2022

ExamKitchen

ΑΥΤΟΜΑΤΟΠΟΙΗΣΗ ΑΞΙΟΛΟΓΙΚΗΣ ΔΙΑΔΙΚΑΣΙΑΣ ΜΕΣΩ ΓΕΝΕΣΗΣ ΘΕΜΑΤΩΝ ΒΑΣΙΣΜΕΝΗΣ ΣΕ ΟΜΑΔΕΣ

ΑΝΔΡΕΑΣ ΜΙΧΕΛΗΣ - 2022 2020 02015

Επιβλέποντες Καθηγητές: **Γρηγόριος Δημητρουλάκος, Κωνσταντίνος Βασιλάκης**
ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΛΟΠΟΝΝΗΣΟΥ – Π.Μ.Σ. ΣΤΗΝ ΕΠΙΣΤΗΜΗ ΥΠΟΛΟΓΙΣΤΩΝ

“The digital revolution is far more significant than the invention of writing or even printing. We need technology in every classroom and in every student and teacher's hand, because it is the pen and paper of our time, and it is the lens through which we experience much of our world.”

Douglas Engelbart (1925-2013)

Περιεχόμενα

ΕΥΧΑΡΙΣΤΙΕΣ III
ΣΥΝΤΟΜΟΓΡΑΦΙΕΣ IV
ΕΙΚΟΝΕΣ V
ΠΙΝΑΚΕΣ VII
ABSTRACT VIII

ΠΡΟΛΟΓΟΣ 2

Οι Προκλησεις της Νεας Ψηφιακης Εποχης 2

ΚΕΦΑΛΑΙΟ 1: FRAMEWORK 4

1.1. Εισαγωγη 4
1.2. Πυρηνικο πλαισιο 4
1.3. Βαση Παραμετρικοτητας (Parametric Object) 6
1.4. Βασικα Modules του ExamKitchen 9
1.4.1.Question Modules 11
1.4.2. Exam Modules 16
1.5. Σειριοποιηση και αποσειριοποιηση παραμετρων 23

ΚΕΦΑΛΑΙΟ 2: WEBAPP 30

2.1. Εισαγωγη 30
2.2. Βαση Δεδομενων και Ταυτοποιηση 32
2.3. Προσβαση στην Βαση δεδομενων και τα assemblies 35
2.4. Μοντελα, Όψεις και Μοντελα Προβολης 37

ΚΕΦΑΛΑΙΟ 3: WALKTHROUGH 40

3.1. Κατασκευη module pack 40
3.2. Χρηση του ExamKitchen WebApp 55

ΚΕΦΑΛΑΙΟ 4: ΠΟΡΙΣΜΑΤΑ 67

Ευχαριστίες

Η επινόηση και η πραγμάτωση μίας νέας ιδέας προϋποθέτει κόπο, χρόνο, πολλές επιτυχίες και ακόμα περισσότερες απογοητεύσεις. Το παρόν πόνημα δεν αποτελεί εξαίρεση. Ως εκ τούτου, θα ήθελα να ευχαριστήσω όλους τους καθηγητές του τμήματός Πληροφορικής και Τηλεπικοινωνιών, οι οποίοι δουλεύουν αδιάκοπα με σκοπό την μεταλαμπάδευση γνώσεων τόσο σε εμένα, όσο και τους συναδέλφους μου, καθώς και τα άτομα που μου συμπαραστάθηκαν σε αυτή τη δοκιμασία.

Ιδιαιτέρως, θα ήθελα να ευχαριστήσω θερμά:

- Τον επιβλέποντα καθηγητή μου, κ. **Γρηγόρη Δημητρουλάκο**, του οποίου η καθοδήγηση συνέβαλε καθοριστικά στην ολοκλήρωση τόσο του παρόντος πονήματος, όσο και της φοίτησής μου στο Π.Μ.Σ. Επιστήμης Υπολογιστών. Κύριε Γρηγόρη, όταν πρωτογνωριστήκαμε, είχα μια παντελώς διαφορετική εικόνα για τη συμβολή μου στην επιστήμη. Η συνεργασία μας αποτέλεσε το συνδετικό κρίκο με ένα πεδίο που μέχρι πρότινος δεν πίστευα ότι μπορώ να αγαπήσω.

- Τον συνεπιβλέποντα καθηγητή και πρόεδρο του Π.Μ.Σ., κ. **Κωνσταντίνο Βασιλάκη**, που με καθοδηγούσε ακούραστα καθόλη την διάρκεια της φοίτησής μου και πολύ συχνά μετά το πέρας αυτής.

- Την σύντροφό μου, **Έλλη**, που πλαισίωσε την προσπάθειά μου τόσο με την ανεκτίμητη συγγραφική της ικανότητα, όσο και με την ατσάλινη υπομονή της. Δεν έπαψε ούτε λεπτό να αγκαλιάζει και να υποστηρίζει πολύπλευρα κάθε μου εγχείρημα.

- Τους συμφοιτητές μου, **Ορέστη** και **Θάνο**, των οποίων η ανατροφοδότηση εξάλειψε κάθε αμφιβολία που είχα τόσο σε επίπεδο υλοποίησης, όσο και σε επίπεδο συγγραφής του πονήματος.

- Και τέλος τους γονείς μου, **Κοσμά** και **Αθανασία**, καθώς και τις αδερφές μου, **Δέσποινα** και **Σοφία**. Δεν πάψατε ούτε δευτερόλεπτο να πιστεύετε σε μένα. Ήσασταν και παραμένετε πάντοτε δίπλα μου. Δίχως το κουράγιο και τη δύναμη που μου δανείζετε, δεν ξέρω αν θα συνέχιζα να προσπαθώ.

Συντομογραφίες

CSS	**C**ascading **S**tyle **S**heet
EF	**E**ntity **F**ramework
EK	**E**xam**K**itchen
HTML	**H**yper**T**ext **M**arkup **L**anguage
IDE	**I**ntegrated **D**evelopment **E**nvironment
JS	**J**ava**S**cript
JSON	**J**ava**S**cript **O**bject **N**otation
MD5	**M**essage-**D**igest Algorithm V**5**
MOOC	**M**assive **O**nline **O**pen **C**ourse
MVVM	**M**odel - **V**iew - **V**iew**M**odel

Εικόνες

Εικόνα	Περιγραφή	Σελ.
Εικόνα 1	Διάγραμμα λειτουργίας του οικοσυστήματος ExamKitchen	5
Εικόνα 2	Γραφική απεικόνιση αλληλεπίδρασης με ένα Parametric Object	6
Εικόνα 3	Παραδείγματα στατικών, ντετερμινιστικών και Modules τυχαιότητας	12
Εικόνα 4	Σχεδιαγραμματική απεικόνιση της λειτουργικότητας της μεθόδου PrepareQuestion	18
Εικόνα 5	Σχεδιαγραμματική απεικόνιση της λειτουργικότητας της μεθόδου PrepareAnswer	19
Εικόνα 6	Σχεδιαγραμματική απεικόνιση της λειτουργικότητας της μεθόδου PrepareExam	21
Εικόνα 7	Γραφική αποτύπωση του τρόπου δημιουργίας ομάδων, συμπεριλαμβανομένου ψευδοκώδικα	22
Εικόνα 8	Παράδειγμα αλληλεπίδρασης με έναν Μετατροπέα Παραμέτρου	24
Εικόνα 9	Γραφική απεικόνιση της αλληλοσυμβατότητας μεταξύ Parse και Serialize	26
Εικόνα 10	Δομή του ExamKitchen WebApp	31
Εικόνα 11	Γραφική αναπαράσταση της χαρτογράφησης του Entity Framework	32
Εικόνα 12	Μοντέλο Οντοτήτων-Συσχετίσεων της βάσης δεδομένων του ExamKitchen WebApp	34
Εικόνα 13	Διαδικασία παροχής των Services στις σελίδες που τα χρειάζονται	36
Εικόνα 14	Σχεδιαγραμματική απεικόνιση των Models, Views και ViewModels του ExamKitchen WebApp	39
Εικόνα 15	Ενσωμάτωση του ExamKitchen Core σε καινούριο Project C#	40
Εικόνα 16	Αρχική δομή του Visual Studio Project	41
Εικόνα 17	Το MyExamTemplate πριν την υλοποίηση	41
Εικόνα 18	Οι παράμετροι του MyExamTemplate	42
Εικόνα 19	Μέθοδος παραμετροποίησης του MyExamTemplate	43
Εικόνα 20	Μέθοδος προετοιμασίας θεμάτων του MyExamTemplate	44
Εικόνα 21	Μέθοδος προετοιμασίας ερώτησης του MyExamTemplate	45

Εικόνα	Περιγραφή	Σελ.
Εικόνα 22	Μέθοδος προετοιμασίας απάντησης του MyExamTemplate	46
Εικόνα 23	Δομή MySumsQuestion	47
Εικόνα 24	Κλάση MulChoiceDataContainer	48
Εικόνα 25	Μετατροπέας MulChoiceConverter	49
Εικόνα 26	Μετατροπέας MulChoiceConverter - Σειριοποίηση και αποσειριοποίηση	50
Εικόνα 27	Μετατροπέας MulChoiceConverter - Συντακτική ανάλυση	51
Εικόνα 28	Κλάση MyMulChoiceQuestion - Δήλωση παραμέτρων	52
Εικόνα 29	Κλάση MyMulChoiceQuestion - Υλοποίηση της μεθόδου γένεσης θεμάτων	53
Εικόνα 30	Τελική δομή του Project	54
Εικόνα 31	Διαδικασία δημοσίευσης του module pack	54
Εικόνα 32	Δημιουργία νέου χρήστη στο ExamKitchen WebApp	55
Εικόνα 33	Νέο Menu κατόπιν σύνδεσης του χρήστη και η αρχική οθόνη των Assemblies	56
Εικόνα 34	Φόρτωση του νεοσύστατου module pack στο WebApp	57
Εικόνα 35	Σελίδα Projects και δημιουργία ενός καινούριου Project	58
Εικόνα 36	Σελίδα επεξεργασίας Project	59
Εικόνα 37	Σελίδα επεξεργασίας Project μετά την επιλογή Exam Module	60
Εικόνα 38	Σελίδα επεξεργασίας Project, κατόπιν προσθήκης ερωτήσεων	61
Εικόνα 39	Παράδειγμα ανάδρασης σε περίπτωση λάθους των παραμέτρων	62
Εικόνα 40	Αποθήκευση των αλλαγών του Project και η είσοδος στην σελίδα γένεσης θεμάτων	63
Εικόνα 41	Σελίδα Γένεσης Θεμάτων	63
Εικόνα 42	Λήψη και αποσυμπίεση των νεόδμητων θεμάτων	64
Εικόνα 43	Αποτελέσματα γένεσης θεμάτων με την βοήθεια του ExamKitchen (Group 1)	65
Εικόνα 44	Αποτελέσματα γένεσης θεμάτων με την βοήθεια του ExamKitchen (Group 2)	66

Πίνακες

Πίνακας	Περιγραφή	Σελ.
Πίνακας 1	Παραδείγματα αποτελεσμάτων της μεθόδου PrepareQuestion για ερωτήσεις με και χωρίς υποερωτήματα	18-19
Πίνακας 2	Παραδείγματα αποτελεσμάτων της μεθόδου PrepareAnswer για ερωτήσεις με και χωρίς υποερωτήματα	20
Πίνακας 3	Σειριοποίηση και αποσειριοποίηση απλών δομών δεδομένων	23
Πίνακας 4	Παράδειγμα απόκρυψης του γνωρίσματος τύπου της παραμέτρου	24
Πίνακας 5	Πίνακας μετατροπής IntParameterConverter	26
Πίνακας 6	Πίνακας μετατροπής FloatParameterConverter	27
Πίνακας 7	Πίνακας μετατροπής DoubleParameterConverter	27
Πίνακας 8	Πίνακας μετατροπής StringParameterConverter	28
Πίνακας 9	Πίνακας μετατροπής BoolParameterConverter	28
Πίνακας 10	Πίνακας μετατροπής EnumParameterConverter	29
Πίνακας 11	Πίνακας μετατροπής JsonParameterConverter	30

Abstract

During the COVID-19 pandemic remote learning was rendered the only viable means of instructional delivery, introducing new challenges and highlighting the necessity for new techniques, able to safeguard the academic integrity of the examination process. This article introduces a piece of software able to batch-generate unique - albeit equivalent - examination sheets. ExamKitchen relies on Modules created by its users. Based on the purposes - and the creativity! - of the examiner, each Module represents a different type of question (e.g., multiple choice, matching, development, etc.) and acts as a template based on which users can generate unique problems and their corresponding answers. By open sourcing its object-oriented framework, ExamKitchen is scalable and allows each user to implement their own, highly parameterized Modules, thus actualizing a variety of scenarios.

Our proposal was tested during examining “Software Architecture” and “Compilers” of the Informatics and Telecommunications department in the University of Peloponnese, two courses plagued by integrity vulnerabilities during the pandemic with promising results. The success rates assimilated those of previous - traditional - exams. When compared with other, oftentimes high-cost commercial tools, ExamKitchen equips examiners with more sophisticated randomization methods, while significantly decreasing the preparation-and-evaluation time cost of the examination process by offering a strong and reliable automation procedure that facilitates and expedites the tests’ correction.

Πρόλογος

Οι Προκλήσεις της Νέας Ψηφιακής Εποχής

Εάν η μάθηση είναι μια σχετικά μόνιμη αλλαγή της συμπεριφοράς που πηγάζει από την εμπειρία, η εκπαίδευση ταυτίζεται με το είδος της εμπειρίας που τροποποιεί σχετικά μόνιμα τη συμπεριφορά. Από την απομνημόνευση των προφορικών παραδόσεων των προϊστορικών κοινωνιών και την ίδρυση της πλατωνικής Ακαδημίας έως τη συγκρότηση των σύγχρονων εκπαιδευτικών θεσμών, ο τρόπος διάχυσης γνώσεων, ικανοτήτων και παραδόσεων στις ανθρώπινες κοινωνίες έχει ριζικά αλλάξει. Θα μπορούσε να ειπωθεί ότι η ψηφιακή επανάσταση έχει υποκινήσει μια δεύτερη, εκπαιδευτική: Το πάλαι ποτέ προνόμιο των λίγων, εύπορων και βασικά αρρένων μελών των κοινωνιών πλέον αποτελεί ένα πανταχού προσβάσιμο αγαθό που προϋποθέτει μόνο μια σταθερή σύνδεση στο διαδίκτυο. Όπως ισχυρίστηκε ο Douglas Engelbart, η τεχνολογία «είναι το χαρτί και το μολύβι της εποχής μας».

Μολονότι η ενδελεχής διερεύνηση της συμβολής του διαδικτύου στο σύγχρονο εκπαιδευτικό σύστημα υπερβαίνει τους σκοπούς της εν λόγω εργασίας, η εμφιλοχώρηση της τεχνολογίας σε σχολεία και πανεπιστήμια, εμπλουτίζει μεταξύ άλλων, τους διαθέσιμους πόρους μάθησης, πολλαπλασιάζει τις πηγές γνώσεις, εισάγει στοιχεία διαδραστικότητας και, φυσικά, επιτρέπει την εξ αποστάσεως επικοινωνία και διδασκαλία. Κατά τα φαινόμενα, η τεχνολογικά υποβοηθούμενη εκπαίδευση αμβλύνει τη μάστιγα της προσβασιμότητας και προασπίζεται τα άνευ όρων δικαιώματα των ατόμων στη γνώση. Η καθοδήγηση και η επίβλεψη των μαθητών είναι άμεση, ουσιαστική και εξατομικευμένη, η κατανόηση άλλοτε αφηρημένων και σύνθετων εννοιών διευκολύνεται με πολλαπλές μεθόδους απεικόνισης, ενώ ενθαρρύνεται η συνεργασία τόσο μεταξύ των μαθητών όσο και των μαθητών με το διδακτικό προσωπικό.

