){
50.                         obstacles[y1+i][j]=0;
51.                 }
52.                 }
53.             }
54.             else{
55.                 for(var j=x1;j<=x2;j++){
56.                 for(i=0;i<5;i++){
57.                         obstacles[y1-i][j]=0;
58.                 }
59.                 }
60.             }
61.
62.             }
63.         }
64.
65.     }
66. }
```

source code 31

Κατά την υλοποίηση αυτής της συνάρτησης, συνειδητοποιήσαμε ότι τα εμπόδια που προστίθενται στους εξωτερικούς τοίχους επικαλύπτουν τις πόρτες εισόδου/εξόδου του μουσείου. Αυτό σημαίνει ότι δεν θα είναι δυνατή η είσοδος και η έξοδος από το μουσείο. Γι' αυτό το λόγο πρέπει να βρούμε όλες τις πόρτες του μουσείου και να αλλάξουμε από τον πίνακα obstacles την προσβασιμότητα στις ανάλογες συντεταγμένες.

Οπότε, για κάθε πόρτα που υπάρχει, ελέγχουμε αν είναι οριζόντια ή κάθετη και αλλάζουμε τις τιμές του πίνακα obstacles στις συντεταγμένες της. Επίσης, ανάλογα με τον τοίχο, που βρίσκεται η πόρτα, δημιουργούμε ένα μονοπάτι, ώστε να είναι ευκολότερη η είσοδος/έξοδος από το μουσείο.

4.2.15 Εκθέματα ως εμπόδια

Παρακάτω μπορούμε να παρατηρήσουμε τη συνάρτηση add_image_as_obstacle(), μέσω της οποίας, πραγματοποιείται η εισαγωγή εμποδίων στο μουσείο μόνο για τα εκθέματα. Στη συγκεκριμένη συνάρτηση μετατρέπουμε κάθε έκθεμα σε «εμπόδιο», ώστε να μη μπορεί κάποιος επισκέπτης να περνάει πάνω από αυτό.

```
1.  //marks the acreage of an exhibit as an obstacle(not-
    walkable). It only excludes it's top left corner
2.  function add_image_as_obstacle(){
3.      var img_obstacles=museum.getElementsByTagNameNS(svgNS,"image");
4.
5.      for(var img=0;img<img_obstacles.length;img++){
6.              console.log(img_obstacles[img].transform);
7.              var image_x=parseInt(img_obstacles[img].getAttributeNS(null,"x"));
8.              var image_y=parseInt(img_obstacles[img].getAttributeNS(null,"y"));
9.              if(img_obstacles[img].transform.baseVal.length!=0){
10.                 transform_x=parseInt(img_obstacles[img].transform.baseVal[0].matrix
    .e);
11.                 transform_y=parseInt(img_obstacles[img].transform.baseVal[0].matrix
    .f);
12.                 image_x+=transform_x;
13.                 image_y+=transform_y;
14.
15.             }
16.             var image_width=parseInt(img_obstacles[img].getAttributeNS(null,"width"
    ));
17.             var image_height=parseInt(img_obstacles[img].getAttributeNS(null,"heigh
    t"));
18.             for( var j=image_x; j<image_x+image_width;j++) {
19.                     for( var k=image_y; k<image_y+image_height;k++){
20.                         obstacles[k][j]=1;
21.
22.                     }
23.             }
24.             obstacles[image_y][image_x]=0;
25.     }
26.
27. }
```

source code 32

Για να συμβεί αυτό, πρέπει να βρούμε αρχικά τις συντεταγμένες κάθε εικόνας. Όμως ένα πρόβλημα, που παρουσιάστηκε κατά τη διάρκεια της υλοποίησης της συνάρτησης, είναι ότι τα εκθέματα μπορεί να έχουν μετακινηθεί από την αρχική τους θέση. Αυτή η μετακίνησή τους αποθηκεύεται, όπως είδαμε και παραπάνω, στο χαρακτηριστικό transform. Έτσι, για να υπολογίσουμε την πραγματική του θέση προσθέτουμε στην αρχική την αντίστοιχη τιμή του transform.

Έπειτα, αναφέραμε ότι θέλουμε να μετατρέψουμε όλο το έκθεμα ως εμπόδιο, αλλά μέχρι στιγμής διαθέτουμε μόνο την πάνω αριστερή γωνία. Γι' αυτό το λόγο, αποθηκεύουμε σε 2 καινούριες μεταβλητές το πλάτος και το ύψος του. Έτσι, το έκθεμα θα εκτείνεται στον άξονα x μέχρι την αρχική του θέση συν το πλάτος του. Ανάλογα, θα εκτείνεται στον άξονα y μέχρι την αρχική του θέση συν το ύψος. Μετατρέπουμε, λοιπόν, όλα τα στοιχεία του πίνακα obstacles, στα οποία βρίσκεται το έκθεμα σε εμπόδια. Όμως πρέπει να μπορεί ο επισκέπτης να δει με κάποιο τρόπο αυτό το έκθεμα, οπότε δίνουμε στην πάνω αριστερά γωνία την τιμή 0, ώστε να μπορεί ο επισκέπτης να «θαυμάσει» το έκθεμα από αυτό το σημείο.

4.2.16 Δημιουργία μονοπατιού

Παρακάτω μπορούμε να παρατηρήσουμε τη συνάρτηση Createpath, μέσω της οποίας πραγματοποιείται η δημιουργία του μονοπατιού, μέσω του οποίου επιτυγχάνεται η κίνηση κάθε επισκέπτη στο μουσείο. Η συγκεκριμένη συνάρτηση δέχεται σαν ορίσματα 3 μεταβλητές:

- Array → η λίστα των εκθεμάτων, που θα περάσει το γκρουπ των επισκεπτών

- x,y → οι συντεταγμένες εκκίνησης του animation, δηλαδή οι συντεταγμένες της πόρτας εισόδου

Για τη δημιουργία του μονοπατιού χρησιμοποιούμε, όπως έχουμε αναφέρει και παραπάνω, τη Javascript βιβλιοθήκη PathFinding.js. Αυτή η βιβλιοθήκη μας επιτρέπει να βρούμε το συντομότερο μονοπάτι ανάμεσα σε 2 σημεία, σε ένα grid με εμπόδια. Αυτό γίνεται επιλέγοντας έναν από τους 10 γνωστούς αλγορίθμους εύρεσης μονοπατιού, που περιέχει (π.χ. A*, BFS, Dijkstra)

```
1.  // in this function we create the animation path of an object (with pathfinding.js)
2.  function Createpath(array,x,y){
3.      var return_path="";
4.      var random_num;
5.      var finder;
6.      var path1=[];
7.      var point_x=0;
8.      var point_y=0;
9.      var prev_x=x;
10.     var prev_y=y;
11.     var home_x=prev_x;
12.     var home_y=prev_y;
13.     var end;
14.     var img=museum.getElementsByTagNameNS(svgNS,"image");
15.     console.log(img);
16.     random_num=Math.floor(Math.random() * 3);  // random numbers 0-2
17.     switch(random_num){
18.         case 0:
19.             finder = new PF.DijkstraFinder({
20.                 allowDiagonal: true,
21.                 dontCrossCorners: true
22.             });
23.
24.             break;
25.         case 1:
26.             finder = new PF.AStarFinder({
27.                 allowDiagonal: true,
28.                 dontCrossCorners: true
29.             });
30.
31.             break;
32.         case 2:
33.             finder=new PF.BreadthFirstFinder({
34.                 allowDiagonal: true,
35.                 dontCrossCorners: true
36.             });
37.             break;
38.
39.
40.
41.     }
```

source code 33

Αρχικά, για να υπάρχει κάποια τυχαιότητα στις κινήσεις των επισκεπτών έχουμε επιλέξει 3 από τους 10 προαναφερθέντες αλγορίθμους και η εφαρμογή επιλέγει τυχαία έναν από αυτούς. Σε κάθε αλγόριθμο προσθέτουμε 2 επιπλέον χαρακτηριστικά:

- allowDiagonal →επιτρέπει τη διαγώνια κίνηση μέσα στο grid
- dontCrossCorners → δεν επιτρέπει τη διέλευση από τις γωνίες, ώστε να φαίνεται πιο φυσική η κίνηση

```
42.     gridBackup=grid.clone();
43.     array=array.split(",");
44.         for(var i in array){
45.             if(array[i]==","){
46.                 continue;
47.             }
48.
49.             point_x=parseInt(img[array[i]-1].getAttributeNS(null,"x"));
50.             point_y=parseInt(img[array[i]-1].getAttributeNS(null,"y"));
51.             //if this image has changed position(e.g. has been dragged by the user
    in a different position)
52.             if(img[array[i]-1].transform.baseVal.length!=0){
53.
54.                 transform_x=parseInt(img[array[i]-
    1].transform.baseVal[0].matrix.e);
55.                 point_x+=transform_x;
56.
57.                 transform_y=parseInt(img[array[i]-
    1].transform.baseVal[0].matrix.f);
58.                 point_y+=transform_y;
59.
60.             }
61.
```

source code 34

Στη συνέχεια, εκτελούμε έναν επαναληπτικό βρόχο για κάθε έκθεμα που υπάρχει στη λίστα επίσκεψης. Για κάθε έκθεμα, βρίσκουμε τις συντεταγμένες του (x,y). Επίσης, όπως αναφέρθηκε και προηγουμένως, ελέγχουμε αν έχει μετακινηθεί το έκθεμα από την αρχική του θέση και ενημερώνουμε αναλόγως τις αντίστοιχες μεταβλητές.

```
62.             path1 = finder.findPath((prev_x), (prev_y), point_x, point_y, grid.clon
    e());
63.             console.log(path1);
64.             if(path1.length==0){
65.                 alert("Path not found. Possible conflict between exhibit and wall.H
    ead back to load museum in order to fix this");
66.                 stop_movement_error=true;
67.                 return false;
68.             }
69.             for(k=0;k<path1.length;k++){
70.                 visiting_map[path1[k][1]][path1[k][0]]+=1;
71.                 return_path+=(path1[k][0]-x)+","+(path1[k][1]-y)+" ";
72.             }
73.
74.             prev_x=point_x;
75.             prev_y=point_y;
76.             grid=gridBackup;
77.         }
```

source code 35

Αμέσως μετά, μέσω της μεθόδου findPath της βιβλιοθήκης βρίσκουμε το συντομότερο μονοπάτι από την αρχική θέση του επισκέπτη (prev_x,prev_y) μέχρι τη θέση του εκθέματος, που θέλει να «θαυμάσει» (point_x,point_y). Αν δε βρεθεί κάποιο διαθέσιμο μονοπάτι, τότε η εφαρμογή ενημερώνει τον χρήστη ότι κατά πάσα πιθανότητα υπάρχει κάποιο λάθος στην υλοποίηση του μουσείου και τον παροτρύνει να μεταβεί στην κατάλληλη σελίδα, ώστε να το διορθώσει. Αν όλα πάνε καλά, τότε για κάθε σημείο, που περνάει ο επισκέπτης, ενημερώνουμε τον πίνακα visiting_map, ο οποίος χρησιμοποιείται για τη δημιουργία heatmap και προσθέτουμε στο επιστρεφόμενο string

(return_path) το ολοκληρωμένο μονοπάτι. Επίσης, ενημερώνουμε τις μεταβλητές, που αποθηκεύουν τη θέση εκκίνησης του μονοπατιού, με τις συντεταγμένες του εκθέματος, που μόλις επισκέφθηκε.

```
78.         end=findDoor(true);
79.         console.log(parseInt(end.x)+",,"+parseInt(end.y));
80.         console.log(museum);
81.         path1 = finder.findPath((prev_x), (prev_y), parseInt(end.x), parseInt(end.y
    ), grid.clone());
82.         console.log(path1);
83.             for(k=0;k<path1.length;k++){
84.                 visiting_map[path1[k][1]][path1[k][0]]+=1;
85.                 return_path+=(path1[k][0]-x)+","+(path1[k][1]-y)+" ";
86.             }
87.         return return_path;
88. }
```

source code 36

Επιπρόσθετα, πρέπει να δημιουργήσουμε και την κίνηση από το τελευταίο έκθεμα, που πέρασε ο επισκέπτης, προς την πόρτα εξόδου. Αφού βρούμε αυτή την πόρτα, εκτελούμε ξανά την ίδια διαδικασία, που αναλύθηκε και προηγουμένως. Τέλος, επιστρέφουμε το συνολικό μονοπάτι, το οποίο είναι αποθηκευμένο στη μεταβλητή return_path.

4.2.17 Εύρεση πόρτας

Παρακάτω παρατηρούμε τη συνάρτηση findDoor(), μέσω της οποίας μπορούμε να βρούμε τις συντεταγμένες μίας πόρτας του μουσείου. Η κλήση της συνάρτησης μπορεί να περιέχει από 0 έως 1 όρισμα. Όταν η εφαρμογή καλείται για την εύρεση των συντεταγμένων μίας πόρτας εισόδου, τότε δεν υπάρχει κάποιο όρισμα. Σε αντίθετη περίπτωση, αν θέλουμε, δηλαδή, να βρούμε τις συντεταγμένες μίας πόρτας εξόδου, τότε η συνάρτηση καλείται με όρισμα true.

```
1.  function findDoor(end=false){
2.      var point = museum.createSVGPoint();
3.      var doors=[];
4.      var element=museum.getElementsByTagNameNS(svgNS,"polyline");
5.      var doors_and_walls=museum.getElementsByTagNameNS(svgNS,"line");
6.      for(i=0;i<doors_and_walls.length;i++){
7.          var id=doors_and_walls[i].getAttributeNS(null,"id");
8.          if(id.includes("door")){
9.              if(!end){
10.                 console.log(doors_and_walls[i].textContent);
11.                 if(doors_and_walls[i].textContent=="Entrance/Exit" || doors_and_wal
    ls[i].textContent=="Entrance"){
12.                     doors.push(doors_and_walls[i]);
13.                 }
14.             }
15.             else{
16.                 if(doors_and_walls[i].textContent=="Entrance/Exit" || doors_and_wal
    ls[i].textContent=="Exit"){
17.                     doors.push(doors_and_walls[i]);
18.                 }
19.             }
20.         }
21.
22.     }
```

source code 37

Αρχικά δημιουργούμε ένα SVG point, στο οποίο θα αποθηκεύσουμε τις συντεταγμένες της πόρτας. Στη συνέχεια, και αφού αποθηκεύσουμε το SVG στοιχείο τύπου polyline σε μία μεταβλητή, πρέπει να βρούμε όλες τις πόρτες του μουσείου. Κάθε πόρτα είναι ένα SVG στοιχείο τύπου line, όμως αν αναζητήσουμε όλα τα στοιχεία τύπου line, τότε το viewbox εκτός από τις πόρτες επιστρέφει και τους εσωτερικούς τοίχους του μουσείου. Άρα, πρέπει να ξεχωρίσουμε αυτά τα δύο διαφορετικά στοιχεία. Αυτό επιτυγχάνεται με τη χρήση του χαρακτηριστικού id, διότι όλες οι πόρτες έχουν id “door”, ενώ όλοι οι τοίχοι id “wall”. Αφού ξεχωρίσουμε όλες τις πόρτες, πρέπει να δούμε αν η εφαρμογή θέλει τις συντεταγμένες της πόρτας εισόδου ή της πόρτας εξόδου, ανάλογα με το όρισμα end. Οπότε προσθέτουμε σε έναν πίνακα doors, την κατάλληλη πόρτα, αφού ελέγξουμε τον τίτλο κάθε μίας.

```
23.
24.     for(i=0;i<doors.length;i++){
25.         point.x=parseFloat(doors[i].getAttributeNS(null,"x1"));
26.         point.y=parseFloat(doors[i].getAttributeNS(null,"y1"));
27.         point.x2=parseFloat(doors[i].getAttributeNS(null,"x2"));
28.         point.y2=parseFloat(doors[i].getAttributeNS(null,"y2"));
29.         if(element[0].isPointInStroke(point)){
30.             if(point.y===point.y2){
31.                 point.x+=5;
32.             }
33.             else if(point.x===point.x2){
34.                 point.y+=5;
35.             }
36.             console.log(point);
37.             return point;
38.         }
39.     }
40.
41. }
```

source code 38

Αμέσως μετά, αποθηκεύουμε τις συντεταγμένες της πόρτας στο point. Τέλος, ελέγχουμε αν οι συντεταγμένες αυτές βρίσκονται πάνω στο polyline και ανάλογα με τον τύπο της πόρτας (οριζόντια ή κάθετη) αλλάζουμε την κατάλληλη τιμή, ώστε οι συντεταγμένες να δείχνουν στο κέντρο της.

4.2.18 Δημιουργία Heatmap

Παρακάτω μπορούμε να παρατηρήσουμε τη συνάρτηση createHeatmap, μέσω της οποίας δημιουργούμε ένα heatmap κίνησης, το οποίο δείχνει ποια σημεία του μουσείου (εκθέματα και μη) έχουν μεγαλύτερη συχνότητα επίσκεψης. Η δημιουργία του heatmap γίνεται μέσω της Javascript βιβλιοθήκης heatmap.js, η οποία μας επιτρέπει να εισάγουμε δυναμικά heatmap εύκολα σε μία web εφαρμογή, ώστε να μπορέσουμε να οπτικοποιήσουμε διάφορα δεδομένα.