Η υγειονομική κρίση των τελευταίων ετών φωταγώγησε εις βάθος την ωφελιμότητα της τεχνολογίας στους εκπαιδευτικούς θεσμούς. Στην πραγματικότητα, κατά τη διάρκεια της πανδημίας, η απομακρυσμένη ή διαδικτυακή εκπαίδευση (remote or online learning) - το είδος, δηλαδή, του εκπαιδευτικού πλαισίου που επιτρέπει σε διδάσκοντα και διδασκόμενους να χωρίζονται χωροχρονικά - καθιερώθηκε ως ο μοναδικός τρόπος απρόσκοπτης διδασκαλίας. Ταυτόχρονα, ωστόσο, η απουσία υποδομών ικανών να υποστηρίξουν την εν λόγω μετάβαση ανέδειξε πληθώρα προβλημάτων: Για παράδειγμα, η διαμονή στο σπίτι συνοδεύτηκε από περισπασμούς και τεχνικά προβλήματα, ενώ διευκόλυνε ασεβείς συμπεριφορές τόσο προς το

διδακτικό προσωπικό, όσο και προς τους συνομήλικους, φθείροντας την αποτελεσματικότητα και την πλαισίωση των ψηφιακών διαλέξεων. Επιπρόσθετα, η έλλειψη ενός ελεγχόμενου περιβάλλοντος επέτρεψε στους διδασκόμενους να υπονομεύσουν ποικιλοτρόπως την ακεραιότητα των αξιολογήσεων της προόδου τους και κατέστησε τους καθηγητές ανίκανους να αποτρέψουν φαινόμενα αντιδεοντολογικών συμπεριφορών.

Η ελεύθερη ροή πληροφοριών στο διαδίκτυο συγκαταλέγεται στις κυριότερες προκλήσεις της σύγχρονης εκπαίδευσης καθώς - μεταξύ άλλων - διαβρώνει την ακαδημαϊκή ακεραιότητα και διευκολύνει την ακαδημαϊκή απάτη. Η ακαδημαϊκή ακεραιότητα ορίζεται ως ένα πολύπλοκο σύστημα αξιών, πρακτικών και προτύπων που καθορίζει τη λήψη αποφάσεων και την ανάληψη δράσεων στην εκπαίδευση και την έρευνα, ενώ η ακαδημαϊκή ανεντιμότητα περιγράφει τις ηθικά μεμπτές συμπεριφορές ατόμων ή ιδρυμάτων που την παραβιάζουν. Γενικά, η ακαδημαϊκή απάτη αφορά ενέργειες που αποσκοπούν στην εξαπάτηση ή/ και την κατάκτηση πλεονεκτήματος μέσω της παραβίασης των ακαδημαϊκών κανονισμών και περιλαμβάνει την εκούσια ή ακούσια λογοκλοπή, την εξαπάτηση (π.χ. την αντιγραφή στις εξετάσεις), την κατασκευή και, τέλος, την υποβοήθηση λοιπών ανέντιμων συμπεριφορών.

Οι ανέντιμες συμπεριφορές στο εκπαιδευτικό σύστημα από διδασκόμενους και διδάσκοντες όχι μόνο εξαλείφουν την αξιοκρατία στην εκπαίδευση και, μετέπειτα, στην αγορά εργασίας, αλλά διαβρώνουν συλλήβδην την φήμη και την αξιοπιστία των θεσμών και των αποφοίτων τους. Συνυπολογίζοντας τα άνωθεν, το ακόλουθο πόνημα προτείνει μία μέθοδο αυτοματοποίησης και διευκόλυνσης των εξ-αποστάσεως εξετάσεων με γνώμονα την εξασφάλιση της ακαδημαϊκής αξιοπρέπειας. Το προταθέν εγχείρημα αποτελεί ένα οικοσύστημα ικανό να παράγει μαζικά ισοδύναμα ζεύγη θεμάτων/ απαντήσεων. Ως εκ τούτου, προσφέρει μια ιδιαίτερα εξατομικευμένη, αλλά ισόνομη με τους συνεξεταζόμενους εμπειρία αξιολόγησης για κάθε εξεταζόμενο, ανακουφίζει χρονικά τους βαθμολογητές, ενώ ταυτόχρονα προστατεύει τους φοιτητές από τα λάθη του ανθρώπινου παράγοντα στις διορθώσεις και, επομένως, περιορίζει την ανάγκη για αναβαθμολόγηση.

Κεφάλαιο 1: Framework

1.1. Εισαγωγή

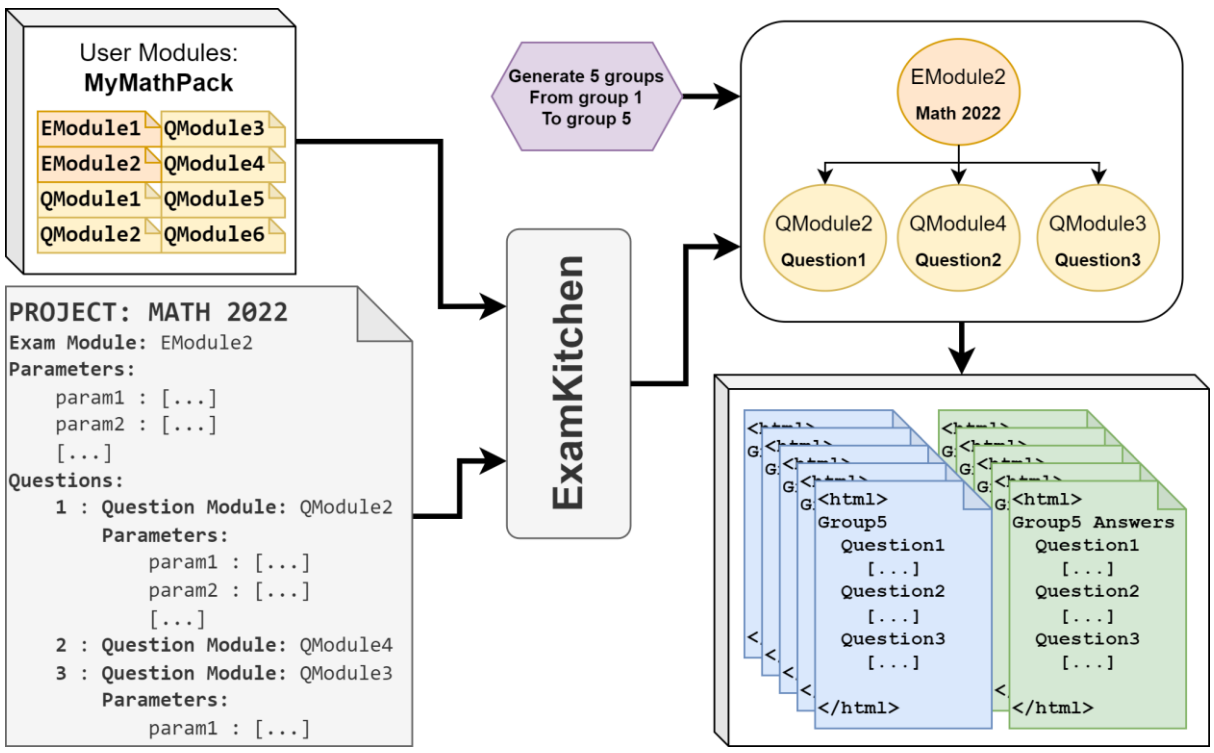
Στην πραγματικότητα, πολύ πριν την έναρξη της πανδημίας, η ψηφιοποίηση των Κέντρων Διά Βίου Μάθησης (Κε.Δι.Βι.Μ.) εθνικών και διεθνών πανεπιστημίων -άλλοτε ως ανεξάρτητες μονάδες και άλλοτε υπό την αιγίδα μαζικών παρόχων ανοιχτών διαλέξεων (massive online open course ή MOOC)- ανέδειξε την ανάγκη για την φερέγγυα αξιολόγηση των συμμετεχόντων. Το ExamKitchen (ΕΚ) είναι ένα σύνολο εφαρμογών και εργαλείων αυτοματοποίησης και ψηφιακής υποβοήθησης γραπτών εξετάσεων. Δεν πρόκειται για χώρο ή μέθοδο αξιολόγησης, αλλά για ένα οικοσύστημα δόμησης, προετοιμασίας και υποστήριξής της, που στηρίζεται σε ένα πλαίσιο (framework) και μια συνοδευτική διαδικτυακή εφαρμογή (WebApp). Το πλαίσιο είναι η ραχοκοκαλιά του εγχειρήματος και σε προγραμματιστικό επίπεδο επιτρέπει τη δημιουργία προτύπων ερωταπαντήσεων. Σε δεύτερο χρόνο μέσω του WebApp, οι χρήστες της εφαρμογής σχεδιάζουν την εξέταση εκμεταλλευόμενοι τα υπάρχοντα πρότυπα.

1.2. Πυρηνικό πλαίσιο

Προς διασφάλιση της ακαδημαϊκής ακεραιότητας, το ExamKitchen υιοθετεί και συστήνει ένα μοντέλο εξέτασης βασισμένο σε ετερογενείς, πλην ισοδύναμες ομάδες θεμάτων δίχως να περιορίζεται σε αυτό. Μολονότι η κατάτμηση των εξεταζόμενων σε ομάδες παραδοσιακά συνδέεται με την ενίσχυση της φερεγγυότητας των εξετάσεων, ελλοχεύει μία σειρά περιορισμών που υπονομεύουν την αποτελεσματικότητά της. Σε αυτούς συγκαταλέγεται ο υψηλός χρόνος προετοιμασίας και ο κατά συνέπεια περιορισμένος αριθμός ομάδων, καθώς ο αξιολογητής καλείται να επινοήσει, να επιλύσει, να διορθώσει ή να αναθεωρήσει ένα σημαντικό αριθμό ομοιογενών και ισοδύναμων ως προς τη δυσκολία τους προβλημάτων. Για την εξάλειψη τέτοιων περιορισμών, το εγχείρημά μας συνοδεύει παράλληλα κάθε ομάδα θεμάτων που παράγεται με ένα ακόμα φύλλο που διατηρεί τις απαντήσεις στις ερωτήσεις που περιλαμβάνει.

Ευελιξίας ένεκα, το ΕΚ είναι σχεδιασμένο σπονδυλωτά (**modular design**): στο framework φιλοξενείται ένα σύνολο διεπαφών (**interfaces**) και αφηρημένων κλάσεων (**abstract classes**) που εγκαθιδρύουν τα απαραίτητα δομικά πρότυπα. Μολονότι τα στοιχεία

αυτά υποδεικνύουν και θεμελιώνουν τις αναγκαίες για την λειτουργία τους μεθόδους, η λειτουργικότητα που ενέχουν βασίζεται σε **crowdsourcing**: την διαδικασία κατά την οποία ένα λογισμικό συντηρείται και εμπλουτίζεται από τρίτους με γνώσεις προγραμματισμού. Εθελοντές προγραμματιστές μπορούν να συνεισφέρουν στην εκπαιδευτική κοινότητα με δημοσίως διαθέσιμα δομοστοιχεία (Modules), ενώ εκπαιδευτικά ιδρύματα μπορούν σχεδιάσουν και υλοποιήσουν εξατομικευμένες ερωτήσεις και πρότυπα που ικανοποιούν εξ ολοκλήρου τις ανάγκες τους. Έτσι, η σχεδιαστική λογική προσδίδει μεγάλη ελαστικότητα ως προς τη φύση, την ετερογένεια και την πολυπλοκότητα των ερωτήσεων, ενώ παράλληλα επιτρέπει σε χρήστες και συνεισφέροντες (**contributors**) να βελτιστοποιήσουν την χρηστικότητα του πλαισίου.

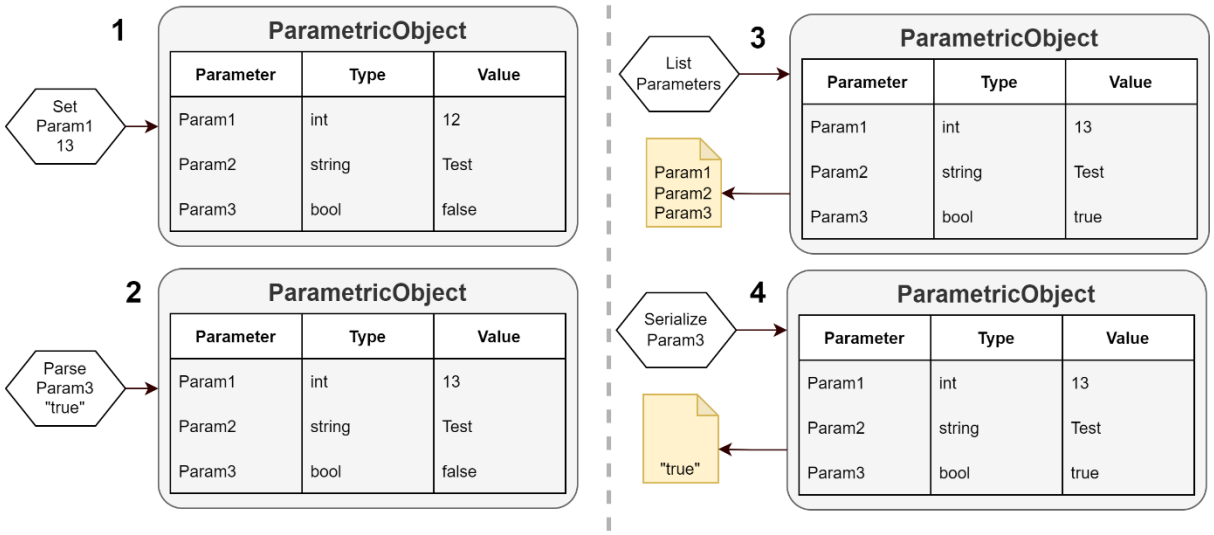


Εικόνα 1: Διάγραμμα λειτουργίας του οικοσυστήματος ExamKitchen

Η ευελιξία του ΕΚ περεταίρω ενισχύεται από την παραμετροποίηση των δομοστοιχείων κατά την χρήση. Η δυνατότητα αυτή προσδίδει επαναχρηστικότητα στα υπάρχοντα Modules, καθώς μπορούν να καλύψουν ένα ετερογενές φάσμα σεναρίων. Η υποστήριξη της παραμετροποίησης περιγράφεται και υλοποιείται μέσω της αφηρημένης κλάσης **Parametric Object**.

1.3. Βάση Παραμετρικότητας (Parametric Object)

Η κλάση Parametric Object εμπλουτίζει τα αντικείμενα που την κληρονομούν τόσο με παραμέτρους, όσο και με την απαραίτητη λειτουργικότητα για την απαρίθμηση (enumeration), ανάθεση (assignment), ανάκτηση (retrieval), σειριοποίηση (serialization) και αποσειριοποίησή (de-serialization) τους.



Εικόνα 2: Γραφική απεικόνιση αλληλεπίδρασης με ένα Parametric Object

Οι παράμετροι που παρέχουν τα Parametric Objects βασίζονται σε δημόσιες (public) ή προστατευμένες (protected) ιδιότητες (properties) αντικειμένων. Κάθε ιδιότητα με *Χαρακτηριστικό Περιγραφής Παραμέτρου* (**Parameter Definition Attribute**· σημειώνεται ως **[ParameterDefinition]**) θεωρείται και χρησιμοποιείται ως παράμετρος. Το Parameter Definition Attribute διατηρεί διάφορα μεταδεδομένα της παραμέτρου, όπως ο τύπος, η περιγραφή, η προεπιλεγμένη τιμή και οδηγίες σειριοποίησης και αποσειριοποίησής της. Τα ορίσματα (arguments) που απαιτεί χωρίζονται σε βασικά και προαιρετικά. Στα βασικά ορίσματα του Attribute συγκαταλέγονται τα:

- **Type**

  Το όρισμα *Type* συγκρατεί τον τύπο δεδομένων της παραμέτρου (π.χ. `int`, `string`, `bool`, `MyObject` κλπ.)

- **Description**

  Το όρισμα *Description* διατηρεί ένα σύντομο επεξηγηματικό κείμενο σε σύνταξη MarkDown που παρέχει πληροφορίες σχετικά με τον τρόπο σύνταξης και χρήσης της.

- **Default Value**

  Το όρισμα *Default Value* αποθηκεύει σε σειριοποιημένη μορφή την προεπιλεγμένη τιμή της παραμέτρου. Η τιμή ανατίθεται αυτόματα στην παράμετρο κατόπιν αποσειριοποίησης, ενόσω αρχικοποιείται ένα νέο αντικείμενο που την περιέχει.

Όσον αφορά τα προαιρετικά ορίσματα, αυτά περιλαμβάνουν:

- **Parameter Converter Type**

  Το όρισμα *Parameter Converter Type* συγκρατεί τον τύπο *Μετατροπέα Παραμέτρου* (*ParameterConverter*) που θα αναλύσουμε σε βάθος αργότερα (βλ. 1.5. Σειριοποίηση και αποσειριοποίηση παραμέτρων, σελ. 23). Σε γενικές γραμμές, είναι μια υποστηρικτική δομή υπεύθυνη για την σειριοποίηση και αποσειριοποίηση των παραμέτρων.

- **Parameter Converter Construction Arguments**

  Το όρισμα *Parameter Converter Construction Arguments* επιτρέπει τον ορισμό των γνωρισμάτων παρασκευής του δηλωθέντος Μετατροπέα Παραμέτρου, εφόσον απαραίτητα.

Όταν ένα Parametric Object δημιουργείται, ανακτά εσωτερικά μία λίστα με τις περιεχόμενες παραμέτρους του. Οι παράμετροι αρχικοποιούνται με την προεπιλεγμένη τιμή που δηλώνεται στο Parameter Definition Attribute τους. Η νεοσύστατη λίστα αποθηκεύεται

εντός του αντικειμένου, ώστε να χρησιμοποιηθεί ως πίνακας αναφοράς (lookup table) για την υποβοήθηση της λειτουργικότητας του παραμετρικού αντικειμένου. Με τη σειρά της, η εν λόγω λειτουργικότητα αποτελείται από τέσσερις μεθόδους (methods):

- `object GetParameter (string paramName)`

  Η μέθοδος *GetParameter* επιστρέφει υπό μορφή αντικειμένου την τιμή της παραμέτρου με όνομα που δηλώνεται στο όρισμα `paramName`. Εάν δεν βρεθεί παράμετρος με το δοθέν όνομα, επιστρέφεται η τιμή `null`.

- `void SetParameter (string paramName, object paramValue)`

  Η μέθοδος *SetParameter* αναθέτει στην παράμετρο με όνομα που δηλώνεται στο όρισμα `paramName` την τιμή που δηλώνεται στο όρισμα `paramValue`. Σε περίπτωση που η δοθείσα τιμή είναι ασύμβατη με το αντικείμενο ή δεν εντοπιστεί παράμετρος με το δοθέν όνομα, η ανάθεση αποτυγχάνει σιωπηλά, δίχως πρόκληση εξαίρεσης και χωρίς να γνωστοποιηθεί στο χρήστη.

- `string SerializeParameter (string paramName)`

  Η μέθοδος *SerializeParameter* επιστρέφει ως συμβολοσειρά (`string`) τη σειριοποιημένη τιμή της παραμέτρου με όνομα που δηλώνεται στο όρισμα `paramName`. Εφόσον δεν βρεθεί παράμετρος με το δοθέν όνομα ή η σειριοποίηση της τιμής της αποτύχει, επιστρέφεται μία άδεια συμβολοσειρά.

- `void ParseParameter (string paramName, string paramValue)`

  Η μέθοδος *ParseParameter* αναθέτει, κατόπιν αποσειριοποίησης, την τιμή που περιγράφεται στο όρισμα `paramValue` της παραμέτρου με όνομα που καταχωρείται στο όρισμα `paramName`. Εάν η αποσειριοποίηση κριθεί ανέφικτη ή δεν ανακτηθεί παράμετρος με το δοθέν όνομα, η ανάθεση αποτυγχάνει σιωπηλά.

Προς ενίσχυση της προσαρμοστικότητας και της ευχρηστίας των παραμετρικών αντικειμένων, το ExamKitchen προσφέρει ένα σύνολο επεκτάσεων *αντανάκλασης* (Reflection) που παρέχουν δυνατότητες ανακάλυψης και απαρίθμησης παραμετρικών κλάσεων.

- `bool IsParametric (this Type type)`

  Η μέθοδος *IsParametric* διευκρινίζει εάν ο τύπος αντικειμένου από τον οποίο κλήθηκε αντιπροσωπεύει μια μη αφηρημένη κλάση, που κληρονομεί -άμεσα ή έμμεσα- τη λειτουργικότητα των παραμετρικών αντικειμένων.

- `ParamInfo[]? GetParameters (this Type type)`

  Η μέθοδος *GetParameters* επιστρέφει μία λίστα με τις παραμέτρους που ενέχει ο τύπος αντικειμένου από το οποίο καλείται. Εάν ο τύπος αντικειμένου δεν πληροί τα κριτήρια που περιγράφονται στην μέθοδο *IsParametric*, επιστρέφεται η τιμή `null`.

1.4. Βασικά Modules του ExamKitchen

Η αλληλεπίδραση με το πλαίσιο του ExamKitchen έγκειται στον σχεδιασμό **projects**. Τα projects είναι διφυή και καθορίζονται από το είδος της γραπτής εξέτασης και το περιεχόμενο των ερωταπαντήσεων. Κάθε project αποτελείται από ένα Exam Module και τουλάχιστον ένα Question Module. Τα Exam Modules καθορίζουν την όψη και την δομή των εξετάσεων. Εκπροσωπούν λειτουργικά τη ρίζα ενός project και συμπεριφέρονται ως γεννήτρια (ομάδων) θεμάτων. Τα Question Modules ενέχουν την απαραίτητη λειτουργικότητα για την παραγωγή των εκφωνήσεων των δοκιμασιών και των απαντήσεων που τους αντιστοιχούν. Πρόκειται για αφηρημένες κλάσεις που προσφέρονται από το πλαίσιο και επιτρέπουν στους προγραμματιστές να δημιουργούν εξατομικευμένες στις ανάγκες της εξέτασης υλοποιήσεις.

Κατόπιν υλοποίησης, τα νεοσύστατα Modules εμφωλεύονται από τους προγραμματιστές σε assemblies που έχουν την μορφή *Βιβλιοθήκης Δυναμικής Σύνδεσης* (**Dynamic Link Library**· αρχεία με κατάληξη **DLL**) ή *Πακέτου Επέκτασης NuGet* (**NuGet Extension Pack**· αρχεία με κατάληξη **NUPKG**). Για τους σκοπούς της εργασίας τα παραπάνω τεμάχια δεδομένων θα αναφέρονται εφεξής ως *module packs*. Τα module packs αξιοποιούνται από το ExamKichen WebApp αρωγή ενός πρόσθετου Reflection Extension Set που περιέχει επεκτάσεις ικανές να ανακτήσουν τις πληροφορίες ενός Module και να ανασύρουν τα δομοστοιχεία ερώτησης και εξέτασης εντός ενός assembly. Αναλυτικότερα:

- `bool IsQuestionModule(this Type type)`

  Η μέθοδος *IsQuestionModule* διευκρινίζει εάν ο τύπος αντικειμένου από τον οποίο κλήθηκε αντιπροσωπεύει μια μη αφηρημένη κλάση που κληρονομεί -άμεσα ή έμμεσα- τη λειτουργικότητα των δομοστοιχείων Ερώτησης.

- `QuestionInfo? GetQuestionInfo(this Type type)`

  Η μέθοδος *GetQuestionInfo* επιστρέφει μία πληροφοριακή δομή που περιέχει στοιχεία σχετικά τον τύπο Question Module από τον οποίο κλήθηκε. Σε περίπτωση που ο τύπος δεν ικανοποιεί τις συνθήκες που περιγράφονται στην μέθοδο *IsQuestionModule*, η *GetQuestionInfo* επιστρέφει `null`.

- `QuestionInfo[] GetQuestionModules(this Assembly asm)`

  Η μέθοδος *GetQuestionModules* απαριθμεί όλους τους τύπους που περιέχονται εντός του assembly υπαγωγής της και επιστρέφει μία λίστα με πληροφορίες για τα περιεχόμενα Question Modules. Όταν το assembly δεν περιέχει Question Modules, η μέθοδος επιστρέφει μία κενή λίστα.

- `bool IsExamModule(this Type type)`

  Η μέθοδος *IsExamModule* διευκρινίζει εάν ο τύπος αντικειμένου από τον οποίο κλήθηκε αντιπροσωπεύει μια μη αφηρημένη κλάση, που κληρονομεί -άμεσα ή έμμεσα- τη λειτουργικότητα των δομοστοιχείων Εξέτασης.

- `ExamInfo? GetExamInfo(this Type type)`

  Η μέθοδος *GetExamInfo* επιστρέφει μία πληροφοριακή δομή που περιέχει πληροφορίες σχετικά τον τύπο Exam Module υπαγωγής της. Σε περίπτωση που ο τύπος δεν ικανοποιεί τις συνθήκες που περιγράφονται στην μέθοδο *IsExamModule*, η *GetExamInfo* επιστρέφει `null`.

- `ExamInfo[] GetExamModules(this Assembly asm)`

  Η μέθοδος *GetExamModules* απαριθμεί όλους τους τύπους που περιέχονται εντός του assembly από το οποίο κλήθηκε και επιστρέφει μία λίστα με πληροφορίες έκαστα τα Exam Modules που περιέχει. Όταν το assembly υπολείπεται Exam Modules, η μέθοδος επιστρέφει μία κενή λίστα.

1.4.1.Question Modules

Τα Question Modules (ή δομοστοιχεία ερωτήσεων) αποτελούν δομές δεδομένων που δημιουργούν διαδικαστικά (procedurally) ερωτήσεις αξιολόγησης και, αν το επιτρέπει το είδος των θεμάτων, τις απαντήσεις που τους αντιστοιχούν. Συναρτήσει της ετερογένειας του είδους των ερωτήσεών τους, τα Question Modules χωρίζονται συμβατικά σε τρεις κατηγορίες: τα στατικά Modules, τα ντετερμινιστικά Modules και Modules τυχαιότητας.

- **Στατικά Modules**

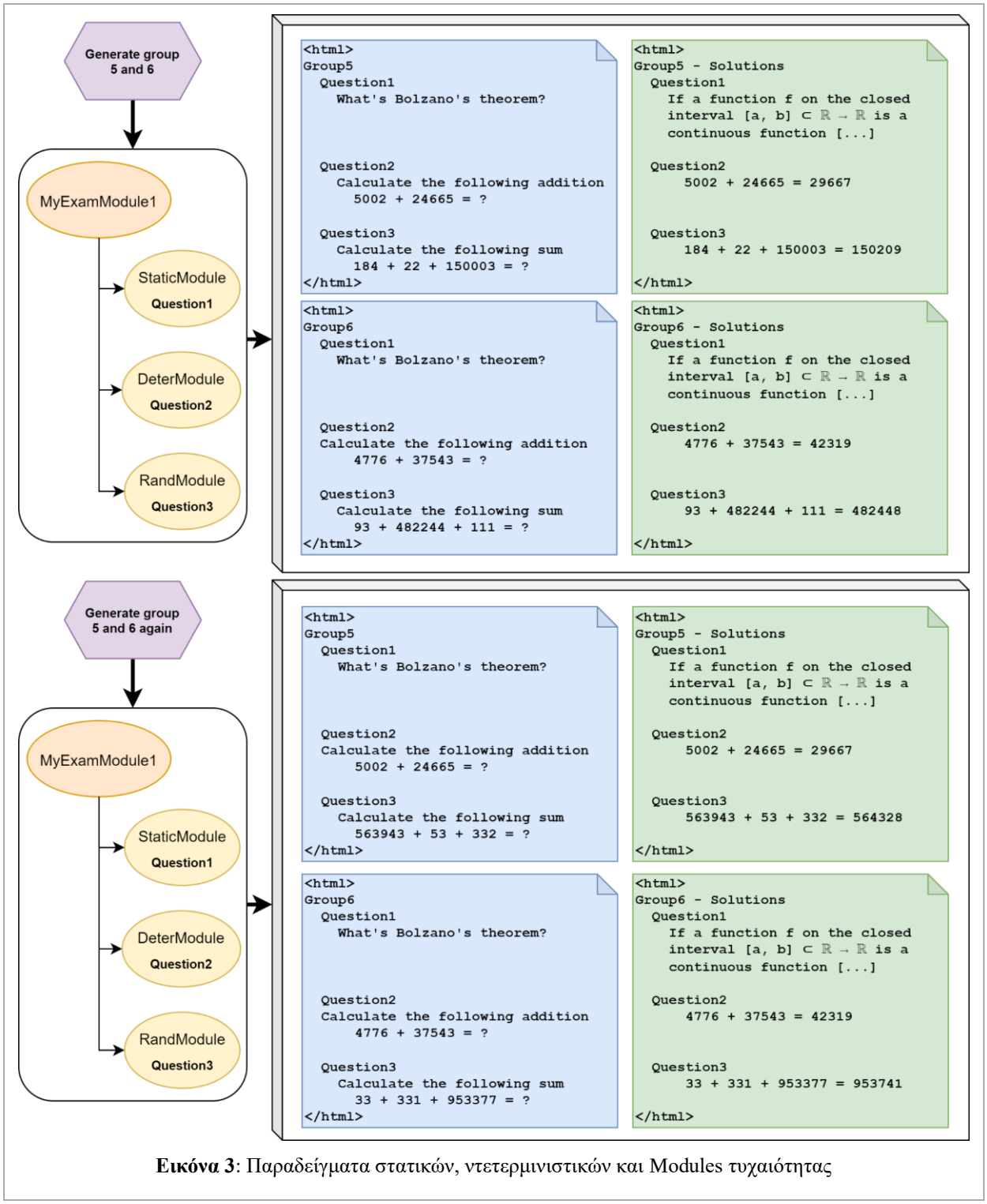
  Οι ερωταπαντήσεις που παράγουν τα στατικά Modules δεν παρουσιάζουν καμία ετερογένεια μεταξύ τους ή, όταν παρουσιάζουν, αυτή περιορίζεται σε επιρροές από τις τιμές των παραμέτρων που συμπληρώνει ο χρήστης. Ως εκ τούτου, εάν οι παράμετροι παραμείνουν αμετάβλητες, ένα στατικό Module θα παράγει την ίδια ερώτηση ξανά και ξανά, ανεξαρτήτως του project που την περιλαμβάνει ή του group που θα κληθεί να επιλύσει την εκφώνηση.

- **Ντετερμινιστικά Modules**

  Τα ντετερμινιστικά Modules περιέχουν ερωτήσεις μερικής ετερογένειας. Οι ερωταπαντήσεις που παράγονται δύνανται να επηρεάζονται από στατικούς παράγοντες, ενώ πάντα μεταβάλλονται συναρτήσει του group στο οποίο απευθύνονται. Οι παραχθείσες ερωταπαντήσεις διαφέρουν από group σε group, αλλά, ελλείψει μεταβολών των παραμέτρων, η ερωταπάντηση που δημιουργείται για ένα group θα είναι πάντα η ίδια, ανεξαρτήτως του project στο οποίο συμπεριλαμβάνεται.

- **Modules Τυχαιότητας**

  Οι ερωταπαντήσεις που παράγουν τα Modules τυχαιότητας, πέρα από μεταβολές στατικών και ντετερμινιστικών παραγόντων, επηρεάζονται από τυχαίους παράγοντες στους οποίους οφείλουν το όνομά τους. Ως αποτέλεσμα κάθε φορά δημιουργούνται μοναδικές ερωταπαντήσεις, ακόμα και όταν αυτές αφορούν τα ίδια groups του ίδιου project που προκύπτουν από διαφορετικές «γέννες».



Εικόνα 3: Παραδείγματα στατικών, ντετερμινιστικών και Modules τυχαιότητας

Δομικά, τα Question Modules διατηρούν τις βασικές τους πληροφορίες εντός ενός *Χαρακτηριστικού Μεταδεδομένων Ερώτησης* (**Question Metadata Attribute**· σημειώνεται ως **[QuestionMetadata]**), που αποβλέπει στην ευκολότερη -από τον προγραμματιστή- ανάκτηση του και την φιλικότερη - για τον ανειδίκευτο χρήστη - αξιοποίηση. Το εν λόγω attribute διατηρεί πάντοτε το **φιλικό όνομά** δομοστοιχείου (π.χ. *«Negative Delta Trinomial Question»*), μία σύντομη **περιγραφή** του σε σύνταξη *MarkDown* και το **όνομα του δημιουργού** του.