```
1.  //heatmap.js
2.  // creates a heatmap based on all previous people movements
3.  function createHeatmap(){
4.      var max=0;
5.      var config= {//heatmap instance configuration
6.          container: document.getElementById("heatmap_svg"),
7.          radius: 36.6,
8.          maxOpacity: 1,
9.          minOpacity: 0,
10.         blur: .45,
11.         backgroundColor: 'rgba(0,0,0,.1)',
12.         gradient: { 0.25: "rgb(0,0,255)", 0.55: "rgb(0,255,0)", 0.85: "yellow", 1.0
    : "rgb(255,0,0)"}
13.
14.     };
```

```
15.     //find width and height of newly created div(heatmap_svg)
16.     var heatmapInstance=h337.create(config);
```

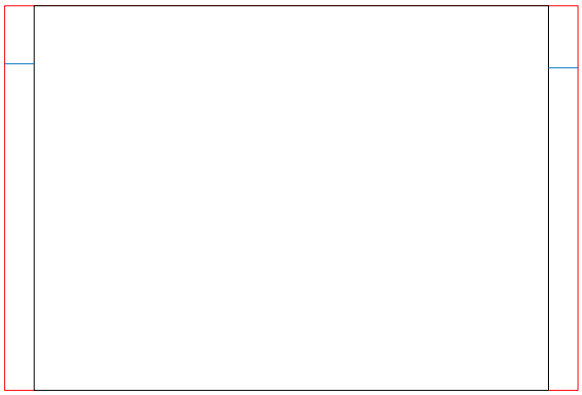
source code 39

Αρχικά, για να μπορέσουμε να δημιουργήσουμε το heatmap, πρέπει να κατασκευάσουμε ένα heatmap instance, το οποίο μπορούμε να το προσαρμόσουμε ανάλογα με ένα αντικείμενο config. Σε αυτό, αναφέρουμε πού θα δημιουργηθεί το heatmap, ποια θα είναι η ακτίνα κάθε datapoint, ποιο χρώμα θα έχει το background κ.ά.

```
17.     var x=document.getElementById("heatmap_svg");
18.     var div_x=x.offsetWidth;
19.     var div_y=x.offsetHeight;
20.
21.     var y=document.getElementById("museum");
22.
23.
24.     //find the difference in height and width between the heatmap_svg div and the v
    iewbox
25.     var data_x=div_x-y.clientWidth;
26.     var data_y=div_y-y.clientHeight;
27.
28.     var polyline=museum.getElementsByTagNameNS(svgNS,"polyline");
29.     let box=y.getBBox();
30.     var pol_points=polyline[0].points;
31.
32.     var mp=museumPoint(museum,pol_points[0].x,pol_points[0].y);
33.
34.     //find number of pixels that each viewbox element represents
35.     var a_x=y.clientWidth/100;
36.     var a_y=y.clientHeight/50;
```

source code 40

Στη συνέχεια, παρατηρήσαμε ότι το heatmap_svg div και το viewbox έχουν κάποιες διαφορές στο ύψος και το πλάτος τους, οπότε έπρεπε να κάνουμε κάποιους υπολογισμούς για να βρούμε αυτή τη διαφορά. Αρχικά, αποθηκεύουμε το ύψος και το πλάτος του svg_heatmap div σε διαφορετικές μεταβλητές και βρίσκουμε τη διαφορά αυτών των δύο. Έπειτα βρίσκουμε τον αριθμό των pixels, που αντιπροσωπεύει κάθε στοιχείο του viewbox. Στο παρακάτω σχήμα, μπορούμε να παρατηρήσουμε τη διαφορά, που υπάρχει ανάμεσα στο viewbox (μαύρο) και στο heatmap_svg (κόκκινο). Όπως βλέπουμε τα δύο στοιχεία έχουν ίσο ύψος, αλλά διαφορετικό πλάτος. Αυτή η διαφορά τους, επισημαίνεται στο σχήμα από τις δύο μπλε γραμμές.



```
37.     var datapoints=[];
38.     for(var i=0;i<50;i++){
```

```
        for(var j=0;j<100;j++){
            if(visiting_map[i][j]>max){
                max=visiting_map[i][j];
            }
            var dataPoint={
                x:parseInt((data_x/2)+(j*a_x)),
                y:parseInt((data_y)+(i*a_y)),
                value:visiting_map[i][j]
            };
            datapoints.push(dataPoint);

        }

    }

    var data={
        max:max,
        min:1,
        data: datapoints

    };
    heatmapInstance.setData(data);
    var current_data=heatmapInstance.getData();
}
```

source code 41

Αμέσως μετά, δημιουργούμε έναν πίνακα με όνομα datapoints, ο οποίος θα περιέχει πληροφορίες για κάθε στοιχείο του heatmap. Όπως είχαμε αναφέρει και παραπάνω, χρησιμοποιούμε τον πίνακα visiting_map, ώστε να γνωρίζουμε την επισκεψιμότητα κάθε σημείου του μουσείου. Επίσης ανάλογα με τους υπολογισμούς, που εξηγήσαμε στην προηγούμενη παράγραφο, εισάγουμε τις κατάλληλες τιμές για τα χαρακτηριστικά x και y κάθε datapoint. Άρα για να βρούμε τη x συντεταγμένη ενός στοιχείου, διαιρούμε τη διαφορά πλάτους των δύο στοιχείων με το 2 και προσθέτουμε την τιμή του στοιχείου στον άξονα x επί τον αριθμό των pixel, που καταλαμβάνει κάθε ένα σε αυτό τον άξονα. Αντίστοιχα για τη συντεταγμένη y, προσθέτουμε στη διαφορά ύψους των δύο στοιχείων, η οποία είναι 0 στην προκειμένη περίπτωση, την τιμή του στοιχείου στον άξονα y επί των αριθμό των pixel, που καταλαμβάνει κάθε ένα σε αυτό τον άξονα. Τέλος, δημιουργούμε ένα αντικείμενο data, το οποίο θα εισαχθεί στο heatmap instance με τα εξής χαρακτηριστικά:

- max → η μέγιστη τιμή, που θα έχει ένα στοιχείο του heatmap
- min → η ελάχιστη τιμή
- data → τα δεδομένα, που θα εισαχθούν στο heatmap (πίνακας datapoints)

4.2.19 Αποθήκευση μουσείου με heatmap τοπικά

Παρακάτω παρατηρούμε τη συνάρτηση save_heatmap, μέσω της οποίας ο χρήστης μπορεί να αποθηκεύσει το μουσείο του μαζί με το εξαγόμενο heatmap, τοπικά στον υπολογιστή του σε μορφή png. Αυτή η λειτουργία επιτυγχάνεται με τη χρήση της Javascript βιβλιοθήκης DOM to Image.

```
//save created heatmap and museum in your local storage
function save_heatmap(){
var name = prompt("Please enter a name for the file:", "Museum_heatmap");
  if (name == null || name == "") {
    alert("User cancelled the prompt.");
  } else {
      var node = document.getElementById('heatmap_svg');

```

```
9.  domtoimage.toPng(node)
10.     .then(function (dataUrl) {
11.         var img = new Image();
12.         img.src = dataUrl;
13.
14.          var link = document.createElement('a');
15.         link.download = name + '.png';
16.         link.href = dataUrl;
17.         link.click();
18.     })
19.     .catch(function (error) {
20.         console.error('oops, something went wrong!', error);
21.     });
22.     alert("Download file to local storage");
23.   }
24.
25.
26.
27. }
```

source code 42

Αρχικά, η εφαρμογή εμφανίζει ένα popup παράθυρο, το οποίο ζητάει από το χρήστη ένα όνομα για να αποθηκεύσει την παραγόμενη εικόνα τοπικά. Έπειτα επιλέγουμε το DOM αντικείμενο, που θα εξαχθεί ως εικόνα. Στην προκειμένη περίπτωση, πρόκειται για το heatmap_svg div. Στη συνέχεια, ακολουθώντας τις οδηγίες από το documentation της βιβλιοθήκης, μετατρέπουμε το DOM αντικείμενο σε png εικόνα και εκκινείται η διαδικασία λήψης της στον υπολογιστή του χρήστη. Τέλος, ενημερώνουμε τον χρήστη ότι η λήψη της εικόνας ξεκίνησε, με κατάλληλο μήνυμα.

Κεφάλαιο 5: Επίλογος- Συμπεράσματα -Μελλοντικές επεκτάσεις

5.1 Επίλογος

Με την ολοκλήρωση της παρούσας διπλωματικής εργασίας, παρουσιάστηκε μια διαδικτυακή εφαρμογή δημιουργίας μουσείων και προσομοίωσης κίνησης σε αυτά. Κατά την εισαγωγή αναλύθηκε το ζήτημα που μας τέθηκε προς υλοποίηση, καθώς και η λύση, την οποία προτείναμε. Στη συνέχεια, αναφέρθηκαν εκτενώς οι τεχνολογίες και οι επιπρόσθετες βιβλιοθήκες που χρησιμοποιήθηκαν για την εκπόνηση της εφαρμογής. Έπειτα, παρουσιάστηκε ο τρόπος σχεδίασης της εφαρμογής, καθώς και οι λειτουργικές προδιαγραφές της. Αμέσως μετά αναφερθήκαμε στην υλοποίηση της βάσης δεδομένων και των κυριότερων λειτουργιών της εφαρμογής. Τέλος, δόθηκε ένα εγχειρίδιο χρήσης το οποίο εξηγεί όλα τα πιθανά σενάρια της εφαρμογής.

5.2 Συμπεράσματα

Κατά τη διάρκεια εκπόνησης της παρούσας εργασίας, οι δυσκολίες που κληθήκαμε να αντιμετωπίσουμε ήταν πολλές. Παρόλα αυτά με τη βοήθεια του κυρίου Τσελίκα καταφέραμε να ανταπεξέλθουμε στην πλειονότητά τους. Πιο συγκεκριμένα, οι περισσότερες δυσκολίες, που αντιμετωπίσαμε αφορούσαν τη λειτουργία της μετακίνησης των SVG αντικειμένων στο χώρο του μουσείου, καθώς και την αποθήκευση των καινούριων συντεταγμένων τους. Επίσης, άλλο ένα κομμάτι της εφαρμογής, στο οποίο παρουσιάστηκαν κάποιες δυσκολίες είναι στη προσομοίωση κίνησης των επισκεπτών και πιο συγκεκριμένα στη δημιουργία ενός μονοπατιού κίνησης, με βάση την είσοδο του χρήστη. Για να επιλύσουμε αυτή τη δυσκολία, καταφύγαμε τελικά σε διάφορους αλγορίθμους εύρεσης κοντινότερου μονοπατιού και σε μία βιβλιοθήκη Javascript (Pathfinding.js), η οποία δέχεται ως όρισμα ένα grid με εμπόδια και υπολογίζει την κοντινότερη απόσταση μεταξύ δύο σημείων.

5.3 Μελλοντικές επεκτάσεις

Η εφαρμογή μας είναι μια εφαρμογή διαδικτυακού λογισμικού βασισμένη σε κώδικα PHP και JavaScript. Παρουσιάστηκε στην παρούσα πτυχιακή και αποτελεί μια λύση για κάποιον που επιθυμεί να σχεδιάσει ένα μουσείο και να προσομοιώνει την κίνηση μέσα σε αυτό. Με αυτή την εφαρμογή ένας διαχειριστής μουσείου μπορεί να παρατηρήσει την επισκεψιμότητα κάθε εκθέματος και να προβλέψει πιθανά σημεία του μουσείου, όπου μπορεί να υπάρξει συνωστισμός ή απουσία κίνησης. Οπότε, με αυτόν τον τρόπο ο διαχειριστής διασφαλίζει την καλύτερη εμπειρία επίσκεψης μέσα στον χώρο. Οι τεχνικές που χρησιμοποιήθηκαν έγιναν στα χρονικά πλαίσια εκπόνησης μιας μεταπτυχιακής εργασίας, συνεπώς σίγουρα υπάρχουν περιθώρια περαιτέρω βελτίωσης της εφαρμογής στο μέλλον. Μερικά από αυτά είναι:

- Καλύτερο γραφικό περιβάλλον

  Το περιβάλλον στο οποίο ένας χρήστης εισέρχεται, πολλές φορές, έχει μεγαλύτερο αντίκτυπο σε μια εφαρμογή, από ότι άλλα προβλήματα που πιθανόν να έχει. Αυτό σημαίνει ότι πρέπει πρώτα να “ευχαριστηθεί” το μάτι και μετά να γίνουν όλα τα υπόλοιπα.
- Διόρθωση πιθανών bugs

  Σε μια εφαρμογή τέτοιου είδους, είναι λογικό ότι χωρίς χρήση από αρκετούς χρήστες, και περισσότερο χρόνο για την ανάπτυξή της, θα υπάρχουν διάφορα σφάλματα τα οποία πρέπει να διορθωθούν, άλλα με μεγαλύτερη αναγκαιότητα και επιτακτικότατα και άλλα με μικρότερη.
- Περισσότερα στατιστικά στοιχεία και συμβουλές

  Θα ήταν πολύ χρήσιμο για τους διαχειριστές κάθε μουσείου να δέχονται συμβουλές και στατιστικά στοιχεία, ανάλογα με την προσομοίωση. Συγκεκριμένα, η εφαρμογή μπορεί να ενημέρωνει τον χρήστη για πιθανές αλλαγές που πρέπει να γίνουν στη διαρρύθμιση του μουσείου. Επίσης, μπορεί ο χρήστης να δίνει επιπλέον στοιχεία που αφορούν στην ηλικία, στο φύλο, στο είδος του γκρουπ, ακόμα και στην ώρα εισόδου και εξόδου του γκρουπ από το μουσείο, με τα οποία θα δημιουργούνται γραφήματα, διαγράμματα και κάθε είδους στατιστικές αναπαραστάσεις, έτσι ώστε να μπορεί η εφαρμογή να μας παρουσιάσει στατιστικά για κάθε κατηγορία αυτών των στοιχείων. Παραδείγματος χάριν, θα μπορεί ο διαχειριστής να δει ποιες μέρες και ώρες της ημέρας επισκέπτονται το μουσείο περισσότερα άτομα, τον μέσο όρο της ηλικίας των επισκεπτών κτλ.

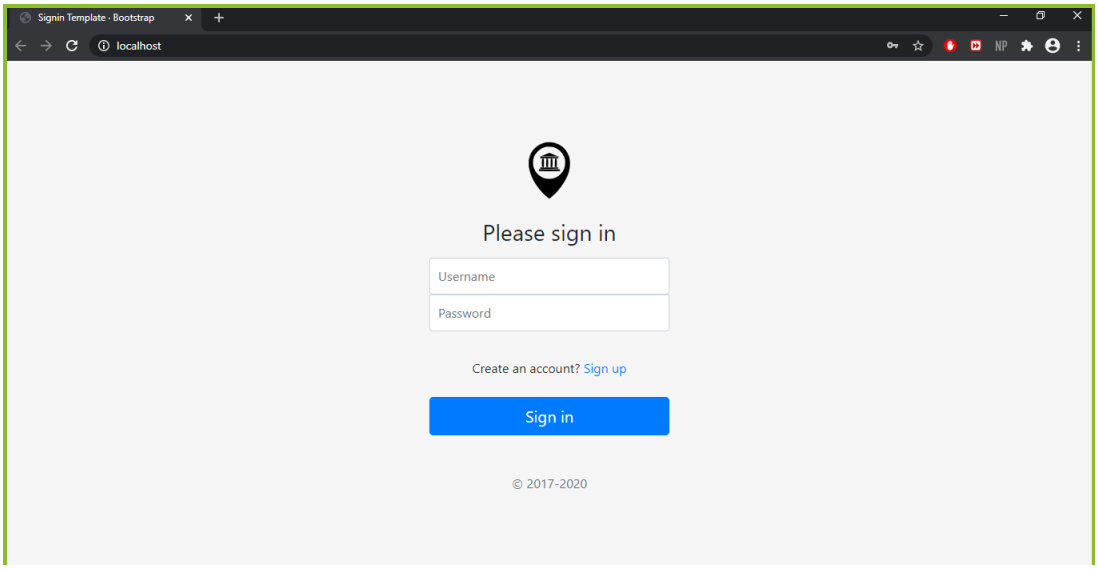
Κεφάλαιο 6: ΠΑΡΑΡΤΗΜΑ

6.1 ΠΑΡΑΡΤΗΜΑ Ι : Εγχειρίδιο χρήσης

Σκοπός του παρόντος εγχειριδίου είναι να δοθούν κατευθυντήριες γραμμές στα άτομα τα οποία πρόκειται να χρησιμοποιήσουν την εφαρμογή. Συγκεκριμένα, δίνονται κατάλληλες οδηγίες για κάθε λειτουργία της εφαρμογής.

6.1.1 Είσοδος στην εφαρμογή