Σε αξιολογικούς ελέγχους, η διατήρηση πληθώρας υποερωτημάτων ενός θέματος είναι συνηθισμένη πρακτική. Προκειμένου να μιμηθεί αυτή τη συμπεριφορά, ένα Module ερώτησης μπορεί να συμπεριλαμβάνει ένα ή περισσότερα Question Modules των οποίων οι ερωταπαντήσεις οργανώνονται ως υποερωτήματα της βασικής εκφώνησης. Για την ενεργοποίηση αυτής της δυνατότητας προσφέρονται δύο προαιρετικά ορίσματα τα οποία ο προγραμματιστής μπορεί να συμπεριλάβει κατά την συμπλήρωση του Χαρακτηριστικού Μεταδεδομένων:

Επιλογή 1: **Is Intermediate? (`bool`)**

> Όταν το πεδίο *Is Intermediate?* συμπληρωθεί με την τιμή **`true`**, το Question Module επιτρέπει την διατήρηση SubModules κάθε τύπου. Η προεπιλεγμένη τιμή του πεδίου είναι **`false`**, υποδηλώνοντας πως το Module δεν μπορεί να διατηρήσει υποερωτήματα.

Επιλογή 2: **Valid Sub-Question Types (`Type[]`)**

> Στην περίπτωση που δηλωθεί το πεδίο *Valid Sub-Question Types*, το Question Module θα επιτρέπει την διατήρηση SubModules των οποίων ο τύπος συμπεριλαμβάνεται στην δοθείσα λίστα αποδεκτών τύπων δομοστοιχείων ερώτησης.

Η πρόσβαση στα υποερωτήματα των Question Modules αποκτάται μέσω της *ιδιότητας* (**property**) `Questions`, ενώ η εισαγωγή ενός καινούριου Question Module στα υποερωτήματα επιτυγχάνεται μέσω της μεθόδου **`void AddSubQuestion (QuestionModule module)`**. Η *AddSubQuestion*, αφού ελέγξει την ορθότητα της εισαγωγής -κατά πόσο, δηλαδή, το δομοστοιχείο δέχεται υποερωτήματα και εάν το δοθέν υποερώτημα είναι τύπου που υποστηρίζεται- την εκτελεί.

Η διαδικασία παραγωγής ενός νέου ζεύγους ερώτησης/απάντησης γίνεται μέσω της μεθόδου **`void Prepare (QuestionModuleResult result, int groupId)`**, μία αφηρημένη μέθοδο την οποία πρέπει να υλοποιήσει ο προγραμματιστής κάθε Module. Σε κάθε κλήση της, η μέθοδος *Prepare* συνθέτει μία ερώτηση, υπολογίζει την απάντησή της και τοποθετεί τις πληροφορίες του νεοσύστατου ζεύγους στο όρισμα `result`. Το νεοσύστατο ζεύγος χωρίζεται σε τμήματα με σκοπό την διευκόλυνση της μορφοποίησης τους κατά την δημιουργία των ομάδων. Τα τμήματα είναι συνολικά επτά και εμφωλεύονται εντός μίας δομής δεδομένων ονόματι **QuestionModuleResult**. Αναλυτικότερα:

- **`string`** `Title`

  Το πεδίο *Title* διατηρεί τον τίτλο της ερώτησης (π.χ. Θέμα 1) και συμπληρώνεται από τον προγραμματιστή αρωγή της μεθόδου **`void SetTitle (string text)`**. Όταν το πεδίο είναι κενό (**`null`**) ή περιέχει μια άδεια συμβολοσειρά, τότε θεωρείται πως η ερωταπάντηση *δεν έχει εκφώνηση*. Υπενθυμίζεται ότι η ύπαρξη τίτλου φανερώνεται χάρης την Ιδιότητα `HasTitle`.

- **`double?`** `Score`

  Το πεδίο *Score* διατηρεί την μέγιστη βαθμολογία που μπορεί να ο εξεταζόμενος απαντώντας σωστά στην ερώτηση. Η συμπλήρωση του πεδίου γίνεται με την μέθοδο **`void SetScore (double score)`**, ενώ μπορεί να αναιρεθεί μέσω της μεθόδου **`void UnsetScore ()`**. Όταν το πεδίο είναι κενό (**`null`**) τότε θεωρείται πως η ερωταπάντηση *δεν έχει βαθμό* και η τιμή της αντίστοιχης ιδιότητας `HasScore` παραμένει **`false`**.

- **`string`** `QuestionPreText`

  Το πεδίο *QuestionPreText* έχει δύο χρήσεις: Εάν το Module περιλαμβάνει υποερωτήματα, αντιπροσωπεύει το κείμενο που τοποθετείται πριν από αυτά. Όταν πρόκειται για μία *Τερματική ερώτηση* (δηλ. μία ερώτηση χωρίς υποερωτήματα), διατηρεί το κείμενο της ερώτησης εξ ολοκλήρου. Όταν το πεδίο είναι κενό (**`null`**) ή περιέχει μια άδεια συμβολοσειρά, τότε θεωρείται πως η ερωταπάντηση *δεν έχει Αρχικό κείμενο ερώτησης* - χαρακτηριστικό που φανερώνεται μέσω της Ιδιότητας `HasQuestionPreText`.

- **string** AnswerPreText

  Το πεδίο *AnswerPreText* λειτουργεί πανομοιότυπα με το *QuestionPreText*: Εφόσον το Module περιλαμβάνει υποερωτήματα, το πεδίο αντιπροσωπεύει το κείμενο που τοποθετείται πριν από αυτά. Ειδάλλως, διατηρεί συνολικά το κείμενο της απάντησης. Όταν το πεδίο είναι κενό (**null**) ή περιέχει μια άδεια συμβολοσειρά, τότε θεωρείται πως η ερωταπάντηση *δεν έχει αρχικό κείμενο*. Η ύπαρξη αρχικού κειμένου απάντησης διαγιγνώσκεται μέσω της Ιδιότητας HasAnswerPreText.

- **List<QuestionModuleResult>** SubQuestions

  Όταν το Module διατηρεί υποερωτήματα, το πεδίο *SubQuestions* διατηρεί τα παραχθέντα ζεύγη ερωταπαντήσεων των υποερωτημάτων. Εάν η ερώτηση είναι τερματική, η λίστα παραμένει κενή.

- **string** QuestionPostText

  Συμπληρώνοντας την λειτουργικότητα του *QuestionPreText*, το πεδίο *QuestionPostText* χρησιμοποιείται σε περίπτωση που το Module διατηρεί υποερωτήματα και αντιπροσωπεύει το κείμενο που τοποθετείται μετά από αυτά. Όταν το πεδίο είναι κενό (**null**) ή περιέχει μια άδεια συμβολοσειρά, τότε θεωρείται πως η ερωταπάντηση *δεν έχει τελικό κείμενο ερώτησης* - χαρακτηριστικό που φανερώνεται μέσω της Ιδιότητας HasQuestionPostText.

- **string** AnswerPostText

  Κατ' αντιστοιχία με το *QuestionPostText*, το πεδίο *AnswerPostText* διατηρεί το κείμενο που τοποθετείται κάτω από τις απαντήσεις των υποερωτημάτων του Module, όταν αυτές υπάρχουν. Όταν το πεδίο είναι κενό (**null**) ή περιέχει μια άδεια συμβολοσειρά, τότε θεωρείται πως η ερωταπάντηση *δεν έχει Τελικό κείμενο απάντησης*. Στην αντίθετη περίπτωση, η ύπαρξη τελικού κειμένου απάντησης καθορίζεται από την ιδιότητα HasAnswerPostText.

Η ανάθεση τιμών στα πεδία *QuestionPreText, QuestionPostText, AnswerPreText* και *AnswerPostText* επαφίεται στις μεθόδους **void SetQuestion (string markdownPreText, string markdownPostText)** και **void SetAnswer (string markdownPreText, string**

`markdownPostText)` οι οποίες -όπως προμηνύουν και τα ονόματα των ορισμάτων τους- αναγνωρίζουν κείμενο σύνταξης MarkDown που αντιστοιχεί στον κορμό των ερωταπαντήσεων. Κατά την εκτέλεση των εν λόγω μεθόδων, τα ορίσματα μεταφράζονται από MarkDown σε HTML και αποθηκεύονται στα κατάλληλα πεδία.

1.4.2. Exam Modules

Τα **Exam Modules** συνιστούν τη βάση κάθε project και, αρωγή τουλάχιστον ενός Question Module, παράγουν ομάδες θεμάτων αξιολόγησης. Κατ' αναλογία με τα δομοστοιχεία ερώτησης, ο προγραμματιστής παρέχει πληροφορίες για το Module μέσω ενός *Χαρακτηριστικού Μεταδεδομένων Εξέτασης* (**Exam Metadata Attribute**· σημειώνεται ως **[ExamMetadata]**) που διατηρεί (1) το φιλικό όνομά του δομοστοιχείου (π.χ. *«APA-Styled exam template»*), (2) μία σύντομη περιγραφή του σε σύνταξη *MarkDown* και (3) το όνομα του δημιουργού του.

Για την περαιτέρω διαμόρφωση τους από τον προγραμματιστή, τα Exam Modules περιλαμβάνουν μία αφηρημένη μέθοδο ονόματι `void ConfigureExam (ExamConfiguration config)`. Η *ConfigureExam* καλείται κατά τη δημιουργία ενός αντικειμένου τύπου Exam Module. Το όρισμα `config` της μεθόδου είναι μία κλάση βασισμένη στο design pattern του **Configurator** και προσφέρει τις ακόλουθες μεθόδους διαμόρφωσης:

- ```
  void SetPipeline (
          Action<MarkdownPipelineBuilder> pipelineBuildFunction)
  ```

  Τα περισσότερα πεδία του ExamKitchen επιτρέπουν την συμπλήρωσή τους σε σύνταξη MarkDown. Ως εκ τούτου, κάθε Exam Module διαθέτει μία εξατομικευμένη διαδικασία μετατροπής του MarkDown κειμένου σε HTML, την οποία κληροδοτεί στα Question Modules που περιλαμβάνει κατά τη χρήση. Η διαμόρφωση του μετατροπέα γίνεται μέσω της μεθόδου *SetPipeline*.
  ```

- `void SetCssFile (string name, Func<string> fileContents)` και
  `void SetCssFile (string name, string fileContents)`

  Η μέθοδος *SetCssFile* επιτρέπει στον προγραμματιστή να ενσωματώσει *Κλιμακωτά Φύλλα Στυλ* (**Cascading Style Sheets** ή **CSS**) σε κάθε ομάδα. Σε κάθε επίκληση της η *SetCssFile* δημιουργεί ή επεξεργάζεται μία καταχώρηση στην λίστα `StyleSheets` του Configurator. Η εν λόγω λίστα διατηρεί το όνομα και το περιεχόμενο του αρχείου. Ο προγραμματιστής δεν χρειάζεται να προβεί σε περαιτέρω ενέργειες στο αρχείο, καθώς η δημιουργία και η ενσωμάτωσή του στα φύλλα ερωτήσεων και απαντήσεων γίνεται αυτόματα από το ExamKitchen WebApp.

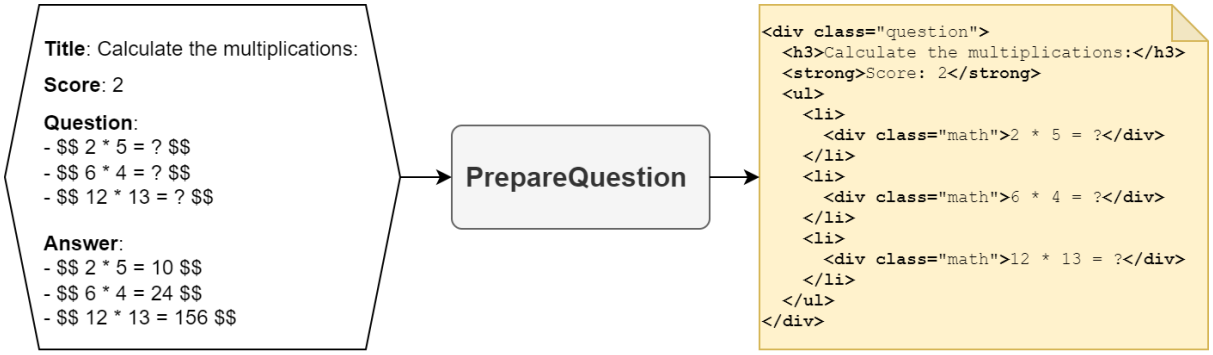
- `void SetJsFile (string name, Func<string> fileContents)` και
  `void SetJsFile (string name, string fileContents)`

  Η μέθοδος *SetJsFile* επιτρέπει στον προγραμματιστή να ενσωματώσει αρχεία JavaScript (JS) στις παραγόμενες ομάδες. Κάθε επίκληση της *SetJsFile* δημιουργεί ή επεξεργάζεται μία καταχώρηση της λίστα `JavaScripts` που βρίσκεται στον Configurator. Κάθε καταχώρηση στη λίστα διατηρεί το όνομα και το περιεχόμενο ενός αρχείου JS. Όπως στη μέθοδο *SetCssFile*, η δημιουργία και η ενσωμάτωσή των αρχείων JavaScript στα φύλλα ερωτήσεων και απαντήσεων γίνεται αυτόματα από το ExamKitchen WebApp.

Για τη γένεση κάθε group θεμάτων οι αρμοδιότητες ενός Exam Module περιλαμβάνουν την διαμόρφωση της τελικής σελίδας ερωτήσεων, τη διαμόρφωση της τελικής σελίδας των απαντήσεων, τη γένεση μεμονωμένων ζευγών ερώτησης/απάντησης από τα Question Modules που περιλαμβάνει, την ενσωμάτωση των δημιουργηθέντων ζευγών στις σελίδες ερωτήσεων και απαντήσεων, και την εξαγωγή των νεόδμητων σελίδων του group ως ζευγάρι Για τους παραπάνω σκοπούς, η δομή του δομοστοιχείου εξέτασης συμπεριλαμβάνει τρεις αφηρημένες μεθόδους.

- **`string`** `PrepareQuestion` **`(QuestionModuleResult`** `question`**`)`**

  Η μέθοδος *PrepareQuestion* είναι υπεύθυνη για την δόμηση και επιστροφή του HTML τμήματος ερώτησης ενός δημιουργηθέντος από Question Module ζεύγους ερώτησης/απάντησης.

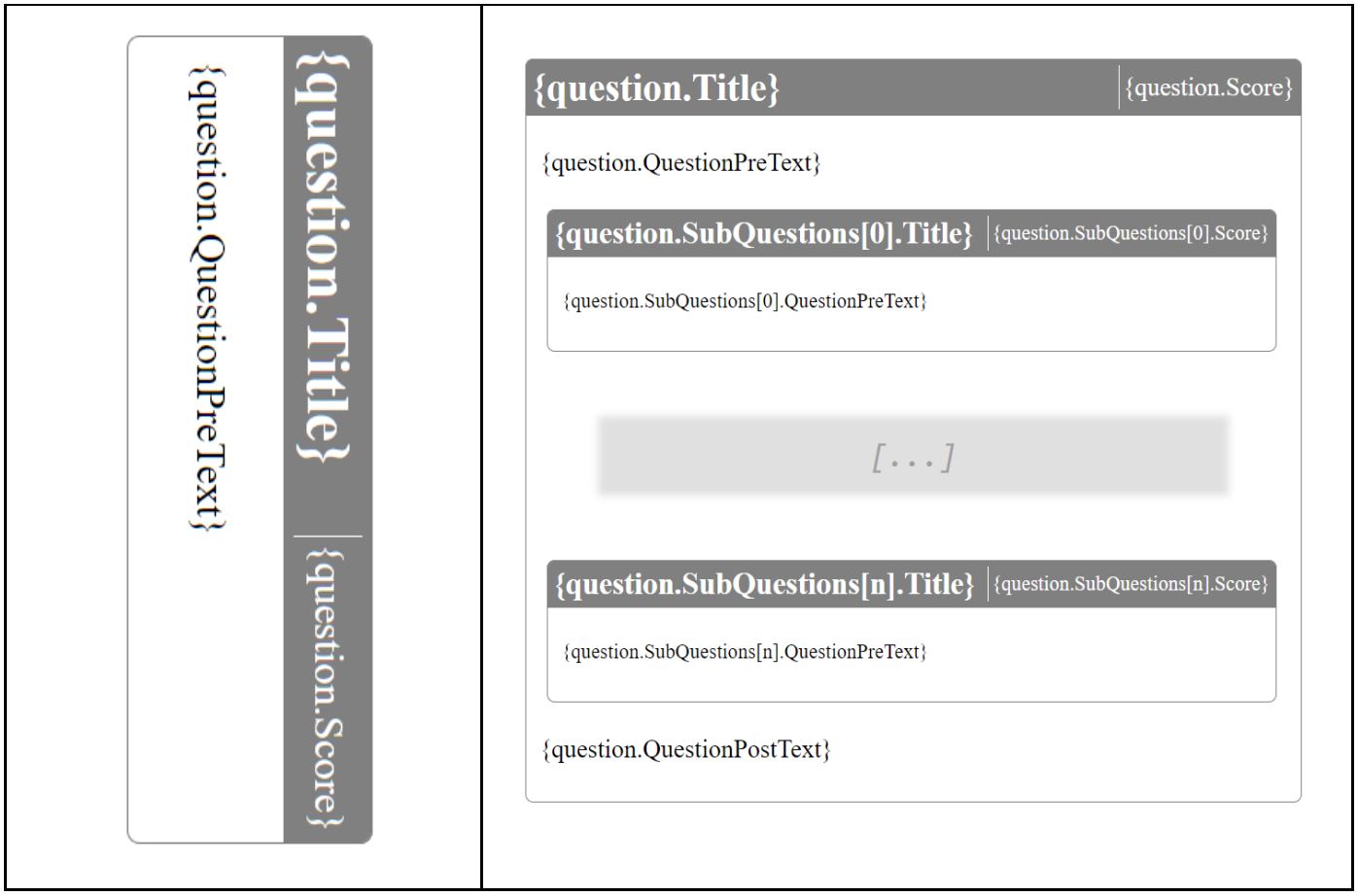


Εικόνα 4: Σχεδιαγραμματική απεικόνιση της λειτουργικότητας της μεθόδου PrepareQuestion

Αν η ερώτηση περιλαμβάνει υποερωτήματα, αυτά διατηρούνται εντός του μέλους `SubQuestions` του γνωρίσματος `question`. Τότε, για κάθε αντικείμενο του `question.SubQuestions` η μέθοδος καλείται αναδρομικά και τα νεοσύστατα υποερωτήματα ενσωματώνονται στο σώμα της αρχικής ερώτησης.

Χωρίς υποερωτήματα	Με υποερωτήματα	

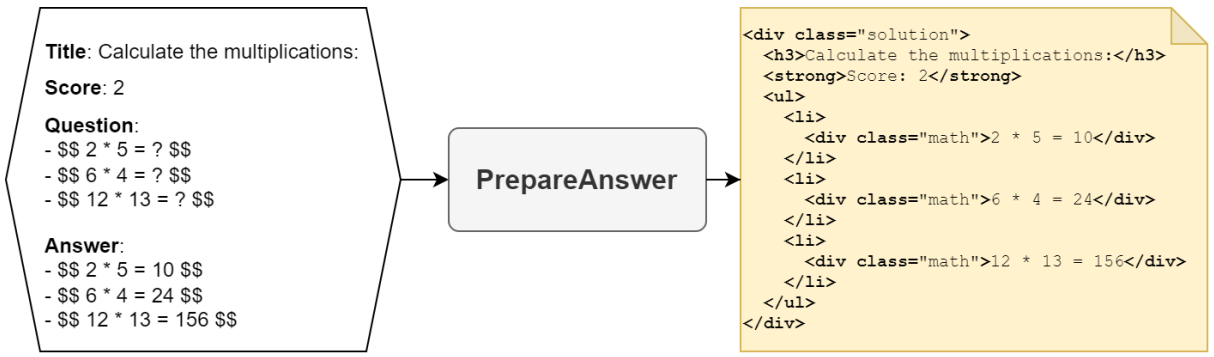
`<div class="question">`
`  <div class="q-header">`
`  <span class="q-title">`
`  {question.Title}`
`  </span>`
`  <span class= "q-score">`
`  {question.Score}`
`  </span>`
`  </div>`
 	
`  <div class="q-body">`
`  {question.QuestionPreText}`
`  </div>`
 	
`</div>`
	`<div class="question">`
	`  <div class="q-header">`
	`  <span class="q-title">`
	`  {question.Title}`
	`  </span>`
	`  <span class= "q-score">`
	`  {question.Score}`
	`  </span>`
	`  </div>`
	
	`  <div class="q-body">`
	`  {question.QuestionPreText}`
	
	`  <div class="question">`
	`  <!--`
	`  Question body of`
	`  question.SubQuestions[0]`
	`  -->`
	`  </div>`
		`<!----- [...] ------->`
		
		`  <div class="question">`
		`  <!--`
		`  Question body of`
		`  question.SubQuestions[n]`
		`  -->`
		`  </div>`
		
		`  {question.QuestionPostText}`
		`  </div>`
		`</div>`



Πίνακας 1 : Παραδείγματα αποτελεσμάτων της μεθόδου PrepareQuestion για ερωτήσεις με και χωρίς υποερωτήματα

- `string PrepareAnswer (QuestionModuleResult question)`

  Κατ' αντιστοιχία με την *PrepareQuestion*, η μέθοδος *PrepareAnswer* διαμορφώνει και επιστρέφει το HTML τμήμα απάντησης ενός ζεύγους ερώτησης/απάντησης.



Εικόνα 5: Σχεδιαγραμματική απεικόνιση της λειτουργικότητας της μεθόδου PrepareAnswer

Παρουσία υποερωτημάτων για κάθε αντικείμενο του `question.SubQuestions`, η μέθοδος καλείται αναδρομικά και τα παραχθέντα HTML τμήματα απάντησης ενσωματώνονται στο σώμα της αρχικής απάντησης.

<table>
<tr><th>Χωρίς υποερωτήματα</th><th colspan="2">Με υποερωτήματα</th></tr>
<tr>
<td>