Μόλις εκκινήσετε την εφαρμογή εμφανίζεται η ακόλουθη φόρμα διασύνδεσης.

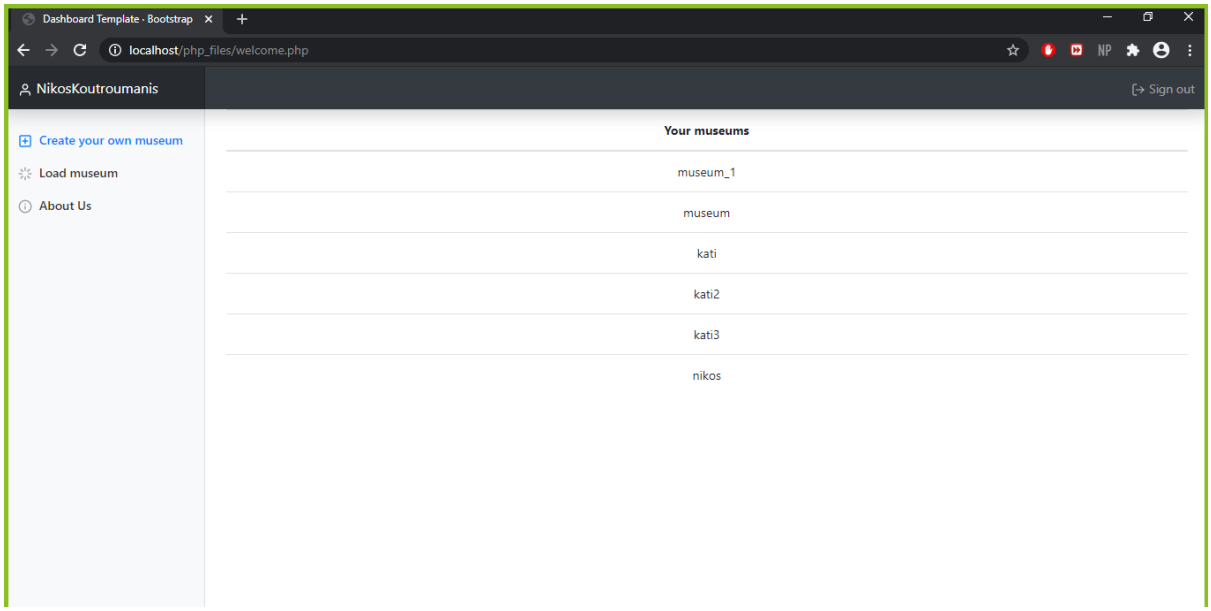


Εικόνα 10 Σελίδα σύνδεσης

Τα στοιχεία που πρέπει να συμπληρώσετε για την είσοδό σας είναι:

- **Όνομα Χρήστη** Το username που έχετε επιλέξει
- **Κωδικός** Ο κωδικός πρόσβασης που επιλέξατε κατά την εγγραφή σας στην εφαρμογή

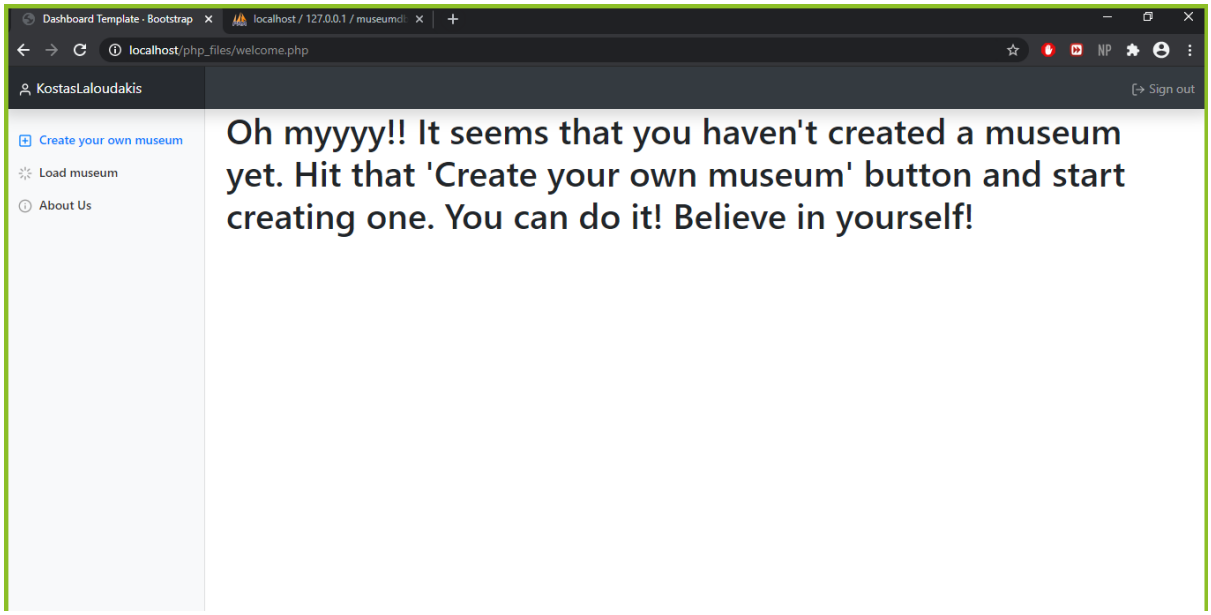
Αφού εισαγάγετε τα σωστά στοιχεία, θα ανακατευθυνθείτε στην αρχική σας σελίδα.



Εικόνα 11 Αρχική σελίδα εφαρμογής

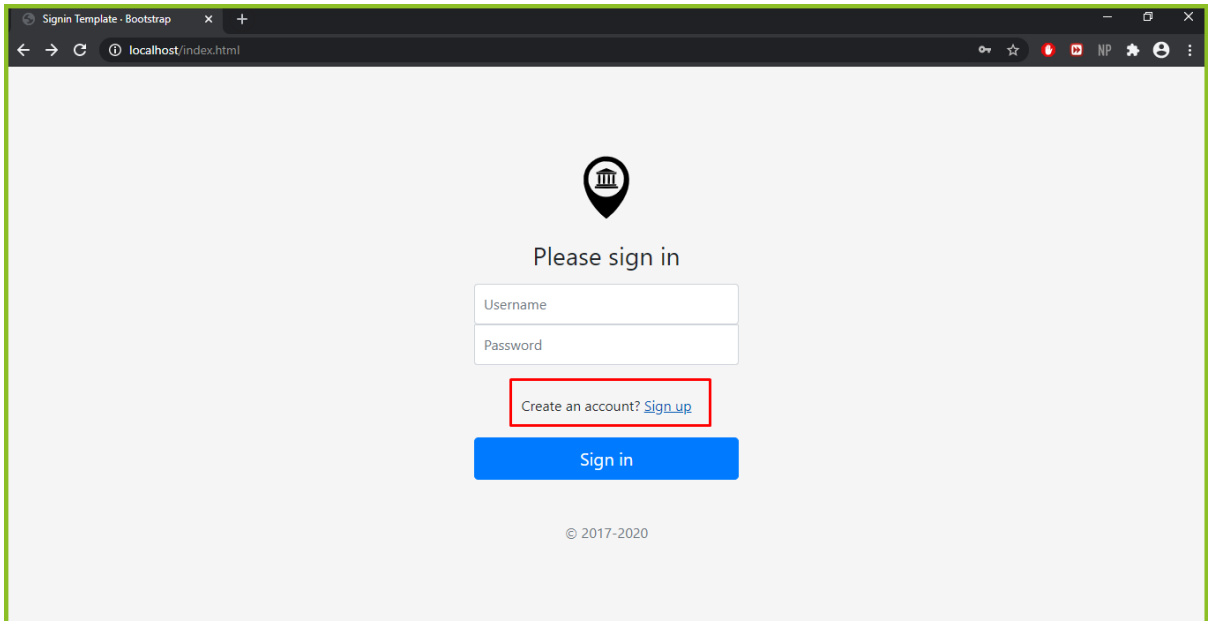
Πάνω αριστερά μπορείτε να παρατηρήσετε το username σας, ενώ από κάτω υπάρχει το menu, από το οποίο μπορείτε να μεταφερθείτε σε όλες τις διαθέσιμες διαδικασίες. Στο κέντρο εμφανίζεται ένας πίνακας με τα ονόματα όλων των μουσείων, που έχετε ήδη δημιουργήσει.

Αν δεν έχετε δημιουργήσει κάποιο μουσείο μέχρι στιγμής, τότε σε εκείνο το σημείο θα εμφανίζεται το ακόλουθο κείμενο.

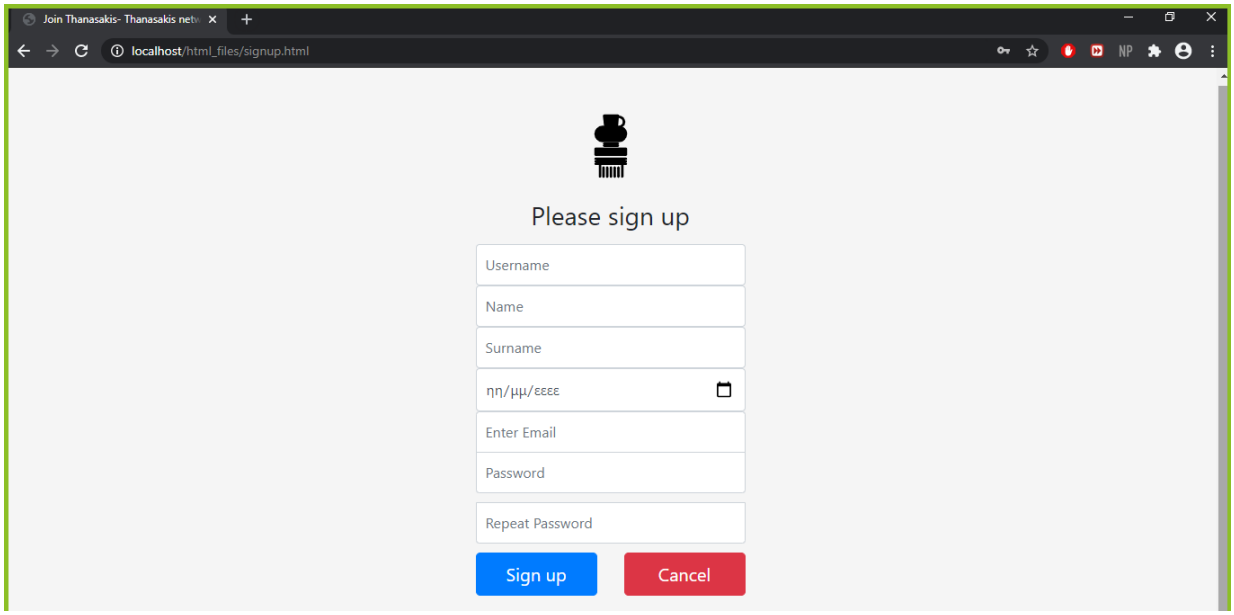


6.1.2 Εγγραφή νέου χρήστη

Αν δεν έχετε ήδη δημιουργήσει κάποιο λογαριασμό στην εφαρμογή μας, σας δίνεται η δυνατότητα να το κάνετε δωρεάν, κάνοντας κλικ στον υπερσύνδεσμο Sign up, στην αρχική σελίδα.



Με αυτόν τον τρόπο θα μεταφερθείτε σε μία άλλη σελίδα, όπου θα υπάρχει η ακόλουθη φόρμα προς συμπλήρωση.



Εικόνα 12 Σελίδα εγγραφής νέου χρήστη

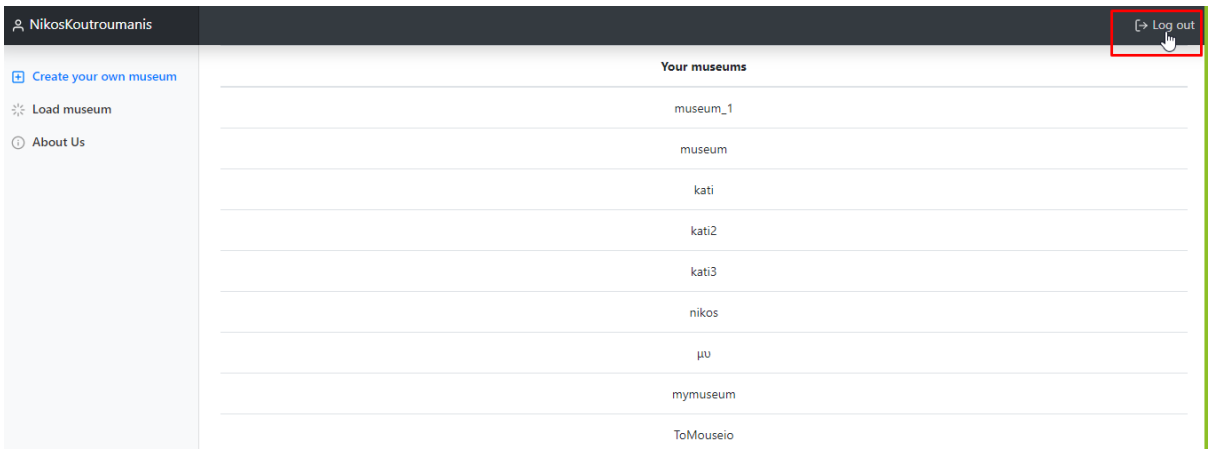
Για την επιτυχή εγγραφή σας στην εφαρμογή θα πρέπει να συμπληρώσετε τα παρακάτω στοιχεία:

1. Όνομα Χρήστη
2. Όνομα
3. Επώνυμο
4. Ημερομηνία Γέννησης
5. Διεύθυνση ηλεκτρονικού ταχυδρομείου
6. Κωδικός Χρήστη (2 φορές για επαλήθευση)

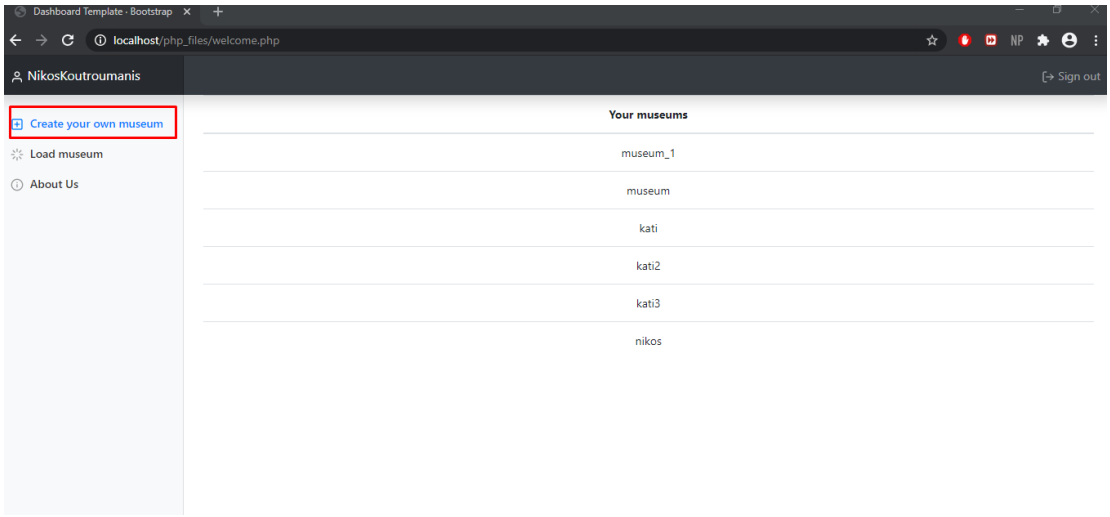
Αν όλα τα στοιχεία, που εισαγάγατε είναι αποδεκτά, τότε μεταφέρεστε ξανά στη σελίδα εισόδου, όπου πρέπει να ακολουθήσετε τα βήματα, που αναλύσαμε παραπάνω (φόρμα εισόδου).

6.1.3 Αποχώρηση από την εφαρμογή (logout)

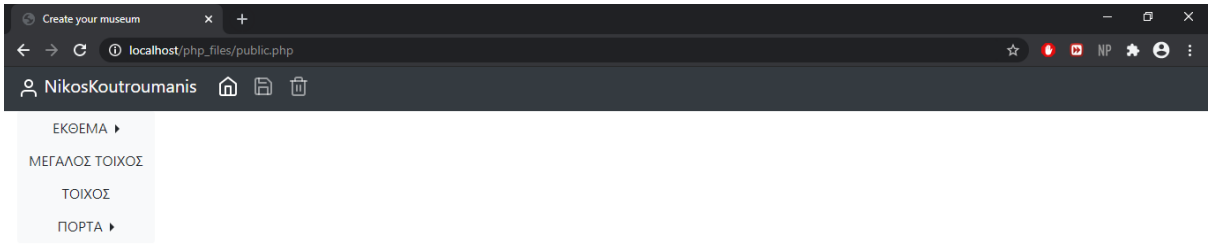
Για να αποχωρήσετε από την εφαρμογή αρκεί να πατήσετε στο κουμπί Log Out, το οποίο βρίσκεται πάνω δεξιά στη σελίδα σας.



6.1.4 Δημιουργία νέου μουσείου

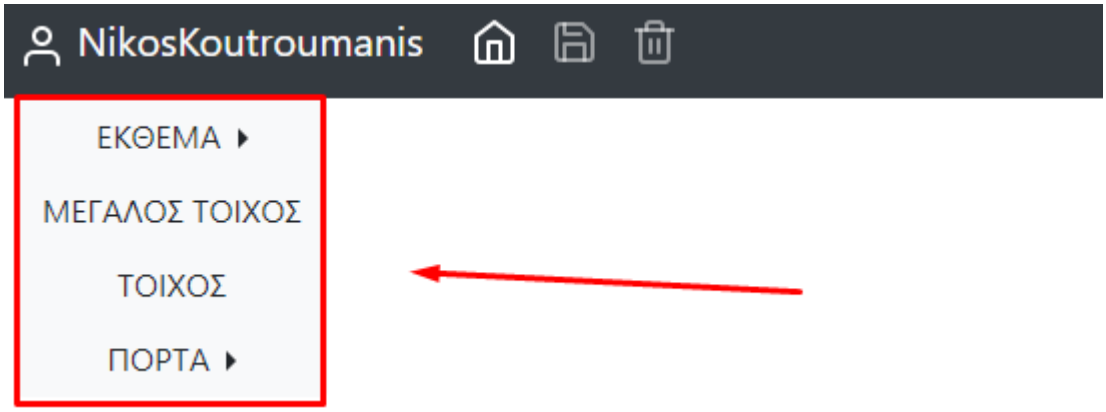


Για να ξεκινήσετε τη δημιουργία ενός νέου μουσείου, αρκεί να πατήσετε το κουμπί Create your own museum, που βρίσκεται στο menu, στην αριστερή πλευρά της σελίδας, κάτω από το όνομα χρήστη σας. Αφού το πατήσετε, μεταφέρεστε σε μία νέα ιστοσελίδα.

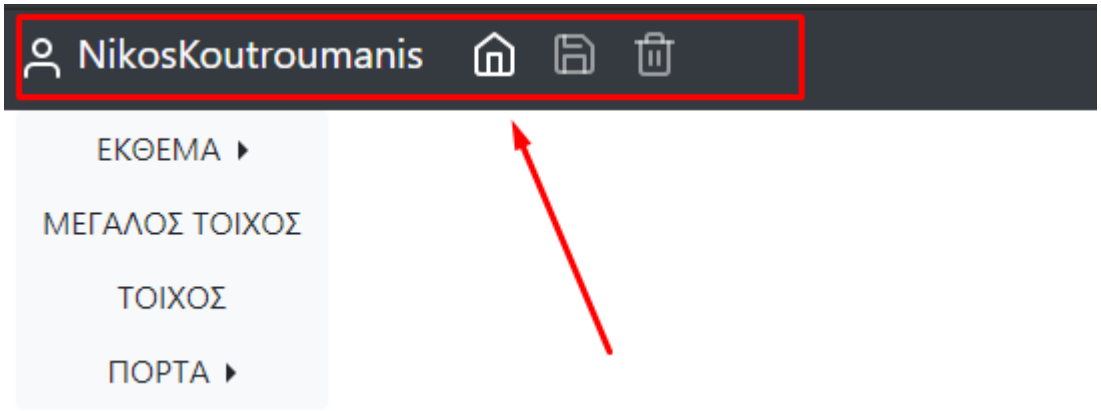


Σε αυτή την ιστοσελίδα, όπως μπορείτε να παρατηρήσετε εμφανίζεται και πάλι το username σας, πάνω αριστερά. Από κάτω υπάρχει ένα καινούριο menu με τα παρακάτω στοιχεία:

1. Έκθεμα
2. Μεγάλος Τοίχος
3. Τοίχος
4. Πόρτα



Επίσης, δίπλα από το username, υπάρχουν και 3 buttons, κάθε ένα από τα οποία είναι υπεύθυνο για μία διαφορετική λειτουργία.



- Σπίτι -> Επιστροφή στην αρχική σελίδα
- Δίσκος -> Αποθήκευση του τρέχοντος μουσείου
- Κάδος ανακύκλωσης -> Διαγραφή ενός στοιχείου

Τέλος, η υπόλοιπη σελίδα είναι κενή, καθώς εκεί θα δημιουργήσετε το μουσείο σας.

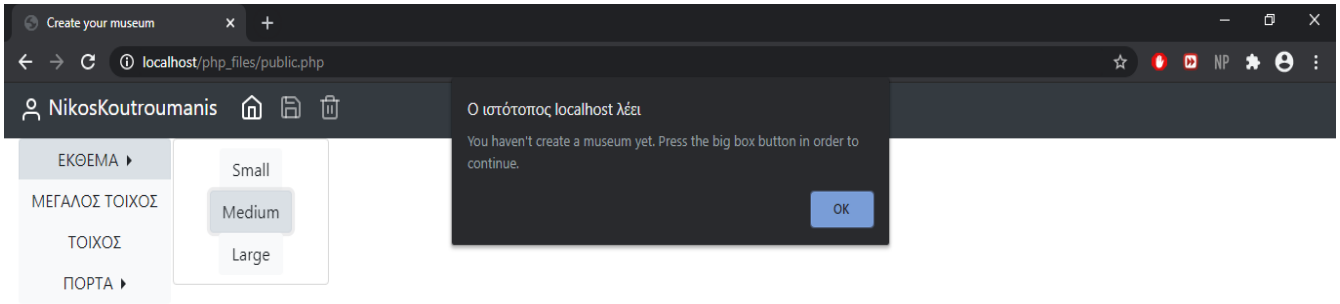
6.1.5 Εισαγωγή μεγάλου τοίχου

Για να ξεκινήσετε να εισάγετε στοιχεία στο μουσείο σας, πρέπει καταρχάς να δημιουργήσετε τους εξωτερικούς τοίχους του. Αυτό γίνεται πατώντας το κουμπί **ΜΕΓΑΛΟΣ ΤΟΙΧΟΣ**, από τη λίστα στα αριστερά της σελίδας. Μεγάλους τοίχους θεωρούμε τους εξωτερικούς τοίχους του μουσείου. Αυτό θα έχει ως αποτέλεσμα να μετατραπεί η σελίδα σας κάπως έτσι:

Μπορείτε να εισάγετε μόνο έναν μεγάλο τοίχο, οπότε, σε περίπτωση επιλογής αυτού του κουμπιού, ενώ έχετε δημιουργήσει ήδη τους εξωτερικούς τοίχους, θα εμφανιστεί κατάλληλο μήνυμα λάθους.

ΠΡΟΣΟΧΗ!!!!!

Αν προσπαθήσετε να εισάγετε κάποιο έκθεμα, κάποια πόρτα ή κάποιον τοίχο, τότε η εφαρμογή σας ενημερώνει ότι αυτό δεν είναι δυνατό, χωρίς την ύπαρξη των εξωτερικών τοίχων, οπότε είναι απαραίτητο το πάτημα αυτού του κουμπιού αν θέλετε να ξεκινήσετε την κατασκευή του μουσείου σας.



Όπως βλέπετε και στην παραπάνω εικόνα, προσπαθούμε να εισάγουμε ένα μεσαίου μεγέθους έκθεμα, χωρίς να έχουμε πατήσει το κουμπί Μεγάλος Τοίχος, και η εφαρμογή μας υποδεικνύει ότι, αν δεν εισάγουμε τους εξωτερικούς τοίχους, δεν μπορούμε να συνεχίσουμε.

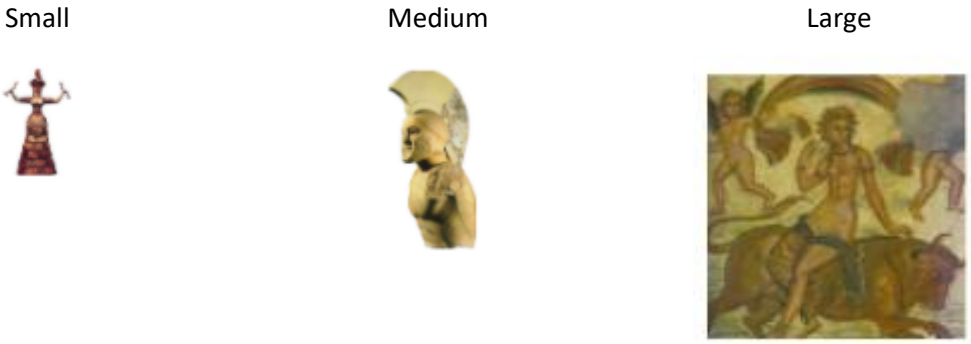
6.1.6 Εισαγωγή εκθέματος

Για την εισαγωγή κάποιου εκθέματος στο μουσείο σας, πρέπει να πατήσετε το κουμπί **ΕΚΘΕΜΑ**, από το αριστερό μενού της σελίδας. Μόλις γίνει αυτό, εμφανίζεται ένα καινούριο μενού με 3 επιλογές:

- Small

- Medium
- Large

Αφού επιλέξετε κάποιο από τα 3, μπορείτε να πατήσετε σε οποιοδήποτε μέρος του μουσείου θέλετε να το εισάγετε. Ανάλογα με την επιλογή σας παραπάνω, το έκθεμα που θα εμφανιστεί θα είναι το εξής:



Εικόνα 13 Εκθέματα

Παραπάνω βλέπουμε ένα μουσείο, όπου έχουμε εισάγει ένα έκθεμα από κάθε μέγεθος και τα έχουμε τοποθετήσει σε συγκεκριμένα σημεία.

Υπάρχει η δυνατότητα μετακίνησης ενός εκθέματος όπως εσείς το επιθυμείτε, για περαιτέρω τεχνικές πληροφορίες μεταφερθείτε στο κεφάλαιο μετακίνηση αντικειμένων (4.12).

6.1.7 Εισαγωγή πόρτας

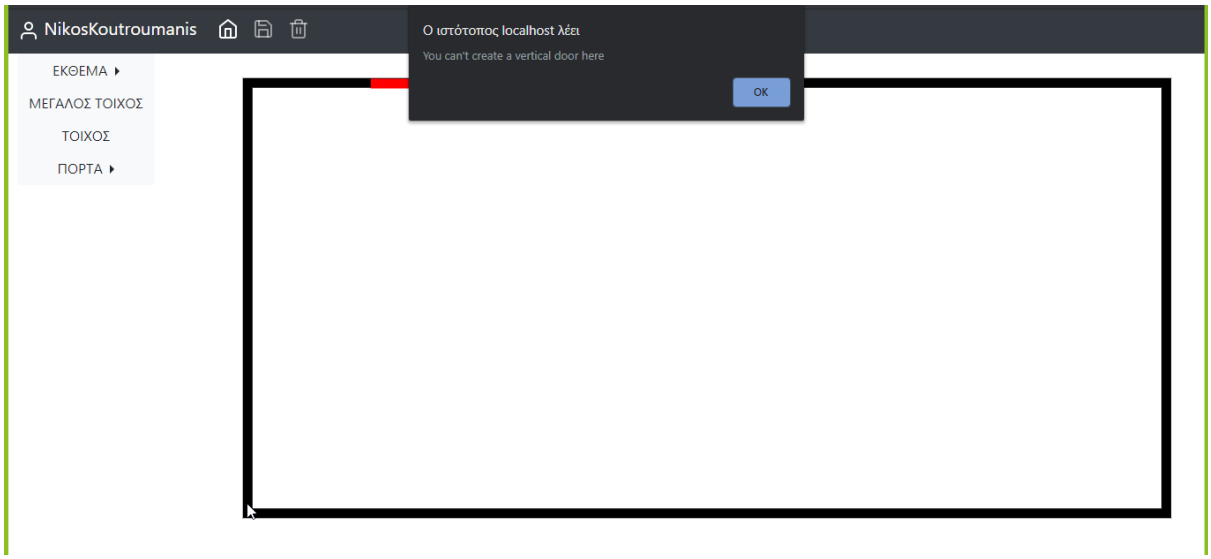
Μέσω της παλέτας στα αριστερά, σας δίνεται η δυνατότητα για την εισαγωγή μιας καινούριας πόρτας στο μουσείο. Πατώντας το κουμπί **ΠΟΡΤΑ** εμφανίζονται δύο επιλογές, η εισαγωγή οριζόντιας ή κάθετης πόρτας. Ο χρήστης έχει τη δυνατότητα να εισάγει μέχρι δύο πόρτες. Μόλις επιλέξετε τον

τύπο της πόρτας, πρέπει να πατήσετε **σε ένα σημείο του εξωτερικού τοίχου**, όπου θέλετε να τη δημιουργήσετε.

Στην παραπάνω εικόνα, επιλέξαμε το κουμπί **ΠΟΡΤΑ** και στη συνέχεια την επιλογή Horizontal. Έπειτα, επιλέξαμε ένα σημείο του εξωτερικού τοίχου, όπου θέλαμε να την τοποθετήσουμε.

ΠΡΟΣΟΧΗ!!!!

Η πόρτα θα έχει ένα σταθερό μέγεθος, οπότε η εφαρμογή ελέγχει αν στο σημείο που επιλέξατε υπάρχει επαρκής χώρος για να φτιαχτεί. Αν δεν υπάρχει, τότε σας ενημερώνει με κατάλληλο μήνυμα, όπου σας προτρέπει να τη δημιουργήσετε κάπου αλλού.



Στην παραπάνω εικόνα, μπορείτε να παρατηρήσετε ότι προσπαθούμε να δημιουργήσουμε μία κάθετη πόρτα, στην κάτω αριστερή πλευρά του μουσείου. Επειδή ο εξωτερικός τοίχος δεν επαρκεί

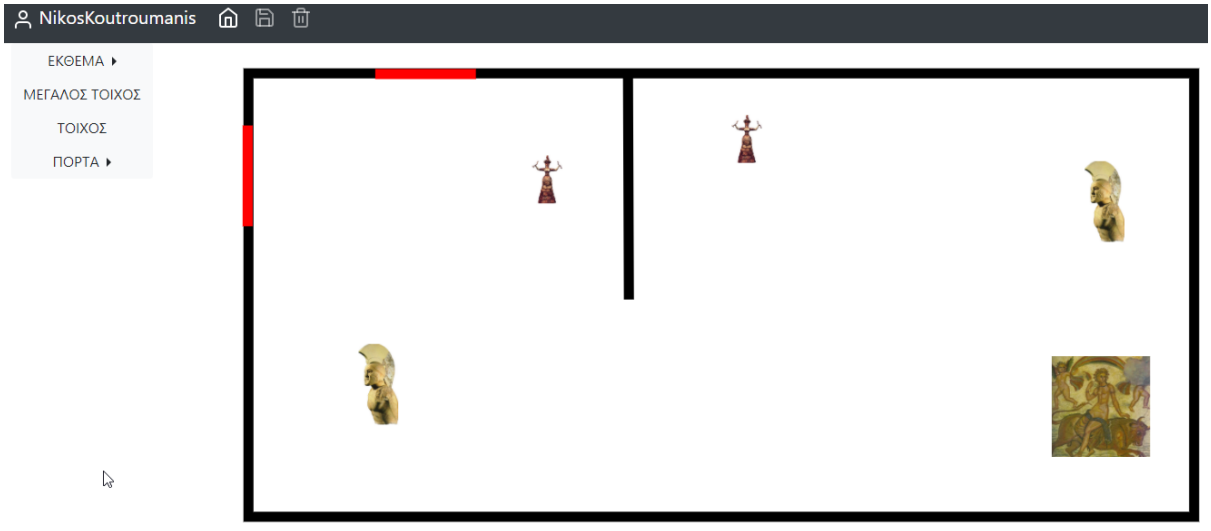
ώστε να φτιαχτεί ολόκληρη η πόρτα, η εφαρμογή μας ενημερώνει ότι η κατασκευή αυτής δεν είναι δυνατή σε εκείνο το σημείο.

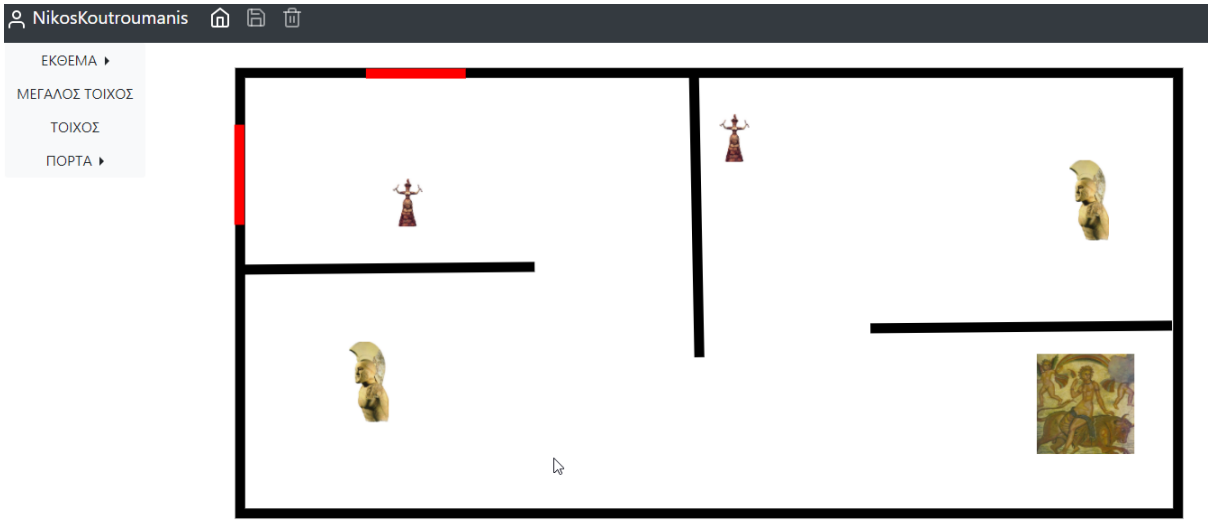
Η πρώτη πόρτα, που θα εισαγάγετε στο μουσείο σας θα είναι πάντα η είσοδός του. Αν επιλέξετε να προσθέσετε και μια δεύτερη, τότε αυτή θα αναπαριστά την έξοδο. Από την άλλη πλευρά, αν επιλέξετε να αποθηκεύσετε το μουσείο με μόνο μία πόρτα, τότε αυτή θα μετατραπεί σε Είσοδο/Έξοδο. Περνώντας το ποντίκι σας πάνω από μία πόρτα, μπορείτε να ενημερωθείτε για τον τύπο της (Είσοδος, Έξοδος ή και τα δύο).

6.1.8 Εισαγωγή εσωτερικών τοίχων

Για να εισάγετε έναν εσωτερικό τοίχο στο μουσείο σας, αρκεί να πατήσετε το κουμπί **ΤΟΙΧΟΣ**, το οποίο βρίσκεται στην αριστερή πλευρά της σελίδας. Αμέσως μετά, κάνετε διπλό κλικ σε κάποιο σημείο εντός του μουσείου σας, το οποίο θα είναι η αρχή του τοίχου. Έπειτα, χωρίς να αφήσετε τον κέρσορα μπορείτε να μεταβάλλετε το μήκος του και την κατεύθυνσή του, ανάλογα με τις κινήσεις του ποντικιού σας. Μόλις αφήσετε τον κέρσορα, η τελική μορφή του τοίχου είναι έτοιμη. Υπάρχει η δυνατότητα μετακίνησης ενός έτοιμου τοίχου όπως εσείς το επιθυμείτε, για περαιτέρω τεχνικές πληροφορίες μεταφερθείτε στο κεφάλαιο μετακίνηση αντικειμένων (4.12).

Παρακάτω, μπορείτε να παρατηρήσετε ότι έχουμε δημιουργήσει έναν εσωτερικό τοίχο στο μουσείο μας. Αφού επιλέξαμε το κουμπί **ΤΟΙΧΟΣ**, αποφασίσαμε να τον φτιάξουμε ανάμεσα στα 2 μικρά εκθέματα, που υπήρχαν.

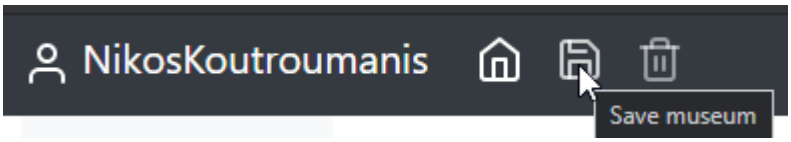




Παραπάνω μπορείτε να παρατηρήσετε ένα μουσείο, στο οποίο έχουμε χρησιμοποιήσει όλες τις παραπάνω λειτουργίες. Όπως μπορείτε να παρατηρήσετε έχουμε προσθέσει 2 πόρτες, μία για είσοδο και μία για έξοδο, 5 εκθέματα (2 μικρού μεγέθους, 2 μεσαίου και 1 μεγάλου), καθώς και 3 εσωτερικούς τοίχους.

6.1.9 Αποθήκευση μουσείου

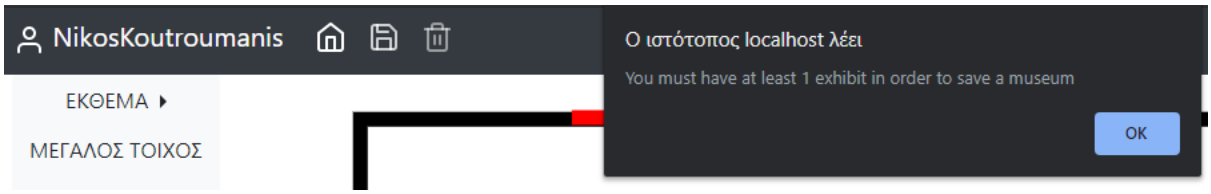
Αφού έχετε δημιουργήσει ένα μουσείο, πρέπει να το αποθηκεύσετε στον λογαριασμό σας, ώστε να είναι δυνατή η τροποποίησή του κάποια άλλη φορά ή και η προσομοίωση κίνησης σε αυτό. Για να αποθηκεύσετε το μουσείο σας, αρκεί να πατήσετε τη δισκέτα, η οποία βρίσκεται στο πάνω αριστερά μέρος της οθόνης.



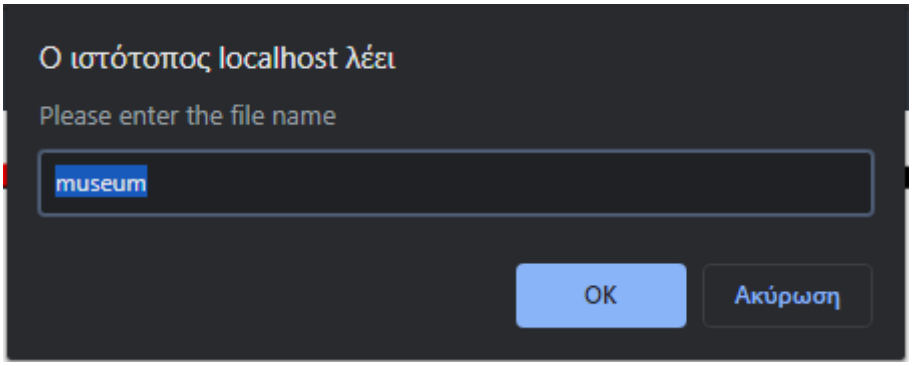
Μόλις πατήσετε αυτό το κουμπί, η εφαρμογή ελέγχει αν πληρούνται κάποιες προϋποθέσεις, οι οποίες είναι απαραίτητες για να θεωρείται ένα μουσείο έγκυρο. Αυτές είναι:

- Ύπαρξη εξωτερικών τοίχων
- Ύπαρξη τουλάχιστον ενός εκθέματος
- Ύπαρξη τουλάχιστον μίας πόρτας

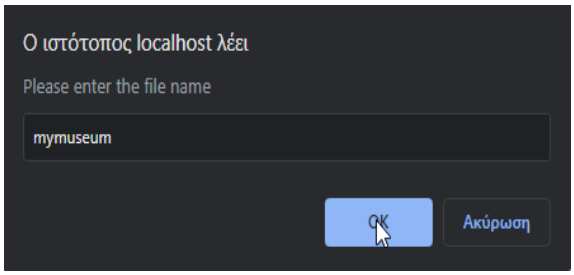
Αν κάποια από αυτές τις 3 προϋποθέσεις δεν ισχύει, τότε η εφαρμογή εμφανίζει κατάλληλο μήνυμα λάθους.

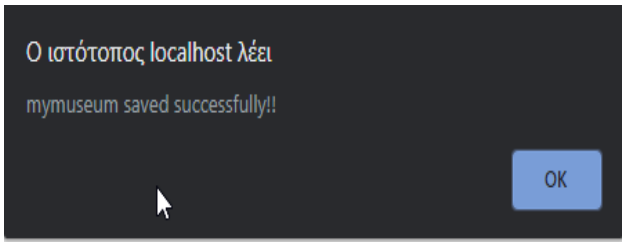


Από την άλλη πλευρά, αν το μουσείο είναι έγκυρο, τότε εμφανίζεται ένα καινούριο παράθυρο διαλόγου, το οποίο σας ζητάει να δώσετε ένα όνομα στο μουσείο σας.

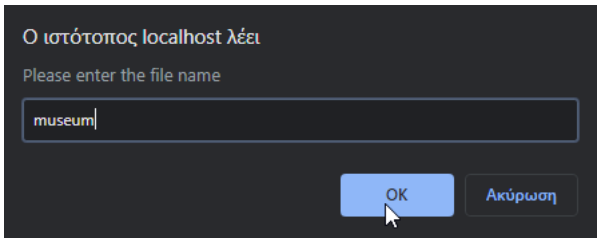


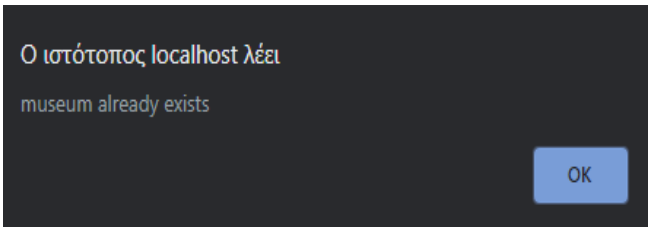
Όπως μπορείτε να παρατηρήσετε, η εφαρμογή σας προτείνει ένα default όνομα (museum), το οποίο μπορείτε να χρησιμοποιήσετε ή να το αγνοήσετε και να πληκτρολογήσετε κάποιο δικό σας όνομα. Αφού αποφασίσετε για το όνομα του μουσείου, κάνετε κλικ στο κουμπί OK, ώστε να αποθηκευτεί στον λογαριασμό σας.





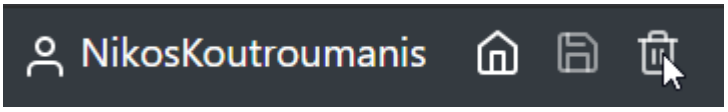
Σε περίπτωση που το όνομα που επιλέξατε έχει χρησιμοποιηθεί ήδη και σε κάποιο άλλο ήδη υπάρχον μουσείο σας, τότε η εφαρμογή σας ενημερώνει ότι αυτό το όνομα υπάρχει, οπότε η αποθήκευση δεν ολοκληρώνεται.



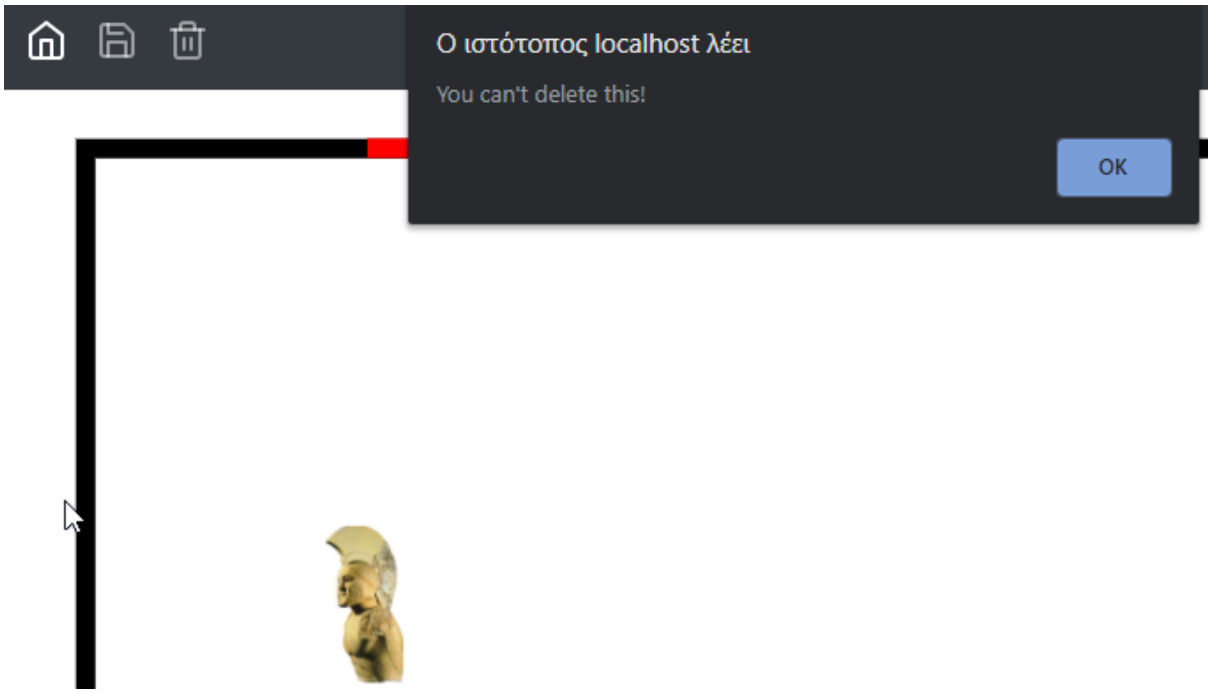


6.1.10 Διαγραφή αντικειμένου

Αν θέλετε να διαγράψετε ένα αντικείμενο από το μουσείο σας, σας δίνεται αυτή η δυνατότητα, με τη χρήση του κάδου ανακύκλωσης, ο οποίος βρίσκεται στο μενού δίπλα από το username σας, στο πάνω μέρος της οθόνης.



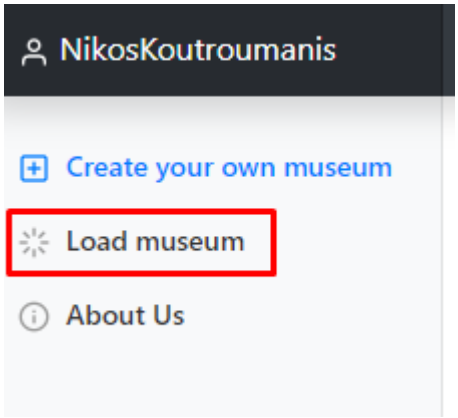
Πατώντας αυτό το κουμπί μπορείτε να διαγράψετε ένα αντικείμενο από το μουσείο σας. Με τον όρο αντικείμενο, μπορεί να εννοηθεί ένα έκθεμα, μία πόρτα ή ένας εσωτερικός τοίχος του μουσείου σας. Για να διαγράψετε ένα αντικείμενο, αφού έχετε πατήσει το παραπάνω κουμπί, απλά κλικάρετε πάνω σε αυτό. Αν κλικάρετε σε κάποιο σημείο του μουσείου, στο οποίο δεν βρίσκεται κάποιο αντικείμενο, τότε η εφαρμογή δεν εκτελεί κάποια ενέργεια και περιμένει να κλικάρετε κάποιο αντικείμενο. Επίσης, αν κλικάρετε εκτός του μουσείου, η εφαρμογή σας ενημερώνει ότι η διαγραφή δεν είναι δυνατή.



Μπορείτε να διαγράψετε μόνο ένα αντικείμενο τη φορά, οπότε για να διαγράψετε κάποιο άλλο αντικείμενο θα πρέπει να επιλέξετε και πάλι το κουμπί του κάδου ανακύκλωσης.

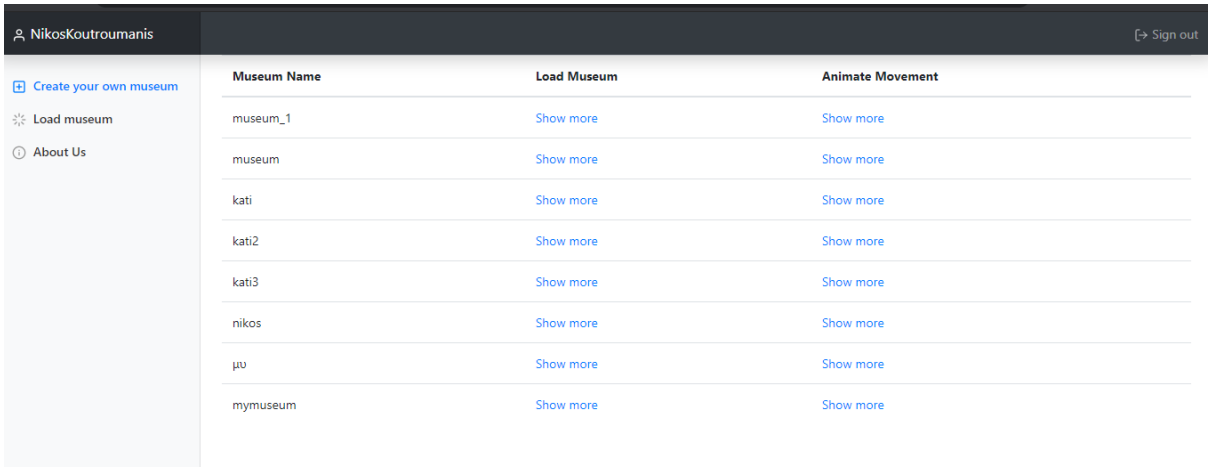
6.1.11 Επαναφόρτωση μουσείου