```
<div class="question">
  <div class="q-header">
    <span class="q-title">
      {question.Title}
    </span>
    <span class= "q-score">
      {question.Score}
    </span>
  </div>

  <div class="q-body">
    {question.AnswerPreText}
  </div>

</div>
```

</td>
<td>

```
<div class="question">
  <div class="q-header">
    <span class="q-title">
      {question.Title}
    </span>
    <span class= "q-score">
      {question.Score}
    </span>
  </div>

  <div class="q-body">
    {question.AnswerPreText}

    <div class="question">
      <!--
        Answer body of
        question.SubQuestions[0]
      -->
    </div>
```

</td>
<td>

```
    <!----- [...] ------->

    <div class="question">
      <!--
        Answer body of
        question.SubQuestions[n]
      -->
    </div>

    {question.AnswerPostText}
  </div>
</div>
```

</td>
</tr>
<tr>
<td>

{question.Title} {question.Score}

{question.AnswerPreText}

</td>
<td colspan="2">

{question.Title} {question.Score}

{question.AnswerPreText}

{question.SubQuestions[0].Title} {question.SubQuestions[0].Score}

{question.SubQuestions[0].AnswerPreText}

[...]

{question.SubQuestions[n].Title} {question.SubQuestions[n].Score}

{question.SubQuestions[n].AnswerPreText}

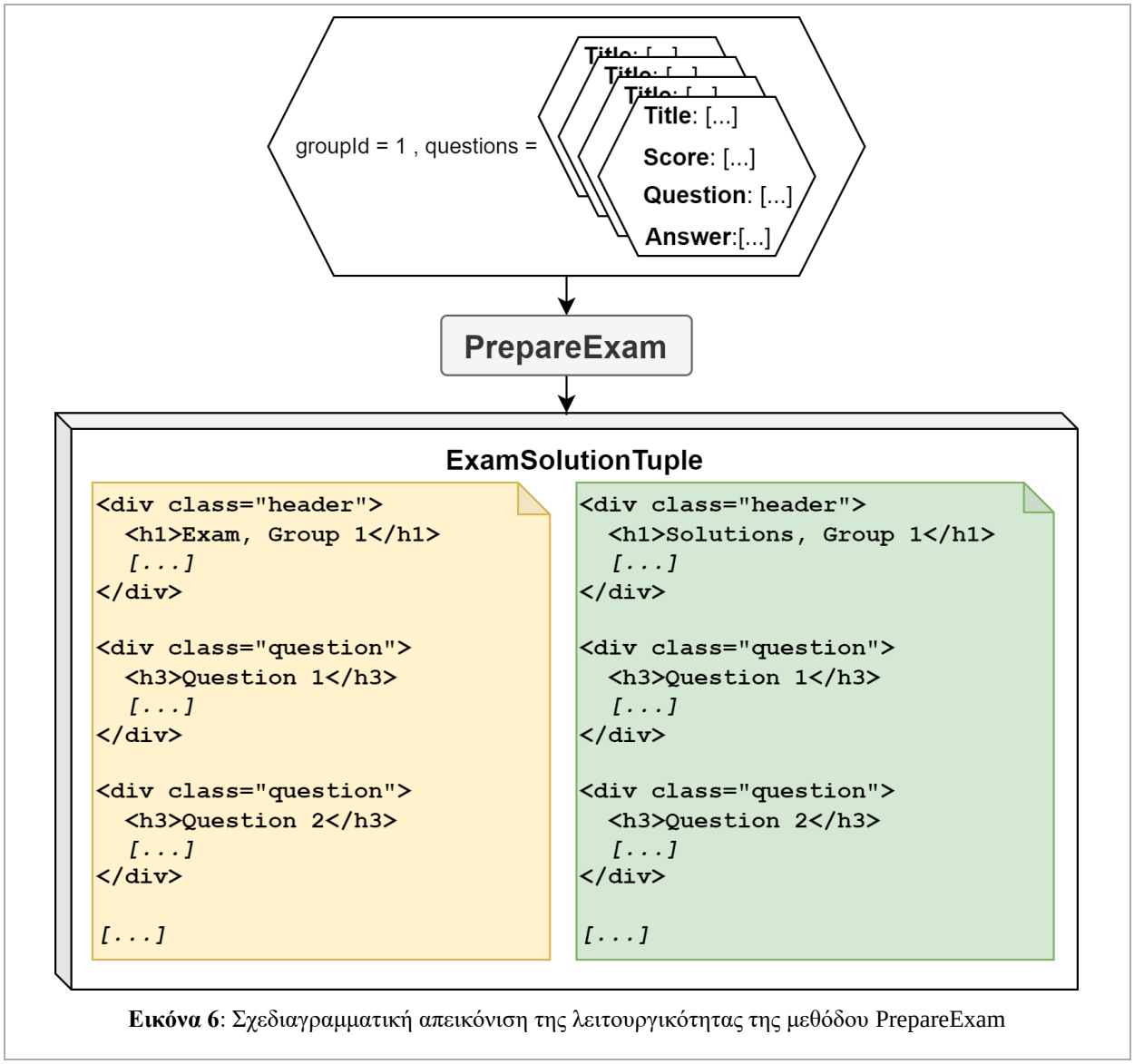
{question.AnswerPostText}

</td>
</tr>
</table>

Πίνακας 2 : Παραδείγματα αποτελεσμάτων της μεθόδου PrepareAnswer για ερωτήσεις με και χωρίς υποερωτήματα

- **`ExamSolutionTuple`** `PrepareExam` **`(int`** `groupId`**`,`**
  `QuestionModuleResult[]` `questions`**`)`**

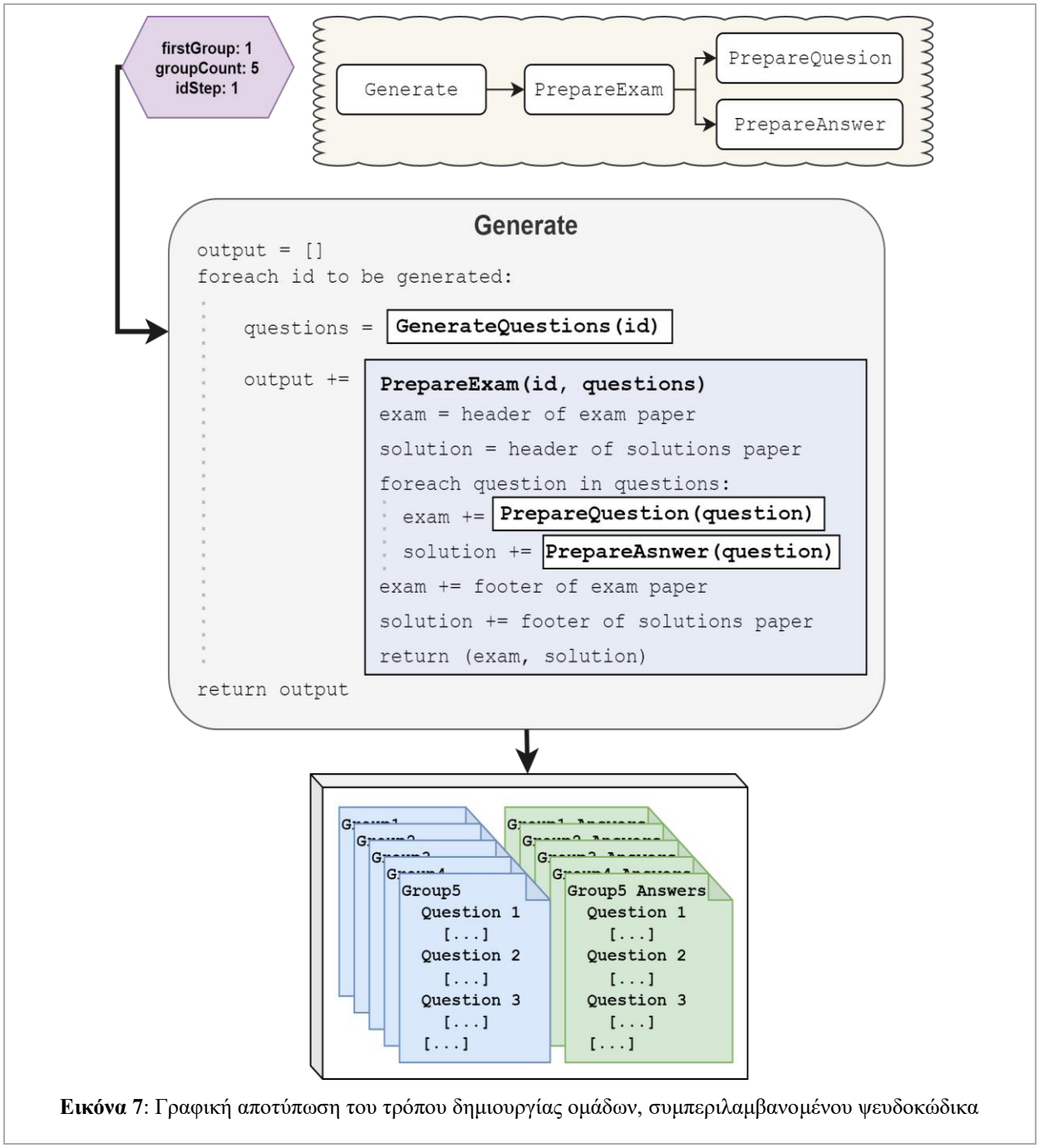
  Αντλώντας από τις τελευταίες δύο μεθόδους, η μέθοδος *PrepareExam* δημιουργεί και επιστρέφει το σώμα (δηλ. οτιδήποτε περιέχεται εντός του HTML στοιχείου `<body></body>`) των εγγράφων εξέτασης και λύσεων ενός group.



Εικόνα 6: Σχεδιαγραμματική απεικόνιση της λειτουργικότητας της μεθόδου PrepareExam

Οι παραπάνω μέθοδοι δεν είναι διαθέσιμες δημόσια, αλλά συντελούν τις επιμέρους λειτουργίες ενός *επίπεδου αφαιρετικότητας* (abstraction layer), που επιτρέπει την μαζική

δημιουργία ομάδων με την κλήση μίας και μόνο συνάρτησης: Η **`ExamSolutionTuple[] Generate(int firstGroupId, int groupCount, int idStep)`** λαμβάνει ως ορίσματα τον αριθμό του πρώτου group, τον αριθμό των groups προς γένεση και το βήμα αυξομείωσης του αριθμού group, ανά βήμα παραγωγής. Για παράδειγμα, προκειμένου να γίνει παραγωγή των groups 1-5, τα ορίσματα θα έχουν τιμές `firstGroupId` = 1, `groupCount` = 5, `idStep` = 1. Αντίστοιχα, για να γίνει παραγωγή των groups 9,7,5,3 και 1, τα ορίσματα λαμβάνουν τις τιμές `firstGroupId` = 9, `groupCount` = 5, `idStep` = -2.



Εικόνα 7: Γραφική αποτύπωση του τρόπου δημιουργίας ομάδων, συμπεριλαμβανομένου ψευδοκώδικα

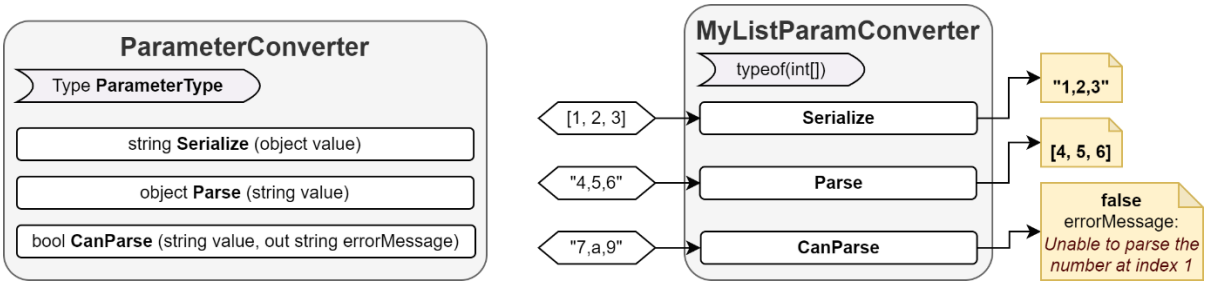
1.5. Σειριοποίηση και αποσειριοποίηση παραμέτρων

Οι δομές δεδομένων που παρέχει το ExamKitchen επιτρέπουν την αποθήκευση και την ανάκτηση της κατάστασης τους -συμπεριλαμβανομένων των τιμών των παραμέτρων τους- μέσω σειριοποίησης/αποσειριοποίησής τους σε/από αρχεία κειμένου σημειογραφίας JSON. Για απλούς τύπους δεδομένων, όπως οι συμβολοσειρές και οι αριθμοί, η μετατροπή είναι ιδιαίτερα εύκολη και δεν προϋποθέτει συγκεκριμένη σύνταξη (βλ. πίνακα 3). Εντούτοις, η μετατροπή πιο σύνθετων δομών, όπως τα αντικείμενα, οι λίστες και άλλες δομές που ενέχονται σε μία αντικειμενοστραφή γλώσσα δεν είναι απαραίτητα τυποποιημένη. Ακόμα και στις περιπτώσεις που υπάρχει κάποια τυποποίηση, όταν παρεμβάλλεται ο ανθρώπινος παράγοντας, το αποτέλεσμα της σειριοποίησης και, συνεπώς, η μορφή της απαιτούμενης εισόδου προς αποσειριοποίηση, είναι δύσκολα αναπαράξιμες και στριφνές ως προς την σύνταξή τους. Όταν οι διαδικασίες μετατροπής είναι προσιτές μόνο σε επίπεδο κώδικα, αυτό δεν αποτελεί πρόβλημα, καθώς ο υπολογιστής δεν αποκλίνει από την ορισμένη από τον προγραμματιστή σύνταξη. Ωστόσο, στην περίπτωση των παραμέτρων ενός Module, η ανάγκη παροχής ενός ευνόητου και αναπαράξιμου τρόπου συγγραφής είναι μείζονος σημασίας για το χρήστη. Διαγραμματικά:

String		Value
"This is a text"	⇌	This is a text
"15.22"	⇌	15.22

Πίνακας 3: Σειριοποίηση και αποσειριοποίηση απλών δομών δεδομένων

Προς υπέρβαση του άνωθεν κωλύματος δημιουργήθηκε μία αφηρημένη κλάση ονόματι *Μετατροπέας Παραμέτρων* (**ParameterConverter**) η οποία (απο)σειριοποιεί παραμέτρους συγκεκριμένου τύπου σε ή από συμβολοσειρές, προσφέροντας προσαρμοσμένη σύνταξη και καθοδηγώντας τον χρήση στη συγγραφή τους αρωγή μηνυμάτων λάθους.



Εικόνα 8 : Παράδειγμα αλληλεπίδρασης με έναν Μετατροπέα Παραμέτρου

Όπως διαφαίνεται στην Εικόνα 8, η υπό συζήτηση κλάση απαιτεί την εισαγωγή του τύπου της παραμέτρου ως όρισμα κατασκευής, παρότι είθισται η απόκρυψη του παρέχοντάς το στατικά κατά την κατασκευή του εκάστοτε προσαρμοσμένου μετατροπέα.