Αν επιθυμείτε να επαναφορτώσετε ένα ήδη αποθηκευμένο μουσείο, από τον λογαριασμό σας, αρκεί να πατήσετε το κουμπί Load Museum, από το αριστερό μενού, της αρχικής σελίδας.



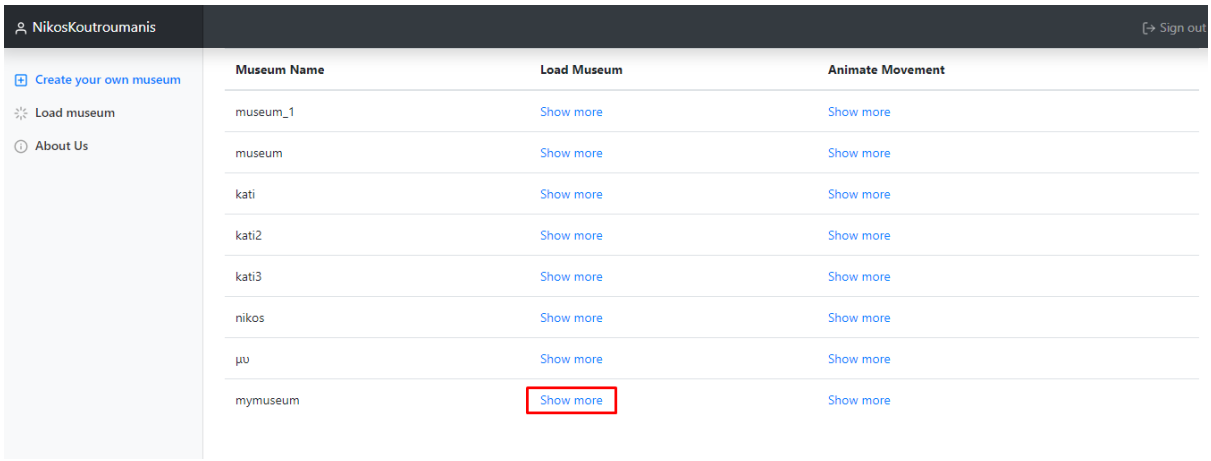
Αφού πατήσετε το παραπάνω κουμπί μεταφέρεστε σε μια νέα σελίδα που περιέχει έναν πίνακα με όλα τα μουσεία που έχετε δημιουργήσει. Από αυτόν τον πίνακα σας δίνεται η δυνατότητα για:

- Επιπρόσθετη επεξεργασία υπάρχοντος μουσείου
- Προσομοίωση κίνησης επισκεπτών μουσείου



Εικόνα 14 Λίστα μουσείων που έχει δημιουργήσει ο χρήστης

Σε αυτή τη σελίδα, μπορείτε να παρατηρήσετε όλα τα μουσεία, που έχει δημιουργήσει ο χρήστης με username NikosKoutroumanis. Για να επαναφορτώσετε κάποιο μουσείο, κάνετε κλικ στο κουμπί Show more της στήλης Load Museum, που βρίσκεται στην ίδια γραμμή με το όνομα του μουσείου.



Για παράδειγμα, αν θέλουμε να επαναφορτώσουμε το μουσείο ***mymuseum***, που κατασκευάσαμε προηγουμένως, επιλέγουμε το κουμπί Show more, όπως φαίνεται και στην εικόνα.

Αφού πατήσετε το κουμπί, που αναφέραμε, μεταφέρεστε σε μία σελίδα, η οποία περιέχει τα ίδια μενού, που αναφέραμε και στην υποενότητα 4.4, καθώς και το μουσείο που είχατε αποθηκεύσει πιο πριν στον λογαριασμό σας.

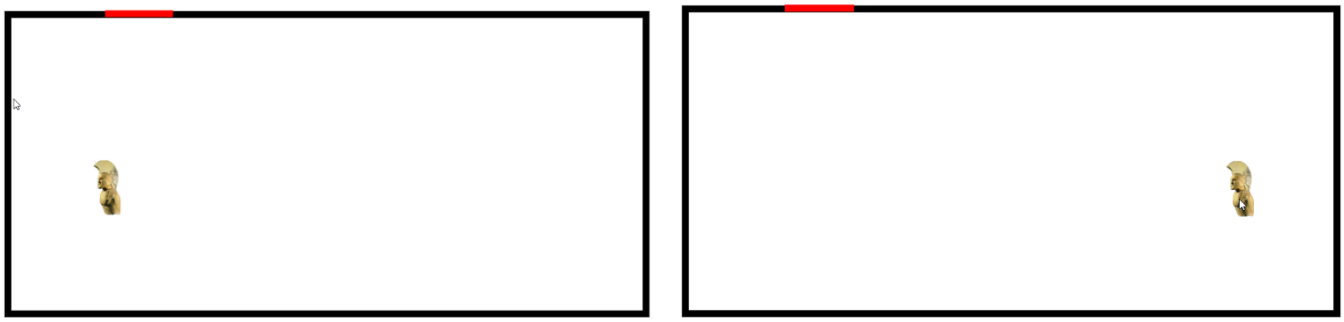
Σε αυτή την σελίδα είναι διαθέσιμες όλες οι λειτουργίες, που αναφέρθηκαν παραπάνω. Αυτό σημαίνει ότι μπορείτε να:

- Προσθέσετε καινούρια εκθέματα

- Προσθέσετε εσωτερικούς τοίχους
- Προσθέσετε πόρτα/πόρτες
- Διαγράψετε αντικείμενα
- Μετακινήσετε εκθέματα και εσωτερικούς τοίχους
- Αποθηκεύσετε εκ νέου το μουσείο

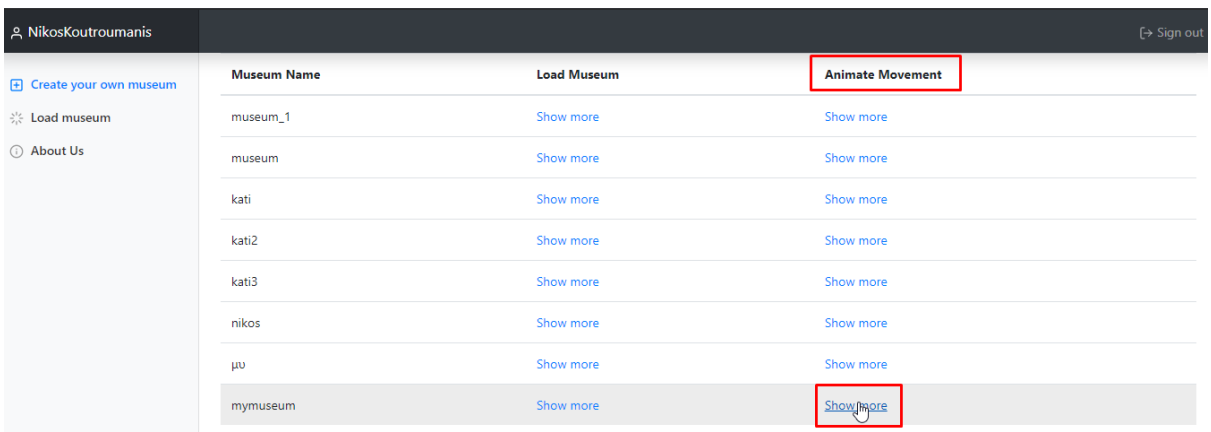
6.1.12 Μετακίνηση αντικειμένων

Σε περίπτωση που επιθυμείτε να αλλάξετε τη θέση ενός εκθέματος ή ενός εσωτερικού τοίχου, η εφαρμογή σας δίνει τη δυνατότητα μετακίνησής τους. Τα αντικείμενα μπορούν να μετακινηθούν από μια αρχική θέση σε οποιαδήποτε άλλη μέσα στον χώρο του μουσείου. Για να μετακινήσετε ένα αντικείμενο, κάνετε αριστερό κλικ πάνω του και στη συνέχεια - χωρίς να το αφήσετε - μετακινήστε τον κέρσορα προς την κατεύθυνση που επιθυμείτε. Όταν είστε ικανοποιημένοι με τη θέση του αντικειμένου, αφήστε απλά το αριστερό κλικ ελεύθερο.



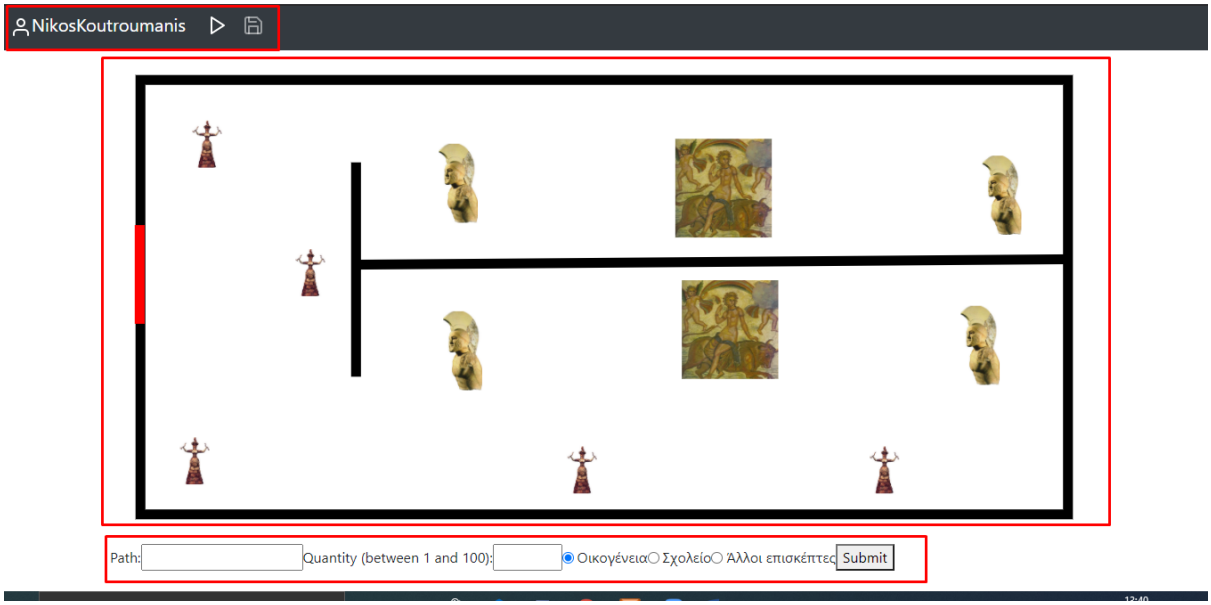
6.1.13 Προσομοίωση κίνησης

Σας δίνεται η δυνατότητα να προσομοιώσετε την κίνηση επισκεπτών στο μουσείο. Πρόκειται για ένα πολύ σημαντικό εργαλείο, το οποίο σας βοηθά στο να καταλάβετε ποια εκθέματα έχουν μεγαλύτερη επισκεψιμότητα και να συμπεράνετε αν η θέση στην οποία έχετε τοποθετήσει κάποιο έκθεμα είναι η σωστή. Αν επιθυμείτε να προσομοιώσετε κίνηση σε ένα υπάρχον μουσείο από τον λογαριασμό σας, αρκεί να πατήσετε το κουμπί Show more στην κατηγορία Animate Museum, από τη δεξιά πλευρά του πίνακα, που εμφανίζεται στην σελίδα Load Museum. Όπως αναφέραμε και παραπάνω, για να μεταφερθείτε στη σελίδα Load Museum, πρέπει να επιλέξετε το κουμπί Load Museum, από το μενού που βρίσκεται στην αριστερή πλευρά της αρχικής σελίδας.

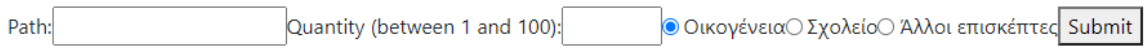


Πατώντας τον σύνδεσμο Show more εμφανίζεται μια νέα σελίδα που περιέχει:

1. Το username σας, καθώς και 2 κουμπιά για τη δημιουργία και την αποθήκευση heatmap κίνησης ως εικόνας στον υπολογιστή σας (πάνω αριστερά)
2. Την κάτοψη του μουσείου που επιλέξατε στην προηγούμενη σελίδα (κέντρο)
3. Μια φόρμα για την εισαγωγή των στοιχείων της κίνησης επισκεπτών (κάτω)

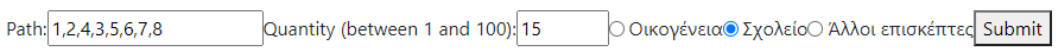


Για να προσομοιώσετε την κίνηση επισκεπτών στο μουσείο σας, πρέπει να συμπληρώσετε αυτή την φόρμα.

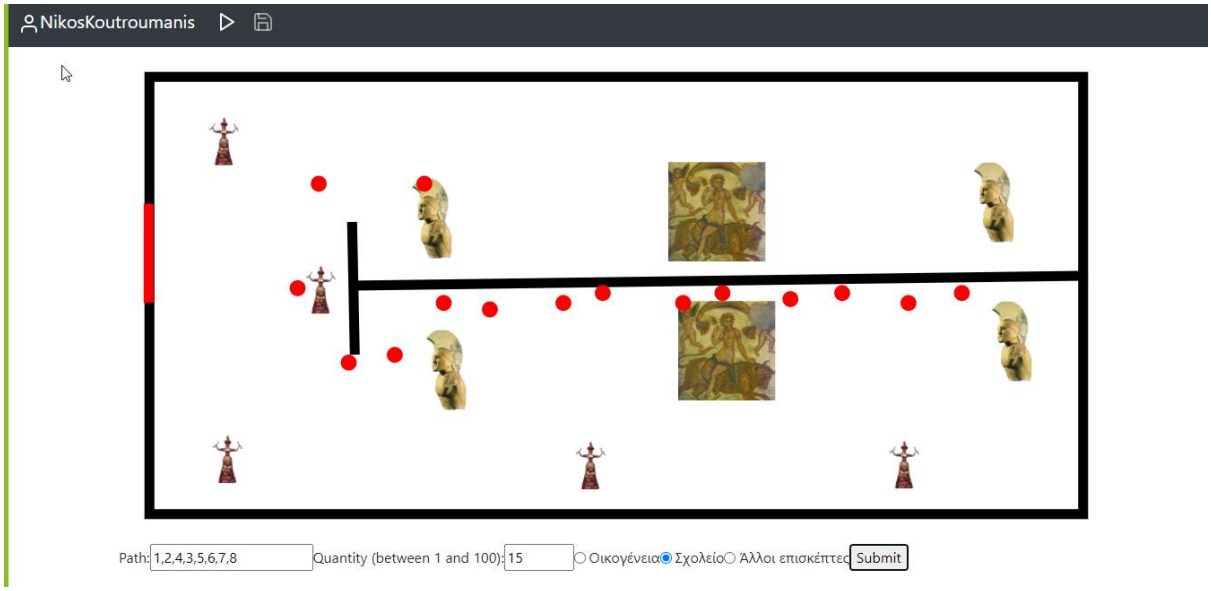


Τα στοιχεία που πρέπει να συμπληρώσετε είναι:

- Path: Το μονοπάτι που θα ακολουθήσει ο υποτιθέμενος επισκέπτης. Εισάγετε τον αριθμό κάθε εκθέματος που θα δει η ομάδα επισκεπτών. Πρέπει να ξεχωρίζετε τα εκθέματα χρησιμοποιώντας τον χαρακτήρα κόμμα(,).
- Quantity: Ο αριθμός της ομάδας των επισκεπτών (εισάγετε έναν αριθμό από το 1 μέχρι το 100)
- Η κατηγορία των ατόμων που εισέρχονται στο μουσείο. Κάθε κατηγορία θα φαίνεται στην προσομοίωση με διαφορετικό χρώμα (π.χ οικογένεια=πράσινο, σχολείο=κόκκινο, άλλοι επισκέπτες=μπλε)



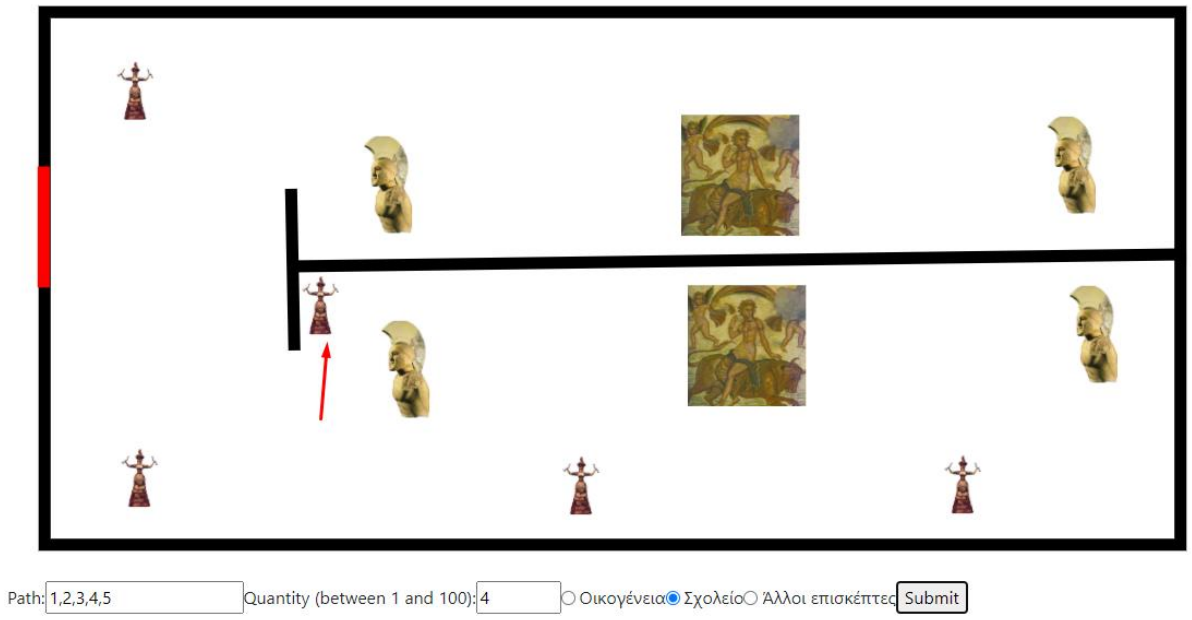
Αφού συμπληρώσετε τη φόρμα, πρέπει να κλικάρετε στο κουμπί submit, ώστε να ξεκινήσει η προσομοίωση της κίνησης, που επιλέξατε. Για παράδειγμα, με βάση τα στοιχεία που εισαγάγαμε στη φόρμα που βλέπουμε παραπάνω, μόλις κλικάρουμε το κουμπί submit, παρατηρούμε το εξής:



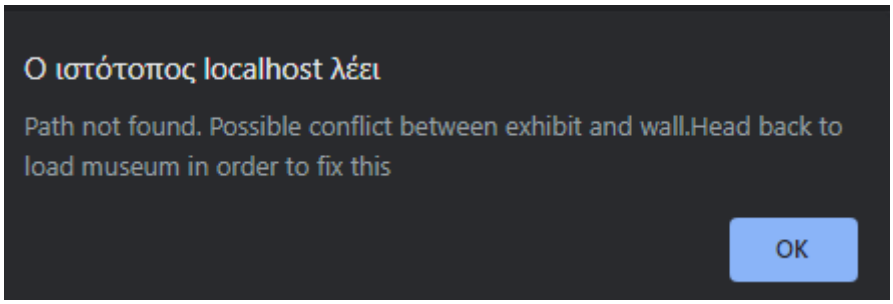
Εικόνα 15 Προσομοίωση κίνησης

Όπως βλέπετε, πλέον υπάρχουν στο μουσείο μας 15 κόκκινοι κύκλοι, που αντιπροσωπεύουν τα 15 άτομα του σχολείου, που εισαγάγαμε, πριν από λίγο. Όλα τα άτομα ακολουθούν την ίδια κίνηση, με κάποια καθυστέρηση. Αφού τελειώσουν την προκαθορισμένη κίνησή τους, όλα τα άτομα φεύγουν από το μουσείο, χρησιμοποιώντας την ανάλογη πόρτα εξόδου.

Σε περίπτωση, που το δοθέν μονοπάτι, δεν είναι δυνατό να πραγματοποιηθεί, διότι δεν υπάρχει μονοπάτι προς ή από κάποιο έκθεμα, τότε η εφαρμογή σας ενημερώνει με κατάλληλο μήνυμα λάθους, ώστε να επιστρέψετε στην προηγούμενη σελίδα (Load Museum) και να πραγματοποιήσετε τις απαραίτητες τροποποιήσεις στο μουσείο σας.



Παραπάνω παρατηρούμε ότι έχουμε τροποποιήσει ελάχιστα το προηγούμενο μουσείο μας. Συγκεκριμένα, μετακινήσαμε το έκθεμα 1, ώστε να μην είναι δυνατό να δημιουργηθεί ένα μονοπάτι από ή προς αυτό. Στη συνέχεια, εισαγάγαμε στην φόρμα, το μονοπάτι 1,2,3,4,5. Το αποτέλεσμα, που προέκυψε είναι το παρακάτω.

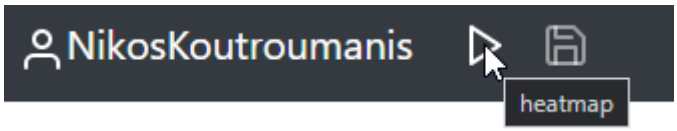


Η εφαρμογή δεν κατάφερε να δημιουργήσει ένα μονοπάτι που να περιέχει όλα τα εκθέματα που δηλώσαμε, οπότε μας ενημερώνει ότι υπάρχει πιθανόν κάποιο λάθος στην κατασκευή του μουσείου και μας προτρέπει να μεταβούμε ξανά στο Load Museum, ώστε να το διορθώσουμε.

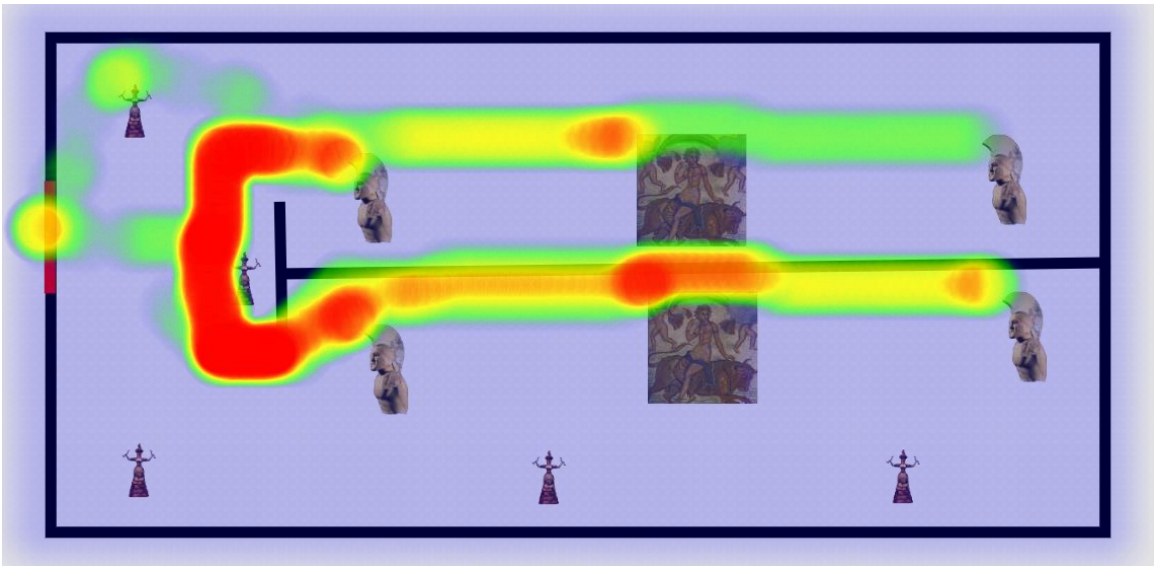
6.1.14 Δημιουργία heatmap

Αφού έχετε εισαγάγει διάφορες κινήσεις ομάδων ατόμων στο μουσείο σας, η εφαρμογή σας δίνει τη δυνατότητα δημιουργίας ενός heatmap κίνησης, το οποίο δείχνει ποια σημεία του μουσείου (εκθέματα και μη) έχουν μεγαλύτερη συχνότητα επίσκεψης.

Για να δημιουργήσετε αυτό το heatmap, αρκεί να πατήσετε το κουμπί Play, το οποίο βρίσκεται στην πάνω αριστερή πλευρά της σελίδας, δίπλα από το username σας.



Μόλις το πατήσετε, θα εμφανιστεί πάνω στο μουσείο σας, το παραγόμενο heatmap, με βάση όλες τις προηγούμενες εισαγόμενες προσομοιώσεις κίνησης.

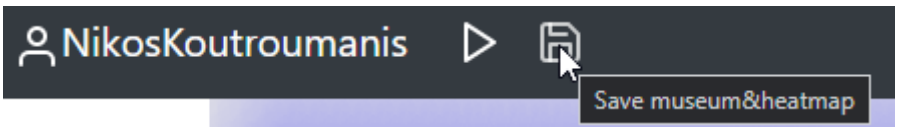


Εικόνα 16 Heatmap

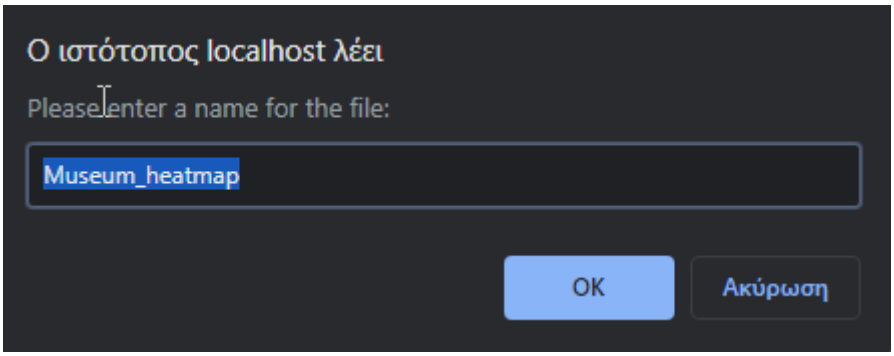
Παραπάνω βλέπουμε το μουσείο μας μαζί με το heatmap, που δημιουργήθηκε. Όσο πιο κόκκινο είναι το heatmap, τόσο μεγαλύτερη είναι η συχνότητα επίσκεψης εκείνου του σημείου. Παρατηρούμε ότι το περισσότερο επισκέψιμο σημείο του μουσείου μας βρίσκεται κοντά στην είσοδό του, εκεί που ενώνονται τα δύο επιμέρους δωμάτια, που υπάρχουν.

6.1.15 Αποθήκευση heatmap ως εικόνα

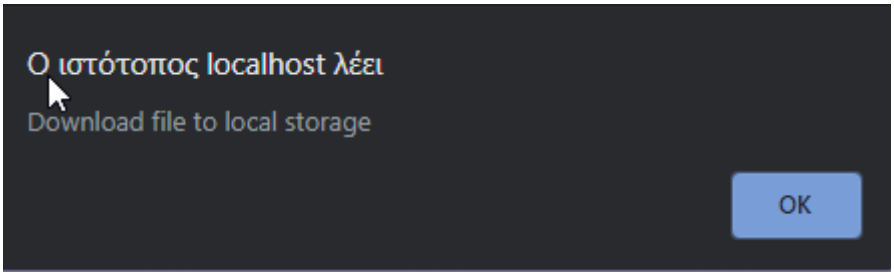
Τέλος, σας δίνεται η δυνατότητα αποθήκευσης του heatmap που μόλις δημιουργήθηκε, σε μορφή εικόνας (png). Αυτό συμβαίνει πατώντας το κουμπί της δισκέτας, που βρίσκεται στην πάνω αριστερά πλευρά της σελίδας δίπλα από το κουμπί της δημιουργίας heatmap.

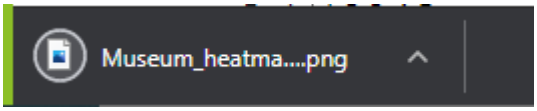


Μόλις πατηθεί το συγκεκριμένο κουμπί εμφανίζεται ένα μήνυμα στην οθόνη, που σας ρωτάει πώς θέλετε να ονομάσετε την εικόνα.



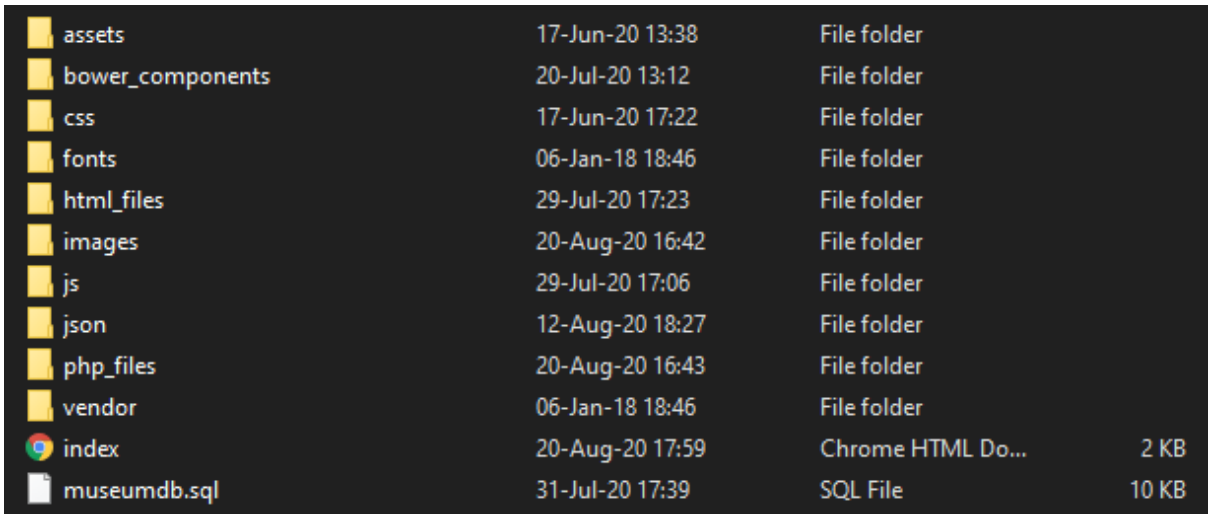
Αφού πληκτρολογήσετε το όνομα που επιθυμείτε να έχει η εικόνα και πατήσετε το κουμπί OK, η εφαρμογή σας ενημερώνει ότι το αρχείο είναι έτοιμο για λήψη στον τοπικό χώρο αποθήκευσής σας και η λήψη ξεκινάει.





6.2 ΠΑΡΑΡΤΗΜΑ ΙΙ : Δέντρο αρχείων

Παρακάτω μπορείτε να παρατηρήσετε τη δομή των αρχείων της εφαρμογής μας.



Εικόνα 17: Όλοι οι φάκελοι που δημιουργήσαμε

Όπως βλέπουμε στην εικόνα, υπάρχουν οι εξής φάκελοι:

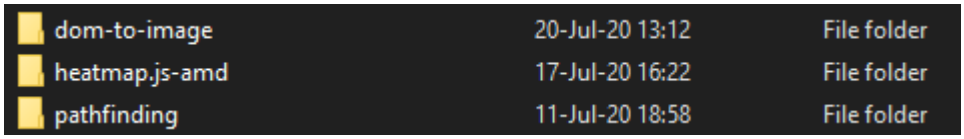
- assets: Περιέχει αρχεία, που χρησιμοποιεί το bootstrap για τη γραφική απεικόνιση της εφαρμογής
- bower_components: Περιέχει τα αρχεία των javascript βιβλιοθηκών που χρησιμοποιήσαμε στην εφαρμογή μας
- css: Περιέχει css αρχεία που χρησιμοποιούνται από την εφαρμογή μας
- fonts: Περιέχει αρχεία, που χρησιμοποιεί το bootstrap για τη γραφική απεικόνιση της εφαρμογής
- html_files: Περιέχει τη σελίδα εγγραφής χρήστη στην εφαρμογή
- images: Περιέχει όλες τις εικόνες, που υπάρχουν στην εφαρμογή μας
- js: Περιέχει όλα τα javascript αρχεία
- json: Περιέχει όλα τα json αρχεία (αποθηκευμένα μουσεία κάθε χρήστη)
- php_files: Περιέχει όλα τα php αρχεία
- vendor: Περιέχει αρχεία που χρησιμοποιεί το bootstrap για τη γραφική απεικόνιση της εφαρμογής

Επίσης υπάρχουν δύο ακόμα αρχεία, το index.html, το οποίο περιλαμβάνει την αρχική σελίδα της εφαρμογής μας (Sign in page) και το museumdb.sql, το οποίο είναι ένα αρχείο, που εξάγαμε από τη βάση, ώστε να είναι δυνατή η φορητότητα της εφαρμογής.

Στη συνέχεια, θα περιγράψουμε τι περιέχει κάθε φάκελος, εκτός από αυτούς, που χρησιμοποιούνται αποκλειστικά από το bootstrap.

6.2.1 Φάκελος Bower Components

Ο φάκελος bower_components περιέχει τους εξής 3 υποφακέλους:

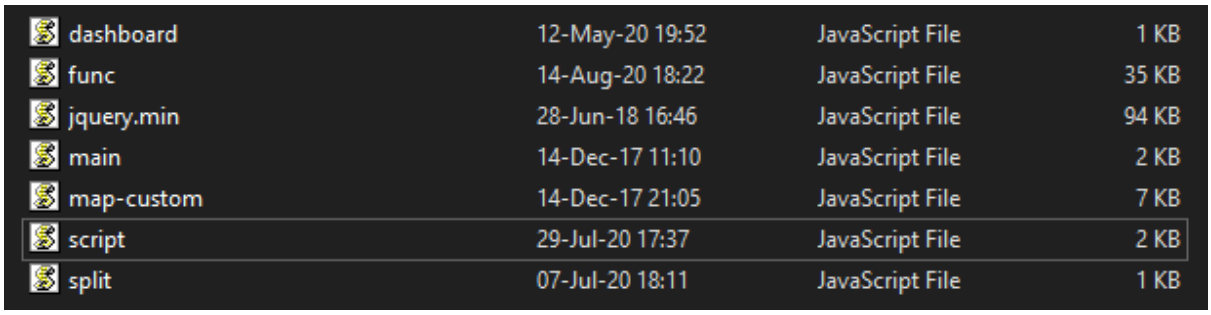


Εικόνα 18: Φάκελος Bower Components

Κάθε φάκελος είναι μία ξεχωριστή javascript βιβλιοθήκη, που εισαγάγαμε στην εφαρμογή μας, μέσω του διαχειριστή πακέτων Bower. Ας εξηγήσουμε πιο αναλυτικά, τι λειτουργίες μας προσφέρει κάθε βιβλιοθήκη.

- Dom-to-image: μας επιτρέπει να μετατρέψουμε οποιοδήποτε html element (π.χ. P, DIV, A, TABLE) σε εικόνα τύπου png. Χρησιμοποιήθηκε στην αποθήκευση των heatmap, που δημιουργούνται, στον υπολογιστή του χρήστη σε μορφή εικόνας png.
- Heatmap.js-amd: μας επιτρέπει να δημιουργήσουμε ένα heatmap, με βάση κάποια συγκεκριμένα δεδομένα, τα οποία δέχεται ως όρισμα. Χρησιμοποιήθηκε στη δημιουργία heatmap, με βάση την κίνηση των επισκεπτών.
- Pathfinding: μας επιτρέπει να υπολογίσουμε την κοντινότερη απόσταση μεταξύ 2 σημείων, χρησιμοποιώντας αλγόριθμους εύρεσης shortest path, όπως Djikstra, A*. Χρησιμοποιήθηκε για την προσομοίωση κίνησης των επισκεπτών μέσα στο μουσείο, ώστε να βρεθεί κατάλληλο μονοπάτι από το ένα έκθεμα στο άλλο.

6.2.2 Φάκελος js



Εικόνα 19: Φάκελος js

Πρόκειται για αρχεία τύπου javascript μέσα στα οποία εκτελούνται σημαντικές ενέργειες της διαδικτυακής εφαρμογής. Ας εξηγήσουμε αναλυτικότερα, τι λειτουργίες προσφέρει κάθε αρχείο:

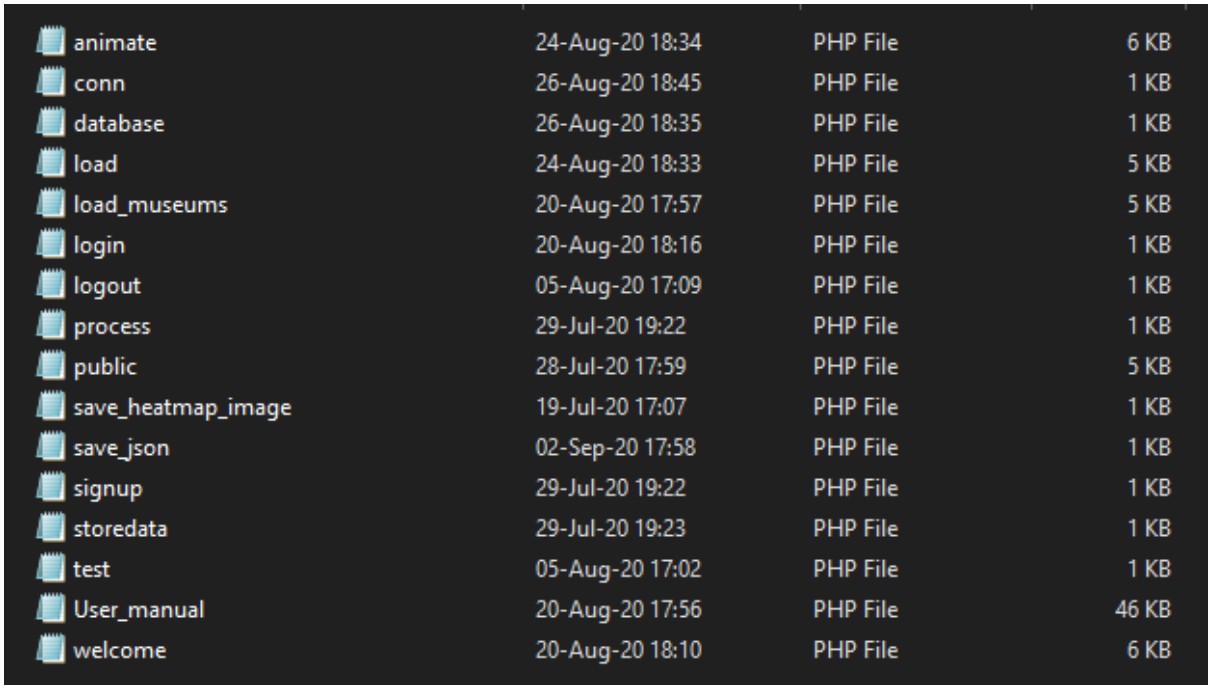
- Func.js: Σε αυτό υλοποιούνται σχεδόν όλες οι λειτουργίες της εφαρμογής μας, όπως εξηγήσαμε και παραπάνω
- Jquery.min.js: Απαραίτητο αρχείο για τη χρήση της jquery βιβλιοθήκης
- Script.js: Ελέγχει αν το username ή το email, που εισήγαγε ο χρήστης κατά την εγγραφή του, χρησιμοποιείται ήδη. Αν ναι, τον ειδοποιεί με κατάλληλο μήνυμα

Όλα τα υπόλοιπα αρχεία, χρησιμοποιούνται από το template, που επιλέξαμε μέσω του bootstrap.