```
// This already exists within ExamKitchen’s Core
public abstract class ParameterConverter
{

    public Type ParameterType { get; }

    protected ParameterConverter(Type parameterType)
    {
        ParameterType = parameterType;
        // [...]
    }

    // [...]
}

// This is how one should approach creating a Parameter Converter
// for objects of type “MyObject”
public class MyObjectParamConverter : ParameterConverter
{

    public MyObjectParamConverter() : base(typeof(MyObject)) // Abstraction Here
    {
        // [...]
    }

    // [...]
}
```

Πίνακας 4 : Παράδειγμα απόκρυψης του γνωρίσματος τύπου της παραμέτρου

Οι μέθοδοι που συναπαρτίζουν έναν Μετατροπέα παραμέτρου είναι οι εξής:

- `object Parse (string value)`

  Βάσει υλοποίησης η μέθοδος *Parse* αποσειριοποεί το περιεχόμενο της συμβολοσειράς `value` και παράγει μία καινούρια συμβολοσειρά που το αναπαριστά. Στην συνέχεια, επιστρέφει την αντίστοιχη συμβολοσειρά.

- `string Serialize (object value)`

  Η μέθοδος *Serialize* σειριοποιεί το αντικείμενο που δίνεται στο όρισμα `value`, δημιουργώντας μία καινούρια συμβολοσειρά που το αναπαριστά. Στην συνέχεια, επιστρέφει την εν λόγω συμβολοσειρά.

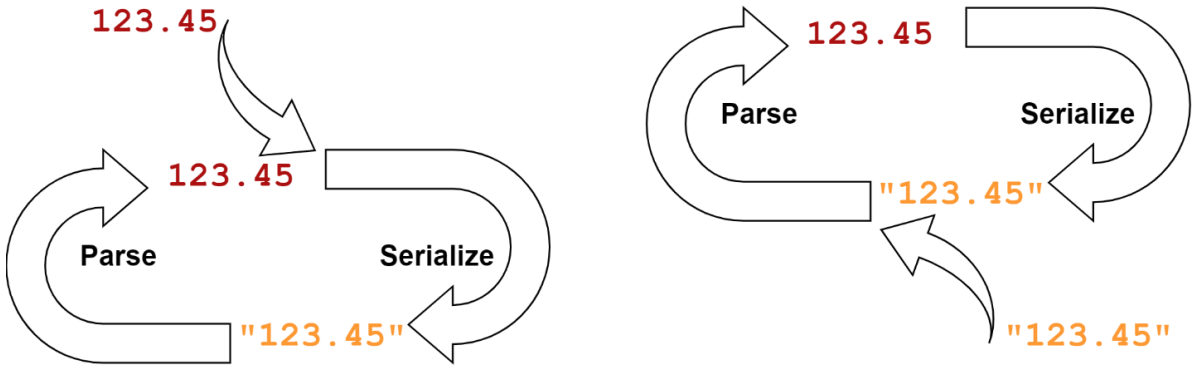
- `bool CanParse (string value, out string errorMessage)`

  Η μέθοδος *CanParse* ελέγχει την συντακτική εγκυρότητα της δοθείσας συμβολοσειράς σύμφωνα με το πρότυπο μετατροπής που υλοποιεί ο μετατροπέας. Σε περίπτωση συντακτικής αρτιότητας, η μέθοδος επιστρέφει την τιμή **`true`**, ενώ το όρισμα `errorMessage` διατηρεί την τιμή **`null`**. Στην αντίθετη περίπτωση, η μέθοδος επιστρέφει την τιμή **`false`**, και συμπληρώνει το όρισμα `errorMessage` με το κατάλληλο μήνυμα λάθους.

Σε κάθε υλοποίηση Μετατροπέα Παραμέτρου, οι παραπάνω μέθοδοι λύνουν υποχρεωτικά κάθε *Εξαίρεση* (**Exception**) που ενδέχεται να προκύψει κατά την εκτέλεσή τους με ασφαλείς προγραμματιστικές πρακτικές και ρήτρες *Try-Catch*. Σε περίπτωση αδυναμίας ολοκλήρωσής τους, επιστρέφουν μία έγκυρη προεπιλεγμένη τιμή. Οι μέθοδοι σειριοποίησης και αποσειριοποίησης είναι πάντοτε **αλληλοσυμβατές**:

Έστω O, ένα αντικείμενο τύπου συμβατού με έναν μετατροπέα C. Κατόπιν σειριοποίησής του με την μέθοδο $C.Serialize$, η παραχθείσα συμβολοσειρά S αποτελεί έγκυρη, μετατρέψιμη είσοδο για την μέθοδο $C.Parse$. Όταν η S μετατραπεί πίσω σε αντικείμενο O' μέσω της μεθόδου $C.Parse$, ισχύει πάντα:

$$O \equiv O'.$$



Εικόνα 9: Γραφική απεικόνιση της αλληλοσυμβατότητας μεταξύ Parse και Serialize

Όσον αφορά την ικανοποίηση βασικών τύπων, το ExamKitchen προσφέρει ένα περιορισμένο σύνολο *Προεπιλεγμένων Μετατροπέων Παραμέτρων*. Όταν δεν δηλωθεί ρητά ο τύπος μετατροπέα μιας παραμέτρου, επιλέγεται αυτόματα ο καταλληλότερος από τους ακόλουθους μετατροπείς:

- **`IntParameterConverter`**

  Ο μετατροπέας *IntParameterConverter* μετατρέπει ακέραιους αριθμούς σε συμβολοσειρά και το αντίστροφο. Εφόσον ο μετατροπέας δεν έχει δηλωθεί ρητά, ανατίθεται αυτόματα σε παραμέτρους τύπου **`int`**.

String		Value
"15"	⇌	15
"0"	⇐	null
"" / null / fail	⇒	0

Πίνακας 5: Πίνακας μετατροπής IntParameterConverter

- **`FloatParameterConverter`**

  Ο μετατροπέας *FloatParameterConverter* ειδικεύεται στην μετατροπή δεκαδικών αριθμών μονής ακριβείας σε συμβολοσειρά και τη μετατροπή συμβολοσειρών σε δεκαδικούς αριθμούς μονής ακριβείας. Σε περίπτωση μη ρητής δήλωσης του τύπου του μετατροπέα, ανατίθεται αυτόματα σε παραμέτρους τύπου **`float`**.

String		Value
"0.5"	⇌	0.5f
"0"	⇐	null
"" / null / fail	⇒	0f

Πίνακας 6: Πίνακας μετατροπής FloatParameterConverter

- **`DoubleParameterConverter`**

  Ανάλογα, ο μετατροπέας *DoubleParameterConverter* μετατρέπει δεκαδικούς αριθμούς διπλής ακριβείας σε συμβολοσειρά και αντιστρόφως. Σε περίπτωση μη ρητής δήλωσης τύπου μετατροπέα, ανατίθεται αυτόματα σε παραμέτρους τύπου **`double`**.

String		Value
"230.03012"	⇌	230.03012d
"0"	⇐	null
"" / null / fail	⇒	0d

Πίνακας 7: Πίνακας μετατροπής DoubleParameterConverter

- **`StringParameterConverter`**

  Καθώς οι συμβολοσειρές αποτελούν εξ ορισμού σειριοποιημένες μορφές δεδομένων, ο μετατροπέας *StringParameterConverter* είναι μία εικονική δομή η οποία, - με εξαίρεση περιπτώσεων μηδενικής εισόδου (**`null`**) την οποία μετατρέπει σε συμβολοσειρά μηδενικού μήκους - επιστρέφει αυτούσια την είσοδο που λαμβάνει στην

έξοδό του. Σε περίπτωση μη ρητής δήλωσης τύπου μετατροπέα, ανατίθεται αυτόματα σε παραμέτρους τύπου **string**.

String		Value
"Lorem Ipsum"	⇌	"Lorem Ipsum"
""	⇐	null
null	⇒	""

Πίνακας 8: Πίνακας μετατροπής StringParameterConverter

- **BoolParameterConverter**

Ο μετατροπέας *BoolParameterConverter* μετατρέπει παραμέτρους δυαδικής (Boolean) άλγεβρας σε συμβολοσειρές και συμβολοσειρές σε παραμέτρους δυαδικής άλγεβρας. Εάν ο μετατροπέας δεν έχει δηλωθεί, ανατίθεται αυτόματα σε παραμέτρους τύπου **bool**.

String		Value
"TrUe" / "fALSE"	⇌	true / false
"1" / "0"	⇒	true / false
"false"	⇐	null
"" / null / fail	⇒	false

Πίνακας 9: Πίνακας μετατροπής BoolParameterConverter

- **EnumParameterConverter**

Ο μετατροπέας *EnumParameterConverter* ειδικεύεται στην μετατροπή παραμέτρων οποιουδήποτε τύπου *δομής απαρίθμησης* (**enum**) σε συμβολοσειρά και αντιστρόφως. Σε περίπτωση μη ρητής δήλωσης τύπου μετατροπέα, ανατίθεται αυτόματα σε παραμέτρους τύπου *δομής* (**struct**) που κληρονομούν την λειτουργικότητα της κλάσης **Enum**. Για κάθε δομή απαρίθμησης της μορφής:

```
enum MyEnum
{
    Aa = 1,
    Bb = 2,
    Cc = 4
}
```

Ο πίνακας μετατροπής έχει την μορφή:

String		Value
"Aa" / "Bb" / "Cc"	⇌	MyEnum.Aa / MyEnum.Bb / MyEnum.Cc
"1" / "2" / "4"	⇒	MyEnum.Aa / MyEnum.Bb / MyEnum.Cc
"Aa"	⇐	null
"" / null / fail	⇒	MyEnum.Aa

Πίνακας 10: Πίνακας μετατροπής EnumParameterConverter

- **`JsonParameterConverter`**

Ο μετατροπέας *JsonParameterConverter* αποτελεί μία απόπειρα μετατροπής δομών δεδομένων που δεν πληρούν τα άνωθεν κριτήρια, όταν δεν έχει ανατεθεί ρητά κάποιος τύπος μετατροπέα γι' αυτές. Χρησιμοποιώντας την βιβλιοθήκη **Newtonsoft.Json** (www.nuget.org/packages/Newtonsoft.Json), ο συγκεκριμένος μετατροπέας επιδιώκει τη μετατροπή αντικειμένων κάθε τύπου σε συμβολοσειρές με σημειογραφία JSON και αντιστρόφως. Για κάθε αντικείμενο με δομή:

```
class MyClass {
    public int Aa;
    public string Bb;
}
```

Ο πίνακας μετατροπής έχει την μορφή:

String		Value

"{
'Aa': 1,
'Bb': 'Test'
}"
	⇌	new MyClass(){
		Aa = 1,
		Bb = "Test"
		}

""	⇐	null / fail
"" / null / fail	⇒	default(MyClass)

Πίνακας 11: Πίνακας μετατροπής JsonParameterConverter

Κεφάλαιο 2: WebApp

2.1. Εισαγωγή

Η in vivo αξιοποίηση των διαθέσιμων δομοστοιχείων πραγματοποιείται μέσω μιας διαδικτυακής εφαρμογής. Το WebApp του ExamKitchen επιτρέπει στους χρήστες - σε εξεταστικούς, δηλαδή, φορείς - να δημιουργήσουν εξατομικευμένα projects και φύλλα αξιολόγησης βασισμένα σε υπάρχουσες βιβλιοθήκες. Φύσει μινιμαλιστική, η ιστοσελίδα αποτελείται από τις ακόλουθες σελίδες:

- **Index (Αρχική σελίδα)**

  Η αρχική σελίδα δεν διατηρεί κάποια πληροφορία. Αποτελεί εξ ολοκλήρου το *σημείο εισόδου* (**entry point**) της εφαρμογής.

- **/Assemblies**

  Η σελίδα */Assemblies* επιτρέπει στο χρήστη να προσθέσει Modules εμφωλευμένα σε Δυναμικές βιβλιοθήκες (αρχεία **.dll**) ή Πακέτα NuGet (αρχεία **.nupkg**) ή να αφαιρέσει, όταν το κρίνει απαραίτητο ήδη εγκατεστημένα Modules.

- **/Projects**

  Η σελίδα */Projects* είναι ο καμβάς του χρήστη - το κέντρο ελέγχου των projects του. Επιτρέπει την δημιουργία, τη μετονομασία, τη διαγραφή και την επεξεργασία projects. Η επεξεργασία των project τον ανακατευθύνει σε διαφορετικό τμήμα της εφαρμογής.

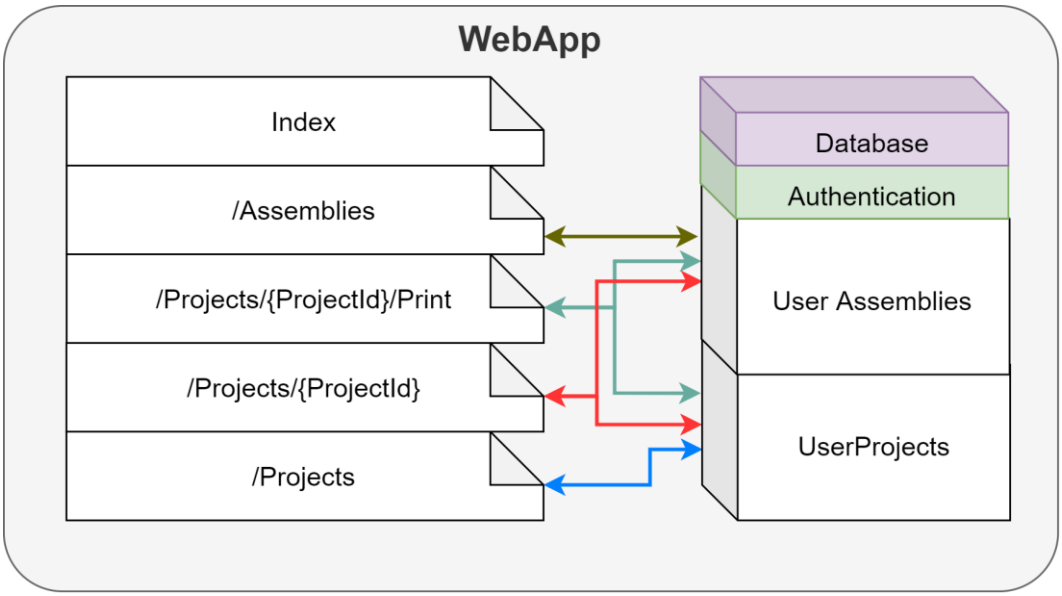
- **/Projects/{projectId}**

  Η σελίδα */Projects/{projectId}* αποτελεί το Ενσωματωμένο Περιβάλλον Ανάπτυξης (Integrated Development Environment ή IDE) της εφαρμογής. Εδώ, ο χρήστης δομεί και επεξεργάζεται το επιλεγμένο project που καθορίζει η παράμετρος `projectId`.

- **/Projects{projectId}/Print**

  Κατόπιν σχεδιασμού ενός project, η σελίδα */Projects/{projectId}/Print* επιτρέπει στους χρήστες να δημιουργήσουν ομάδες θεμάτων από κοινού με τις απαντήσεις τους. Παρέχει τρία πεδία που αναπαριστούν τα ορίσματα της συνάρτησης **`void Generate(int firstGroupId, int groupCount, int idStep)`** (βλ. *1.4.2. Exam Modules*, σελ. 21-22), και ένα κουμπί που ξεκινά την δημιουργία των ομάδων. Κατόπιν επιτυχούς ολοκλήρωσης της διαδικασίας, ξεκινά αυτόματα η λήψη τους.

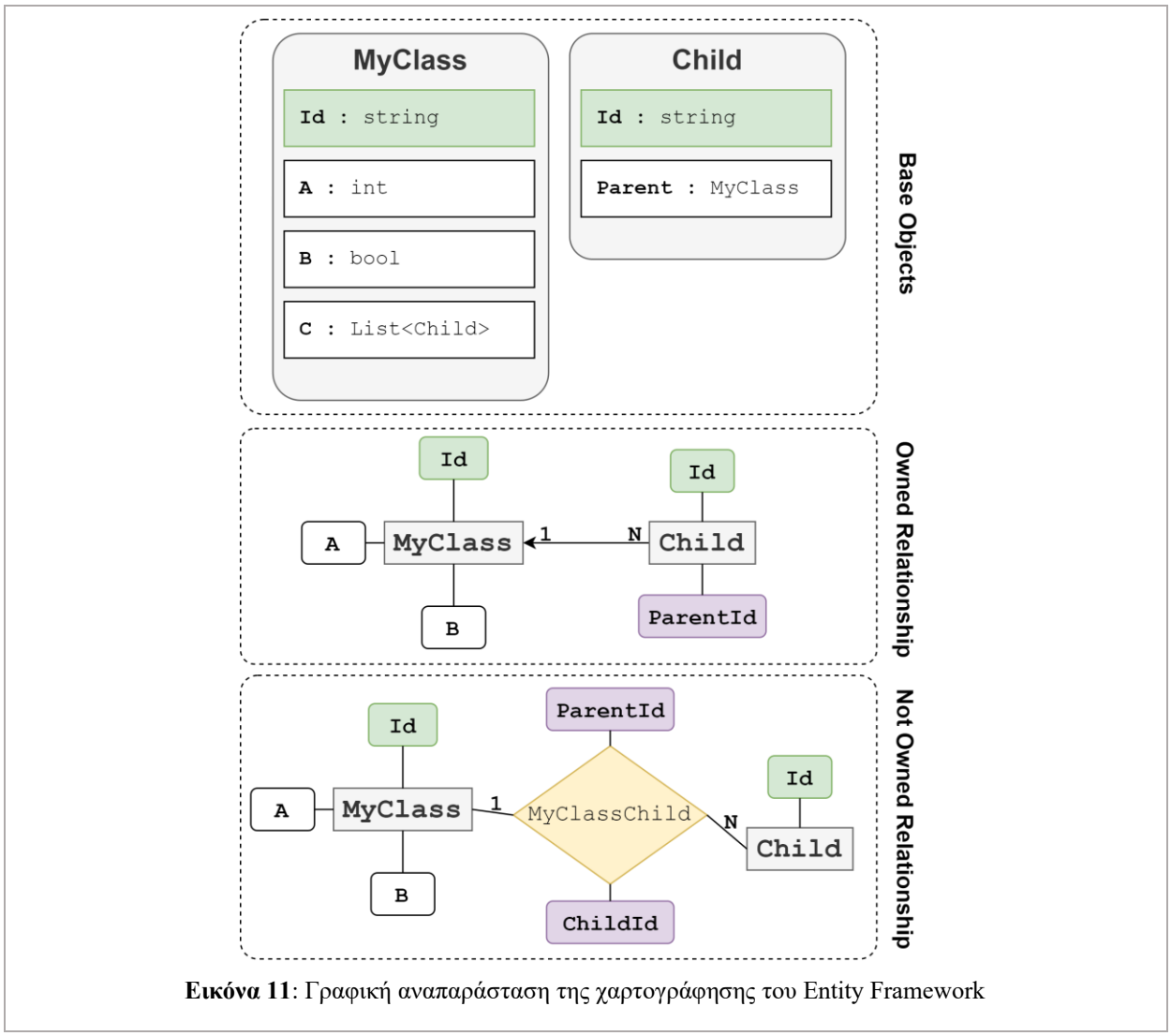
Η εφαρμογή διατηρεί επίσης ένα σύνολο σελίδων που αφορούν την Εγγραφή, Σύνδεση, Επεξεργασία και Αποσύνδεση χρηστών. Αυτές οι σελίδες είναι προ-υλοποιημένες από το *Identity Core* του *ASP.NET 5.0 Framework* και προσφέρουν μία άμεση και ασφαλή λύση *εξουσιοδότησης βασισμένης σε Χρήστες* (**user-based authorization**)



Εικόνα 10: Δομή του ExamKitchen WebApp

2.2. Βάση Δεδομένων και Ταυτοποίηση

Για την αποθήκευση των χρηστών και των πληροφοριών σχετικά με τα module packs και τα projects που φιλοξενούνται στην εφαρμογή, το WebApp χρησιμοποιεί μία **Postgres** βάση δεδομένων. Η βάση αυτή φιλοξενείται σε ξεχωριστό server και διαμορφώνεται αυτόματα με την βοήθεια του ***Entity Framework (EF) Core 6***, ένα πλαίσιο, δηλαδή, *Αντικειμενοστραφούς σχεσιακής χαρτογράφησης* (**Object-Relational Mapping**, **ORM**) που επιτρέπει την αυτόματη δημιουργία οντοτήτων (Entities) και σχέσεων (Relations) μέσω κλάσεων.



Εικόνα 11: Γραφική αναπαράσταση της χαρτογράφησης του Entity Framework

Η βάση δεδομένων της εφαρμογής απαρτίζεται απο τρεις κλάσεις οντοτήτων. Αναλυτικότερα:

- **`EkUser`**

  Αναπαριστώντας έναν χρήστη, η κλάση *EkUser* κληρονομεί την λειτουργικότητα της κλάσης *IdentityUser* που υλοποιεί όλα τα απαραίτητα πεδία και μεθόδους του *Identity Core* για τη διαδικασία ταυτοποίησης. Σε αυτά συγκαταλέγονται το **Id**, που αποτελεί μοναδικό αναγνωριστικό του χρήστη, το **Email**, το **Username** που, για τους σκοπούς της εφαρμογής, ταυτίζεται με το Email, και το *Hash του κωδικού πρόσβασής* (**PasswordHash**), το οποίο χρησιμοποιείται κατά για ταυτοποίηση. Επιπρόσθετα, διαθέτει δύο ακόμα πεδία. Το πεδίο **`assemblies`** διατηρεί μία λίστα με όλα τα module packs που έχουν ανέβει και χρησιμοποιούνται, ενώ το **`projects`** μία λίστα με τα project του χρήστη.

- **`AssemblyEntry`**

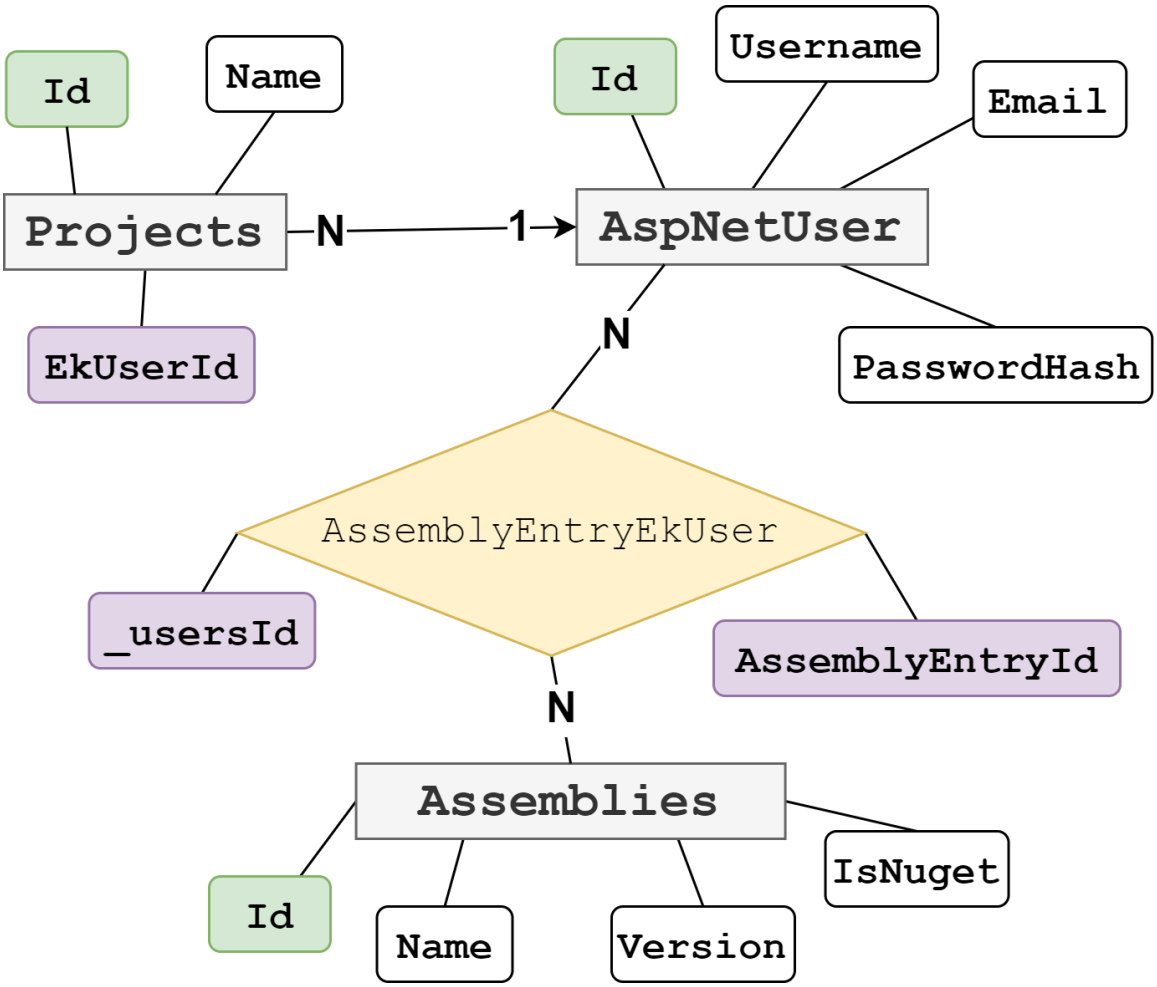
  Κάθε module pack που ανεβάζει ο χρήστης αναλύεται από την. Οι πληροφορίες που το συναποτελούν αποθηκεύονται εντός της βάσης δεδομένων, πλαισιώνονται από ένα αντικείμενο *Assembly Entry*, και περιλαμβάνουν:

  - το *Μοναδικό Αναγνωριστικό* του assembly (**`Id`**) που, στην προκειμένη περίπτωση ταυτίζεται με το MD5 Checksum του αρχείου που ανέβασε ο χρήστης
  - το *Όνομα* του assembly (**`Name`**)
  - ο *αριθμός έκδοσης* (**`Version`**) του assembly (π.χ. *v1.5.2*)
  - η *προέλευση* του assembly -κατά πόσο, δηλαδή, προέρχεται από Βιβλιοθήκη Δυναμικής Σύνδεσης ή Πακέτο Επέκτασης NuGet. (**`IsNuget`**)

- **`ProjectEntry`**

  Η δημιουργία ενός νέου project μέσω του ExamKitchen WebApp πυροδοτεί μία διμερή διαδικασία. Αρχικά, δημιουργείται ένα καινούριο JSON αρχείο που περιλαμβάνει το όνομα του project. Στη συνέχεια, οι πληροφορίες του νεοσύστατου project καταχωρούνται στην βάση δεδομένων αρωγή ενός τύπου δεδομένων ονόματι *Project Entry*. Το *Project Entry* συγκρατεί ένα Μοναδικό αναγνωριστικό (**`Id`**) του project και το όνομα (**`Name`**) που του έχει δώσει ο χρήστης.

Ενώ η συσχέτιση μεταξύ χρηστών και Projects είναι καθαρά ιδιοκτησιακή (ένα project μπορεί να υπάγεται κάτω από έναν και μόνο έναν χρήστη), ο τρόπος που η εφαρμογή διαχειρίζεται τα assemblies διαφέρει. Οικονομίας ένεκα, η εφαρμογή χρησιμοποιεί τη βάση δεδομένων σαν *λεξικό* (**dictionary**) και στοιχειοθετεί τα assemblies βάσει του MD5 Checksum τους. Ως εκ τούτου, αν ένας χρήστης ανεβάσει κάποιο assembly που έχει ήδη καταχωρηθεί από κάποιον άλλο, τότε δεν δημιουργείται διπλότυπο και, αντ' αυτού, η υπάρχουσα καταχώρηση αντιστοιχείται και στους δύο. Ανάλογα, μια καταχώρηση assembly δεν διαγράφεται από το σύστημα, παρά μόνο όταν όλοι οι χρήστες που σχετίζονται με αυτή την διαγράψουν. Βάσει των παραπάνω και μετά από την χαρτογράφηση που κάνει το Entity Framework, προκύπτει η βάση δεδομένων που απεικονίζεται ως εξής:



Εικόνα 12: Μοντέλο Οντοτήτων-Συσχετίσεων της βάσης δεδομένων του ExamKitchen WebApp

2.3. Πρόσβαση στην Βάση δεδομένων και τα assemblies