6.2.3 Φάκελος Php_files

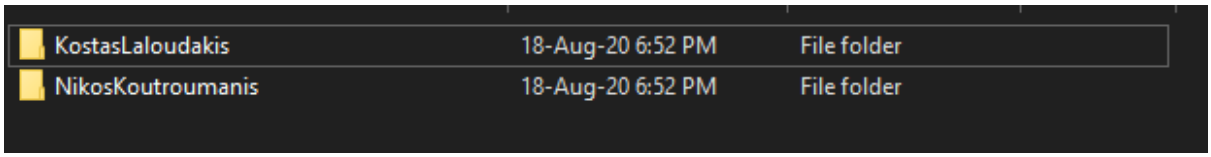
Σε αυτόν τον φάκελο, υπάρχουν όλα τα php αρχεία, που χρησιμοποιήσαμε. Τα περισσότερα από αυτά, είναι οι σελίδες που εμφανίζονται στον χρήστη. Πιο συγκεκριμένα, ας αναλύσουμε κάθε αρχείο χωριστά.

- animate: εμφανίζει τη σελίδα, που αφορά στην προσομοίωση κίνησης επισκεπτών
- conn: περιέχει τα στοιχεία σύνδεσης στη βάση
- database: περιέχει μεταβλητές, οι οποίες είναι απαραίτητες για redirect
- Load: φορτώνει το μουσείο, που επέλεξε ο χρήστης
- Load_museums: εμφανίζει όλα τα μουσεία, που έχει δημιουργήσει ο χρήστης, και του δίνει τη δυνατότητα να επαναφορτώσει κάποιο ή να προσομοιώσει την κίνηση επισκεπτών
- Login: Εισάγει τον χρήστη στην εφαρμογή αφού ελέγξει για την εγκυρότητα των στοιχείων που έδωσε
- logout: τερματίζεται η σύνδεση που δημιουργήθηκε για τον κάθε χρήστη
- process: ελέγχει αν το username ή το email, που έδωσε ο χρήστης κατά τη διάρκεια εγγραφής του χρησιμοποιείται ήδη
- public: εμφανίζει τη σελίδα δημιουργίας νέου μουσείου
- save_heatmap_image: Αποθηκεύει το heatmap του μουσείου στον υπολογιστή του χρήστη, με το όνομα, που θα επιλέξει
- save_json: Αρχείο το οποίο αποθηκεύει τα δεδομένα των μουσείων που δημιουργεί ο κάθε χρήστης
- signup: Δημιουργεί μια νέα εγγραφή στον πίνακα users της βάσης δεδομένων, με τα στοιχεία που έδωσε ο χρήστης
- Storedata: αποθηκεύει τα στοιχεία της προσομοίωσης κάθε κίνησης στον πίνακα visitors της βάσης δεδομένων
- Test: Ελέγχει αν ο χρήστης έχει ήδη αποθηκεύσει κάποιο μουσείο με το ίδιο όνομα. Χρησιμοποιείται στην αποθήκευση του json αρχείου, ώστε να είμαστε σίγουροι ότι δεν υπάρχουν δύο μουσεία με το ίδιο όνομα.
- User_manual: Εδώ εμφανίζεται το εγχειρίδιο χρήσης που είδατε και στα παραπάνω κεφάλαια
- Welcome: Η αρχική σελίδα της διαδικτυακής μας εφαρμογής



Εικόνα 20: Φάκελος php_files

6.2.4 Φάκελος json



Εικόνα 21: Φάκελος json

Με τη δημιουργία ενός νέου χρήστη αυτόματα δημιουργείται και ένας μοναδικός φάκελος με όνομα το username του. Μέσα σε κάθε φάκελο αποθηκεύονται τα μουσεία που δημιουργεί ο εκάστοτε χρήστης.

6.2.5 Φάκελος images

Εικόνα 22: Φάκελος images

Μέσα στον συγκεκριμένο φάκελο είναι αποθηκευμένες όλες οι φωτογραφίες που χρησιμοποιούνται στη δημιουργία των μουσείων (εκθέματα) και την κίνηση (άτομα). Επίσης είναι αποθηκευμένες και οι εικόνες που χρησιμοποιούνται στο εγχειρίδιο χρήσης.

Βιβλιογραφία

[1.1]. Get Programming with JavaScript, John R. Larsen, Manning Publication, 2016

[1.2]. HTML, CSS & JavaScript Web Publishing in One Hour a Day, Laura Lemay, Rafe Colburn, Jennifer Kyrnin, Sams Publishing, 2016

[1.3]. Learning JavaScript: JavaScript Essentials for Modern Application Development, Ethan Brown,

O'Reilly Media, 2016

[1.4]. Learning PHP, MySQL & JavaScript: With jQuery, CSS & HTML5, Nixon Robin, O'Reilly Media, 2018

[1.5] Ανάπτυξη διαδικτυακής εφαρμογής για τη καταχώρηση και παρακολούθηση βαθμολογιών και απουσιών εργαστηριακών μαθημάτων, Δανάσκος Χρήστος, 2017

[1.6] Ανάπτυξη διαδικτυακής εφαρμογής, αυτόματης ενημέρωσης και διάθεσης περιεχομένου, με χρήση σύγχρονων τεχνολογιών Web 2.0, Καραγιάννης Ιωάννης, 2015

[1.7] Ανάπτυξη εφαρμογής ασύρματης παραγγελιοληψίας σε περιβάλλον web. Χρήση του Mean stack & Ionic και εκτυπωτή, Πανόπουλος Κωνσταντίνος, 2019

Επιπλέον πηγές

[2.1]. https://github.com/qiao/PathFinding.js

[2.2]. https://qiao.github.io/PathFinding.js/visual/

[2.3]. https://www.patrick-wied.at/static/heatmapjs/

[2.4]. https://github.com/pa7/heatmap.js

[2.5]. https://github.com/tsayen/dom-to-image

[2.6]. https://github.com/BMPMS

Διαδικτυακές πηγές

[3.1]. https://el.wikipedia.org/wiki/JavaScript

[3.2]. https://hellenictechnologies.com/javascript/

[3.3].https://www.ip.gr/Web_Development/%CE%A4%CE%B9_%CE%B5%CE%AF%CE%BD%CE%B1%CE%B9_%CE%B7_HTML-61.html

[3.4]. http://www.petercollingridge.co.uk/tutorials/svg/interactive/dragging/

[3.5]. https://el.tipsandtrics.com/what-is-html5-how-does-it-change-way-i-browse-759304

[3.6].http://ikee.lib.auth.gr/record/288128/files/%CF%80%CF%84%CF%85%CF%87%CE%B9%CE%B1%CE%BA%CE%AE%20%CE%B5%CF%81%CE%B3%CE%B1%CF%83%CE%AF%CE%B1.pdf

[3.7] https://developer.mozilla.org/en-US/docs/Web/API/HTML_Drag_and_Drop_API

[3.8] https://jqueryui.com/

[3.9] https://getbootstrap.com/

[3.10] https://getbootstrap.com/docs/4.0/components/collapse/

[3.11] http://www.planetb.ca/syntax-highlight-word

[3.12] http://nefeli.lib.teicrete.gr/browse/stef/epp/2014/TheofilisKonstantinos/attached-document-1404397328-133932-32225/Theofilis_Konstantinos.2014.pdf

[3.13] https://el.wikipedia.org/wiki/XAMPP

[3.14]https://el.wikibooks.org/wiki/%CE%92%CE%B1%CF%83%CE%B9%CE%BA%CE%AD%CF%82_%CE%B3%CE%BD%CF%8E%CF%83%CE%B5%CE%B9%CF%82_PHP_%CE%BA%CE%B1%CE%B9_MySQL/%CE%95%CE%B9%CF%83%CE%B1%CE%B3%CF%89%CE%B3%CE%AE_%CF%83%CF%84%CE%B7%CE%BD_PHP

[3.15] https://www.ip.gr/el/dictionary/377-AJAX___Asynchronous_JavaScript_And_XML

[3.16] https://www.wlearn.gr/index.php/home-css-83

[3.17] https://dnhost.gr/kb/article/AA-00274/0/-MySQL-.html

[3.18]http://amitos.library.uop.gr/xmlui/bitstream/handle/123456789/4660/%CE%A0%CF%84%CF%85%CF%87%CE%B9%CE%B1%CE%BA%CE%AE%20%CE%B5%CF%81%CE%B3%CE%B1%CF%83%CE%AF%CE%B1%20%CE%A0%CE%B1%CF%80%CE%B1%CE%B4%CE%BF%CF%80%CE%BF%CF%8D%CE%BB%CE%BF%CF%85%20%CE%9C%CE%B1%CF%81%CE%AF%CE%B1%202018_v2.pdf?sequence=1&isAllowed=y