Προς χρήση της βάσης δεδομένων, των assemblies των χρηστών και του Runtime, η εφαρμογή ενέχει ένα σύνολο *υπηρεσιών* (**services**), που έκαστα παρέχουν *διαχειριζόμενη πρόσβαση* (**managed access**) σε έναν από τους παραπάνω τομείς. Η βασικότερη υπηρεσία του ExamKitchen WebApp ειδικεύεται στα δεδομένα του χρήστη. Έχοντας την ονομασία ***UserManager*** και διάρκεια ζωής που ταυτίζεται με αυτή της καρτέλας του browser, το εν λόγω service ελέγχει την κατάσταση ταυτοποίησης. Σε περίπτωση που ο χρήστης είναι συνδεδεμένος, ανασύρει τα δεδομένα του από την βάση δεδομένων. Ο User Manager δεν χρησιμοποιείται ποτέ απευθείας, αλλά μεσολαβεί ως γενεσιουργός παράγοντας διαφορετικών services που κατακερματίζουν περαιτέρω την πρόσβαση και παρέχουν πιο εξειδικευμένη λειτουργικότητα. Αυτές οι υπηρεσίες είναι:

- **Assembly Service**

  Το *Assembly Service* ειδικεύεται στη διαχείριση των assemblies ενός χρήστη. Είναι υπεύθυνο για τον έλεγχο συμβατότητας και την προσθήκη κάθε assembly που φορτώνει ο χρήστης, καθώς και την διαγραφή κάποιου assembly που ο χρήστης έχει ήδη φορτώσει στο παρελθόν. Δεκτά, γίνονται μόνο έγκυρα .NET 5.0 assemblies που δεν βασίζονται σε συγκεκριμένη πλατφόρμα (πχ Windows ή Linux· γνωστά και ως *Platform-Agnostic*) ή αρχιτεκτονική (πχ x86 ή arm64· γνωστά και ως *Architecture-Agnostic*), καθώς και NuGet packages που περιέχουν assemblies με τα παραπάνω χαρακτηριστικά.

- **Project Service**

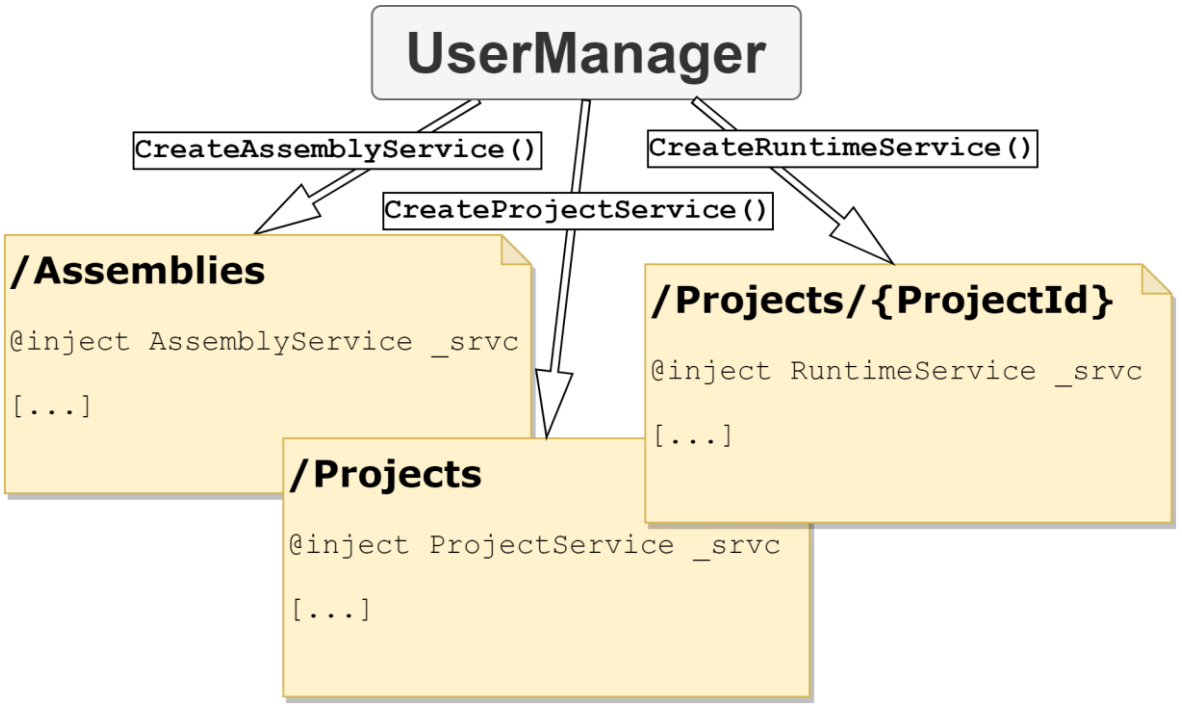
  Η υπηρεσία *Project Service* διαχειρίζεται τα projects των χρηστών. Περιέχει την απαραίτητη λειτουργικότητα για την δημιουργία, διαγραφή και μετονομασία ενός project

- **Runtime Service**

  Η πολυπλοκότερη όλων των υπηρεσιών είναι η υπηρεσία του Runtime. Εδώ φορτώνονται όλα τα assemblies του χρήστη με τα Modules που αυτά περιέχουν. Αποτελεί την «*καρδιά*» του IDE του WebApp, καθώς καθοδηγεί τον χρήστη στην αναζήτηση και την παραμετροποίηση των Modules για τον σχεδιασμό των projects.

Επιπρόσθετα, το *Runtime Service* είναι υπεύθυνο για την σύνθεση ενός project σύμφωνα με τα δεδομένα του χρήστη και την γένεση θεμάτων μέσω αυτού.

Για λόγους ασφαλείας, η διαδικασία φόρτωσης των assemblies είθισται να λαμβάνει χώρα εντός *απομονωμένων «δοχείων»* (**sandboxed containers**), αρωγή κάποιου containerization software όπως το ***docker***. Εντούτοις, καθώς η ασφάλεια της εφαρμογής και δόμηση ενός τέτοιου περιβάλλοντος αποτελούν ιδιαίτερα μεγάλη πρόκληση που δεν συγκαταλέγεται στους στόχους του παρόντος πονήματος, συμμορφωθήκαμε με μία πιο απλοϊκή - πλην αφελή - λογική: Τα assemblies φορτώνονται απευθείας στο Runtime του WebApp μίας *αναλώσιμη* (**disposable**) κλάση ονόματι ***AssemblyCollection***. Η εν λόγω κλάση βασίζεται στην κλάση *AssemblyLoadContext* και επιτρέπει την απομονωμένη φόρτωση και αξιοποίηση των assemblies εντός της εφαρμογής.



Εικόνα 13: Διαδικασία παροχής των Services στις σελίδες που τα χρειάζονται

2.4. Μοντέλα, Όψεις και Μοντέλα Προβολής

Για την αρμονική συνεργασία γραφικού περιβάλλοντος και υποκείμενου πλαισίου, υιοθετήθηκε το σχεδιαστικό μοτίβο *Μοντέλων, Όψεων και Μοντέλων Προβολής* (***Model – View – ViewModel*** design pattern· γνωστό και ως **MVVM**). Σύμφωνα με το MVVM, κάθε δομή δεδομένων που συμβάλλει στο γραφικό περιβάλλον κατακερματίζεται σε τρεις βασικούς τομείς: (1) Το Μοντέλο που εκπροσωπεί την προγραμματιστική απεικόνιση του αντικειμένου (π.χ. Exam Module, Question Module κλπ.), (2) την Όψη που παρέχει τη γραφική αναπαράσταση των εν λόγω δεδομένων με κατανοητό για τον χρήστη τρόπο, και (3) το Μοντέλο Προβολής που εμφωλεύει την λειτουργικότητα που απαιτείται για την σύνδεση ενός μοντέλου με την αντίστοιχη όψη, παρέχοντας όλους τους απαραίτητος μετασχηματισμούς που απαιτούνται για την αντιστοίχιση. Για τις ανάγκες του WebApp δημιουργήθηκαν τρία ζεύγη Μοντέλων Προβολής και Όψεων, τα οποία, πέρα τη δομική σταθερότητα που απαιτείται για την δόμηση του γραφικού περιβάλλοντος, προσφέρουν την δυνατότητα σειριοποίησης και αποσειριοποίησής τους σε/από αρχεία JSON. Υπενθυμίζεται ότι αυτός ο τύπος αρχείων χρησιμοποιείται για την διατήρηση των Projects στην εφαρμογή. Τα υπό συζήτηση ViewModels και τα αντίστοιχα Views τους αναλύονται ως εξής:

- **Exam Model**

  Το ViewModel ενός Exam Module ονομάζεται *Exam Model*. Σε αντίθεση με τα Δομοστοιχεία Εξέτασης που εκπροσωπούν λειτουργικά ένα project, τα Μοντέλα Εξέτασης αποσκοπούν στην δομική τους αναπαράσταση. Ως εκ τούτου, το αποτέλεσμα σειριοποίησης ενός Exam Model ταυτίζεται με το περιεχόμενο του αρχείου JSON που περιγράφει το Project και αποθηκεύεται στο WebApp. Εκτός του **τύπου Exam Module** που εκπροσωπούν και **τα ViewModels παραμέτρων** που αντιστοιχούν στις παραμέτρους τους, τα Exam Models διατηρούν το **Μοναδικό Αναγνωριστικό**, το **Όνομά** και τα **ViewModels των ερωτήσεων** του project.

  Την αλυσίδα του MVVM για το συγκεκριμένο αντικείμενο συμπληρώνει το ***Exam View***, ένα Blazor Component που αποδίδει γραφικά ένα HTML τμήμα που αφορά την επεξεργασία ενός Project. Χωρίζεται σε πέντε τμήματα που αφορούν (1) τις βασικές πληροφορίες του Exam Module, όπως το όνομα και την περιγραφή του, (2) τις παραμέτρους του Module, όταν αυτές υπάρχουν, (3) έναν Selector που δίνει την

δυνατότητα αλλαγής του Exam Module που εκπροσωπεί το υποκείμενο μοντέλο προβολής, (4) έναν Selector που επιτρέπει την αναζήτηση και προσθήκη καινούριων ερωτημάτων και (5) μία λίστα διαχείρισης των υπαρχόντων ερωτημάτων.

- **Question Model**

  Παρόμοια, τα Exam Models, τα Question Models αποτελούν Μοντέλα Προβολής των Question Modules. Οι πληροφορίες που διατηρεί ένα Μοντέλο Ερώτησης είναι **ο τύπος Question Module** που εκπροσωπεί, μία λίστα με **τα ViewModels παραμέτρων** που του αντιστοιχούν και μία λίστα με **τα Question Models υποερωτημάτων**, όταν αυτό υποστηρίζεται.

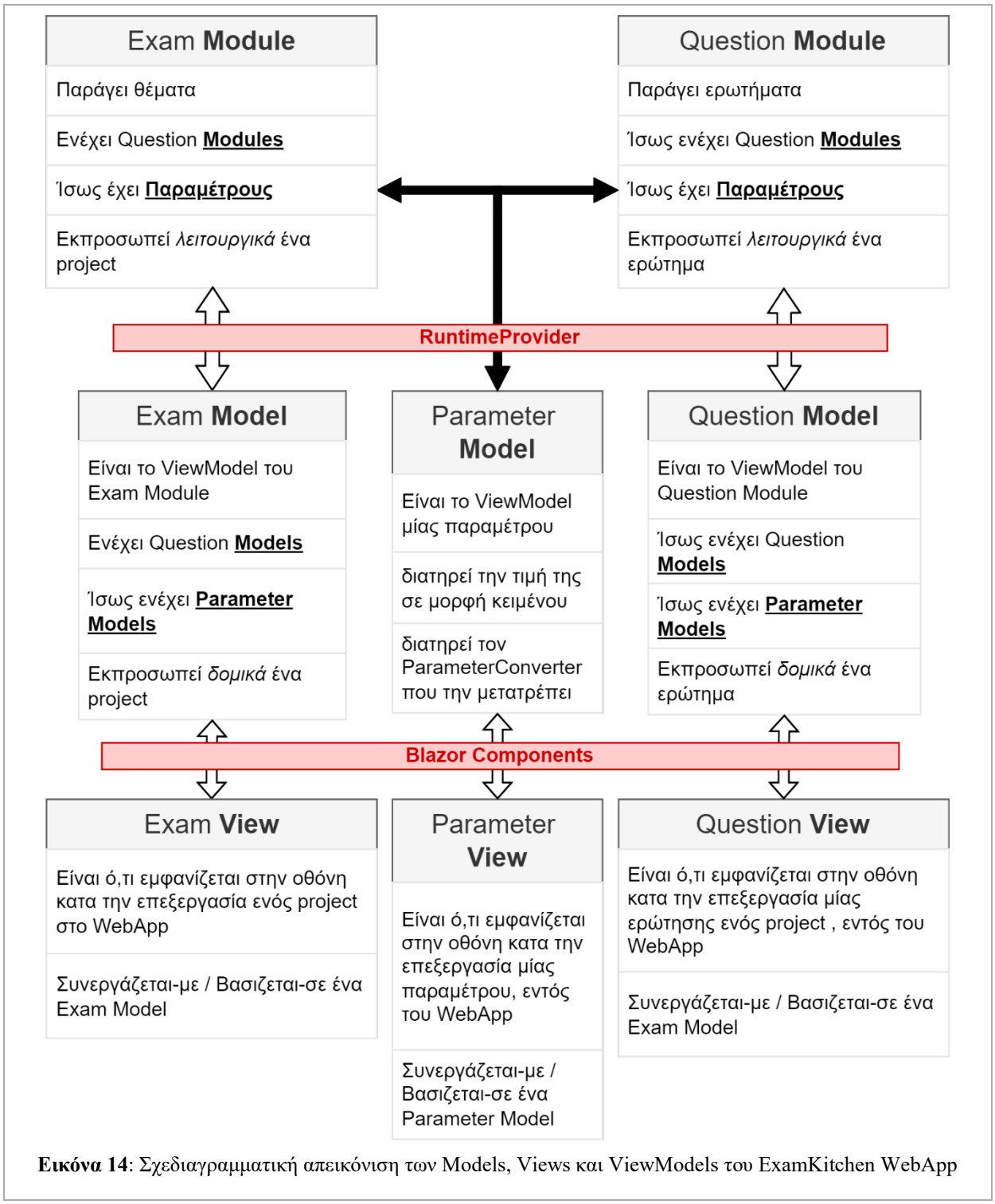
  Η Όψη που σχετίζεται με τα Question Models ονομάζεται ***Question View***. Πρόκειται για ακόμα ένα Blazor Component που χωρίζεται σε τρία HTML τμήματα: (1) τις παραμέτρους του Module, όταν αυτές υπάρχουν, (2) έναν Selector που δίνει την δυνατότητα αλλαγής του Question Module που εκπροσωπεί το υποκείμενο μοντέλο προβολής και (3) έναν Selector που, όταν το ερώτημα επιτρέπει την ύπαρξη υποερωτημάτων, επιτρέπει την αναζήτηση και προσθήκη καινούριων. Ταυτόχρονα, διαμορφώνει μία λίστα διαχείρισης των επιμέρους ερωτήσεων

- **Parameter Model**

  Το μοντέλο προβολής μιας παραμέτρου είναι το μοναδικό που δεν βασίζεται σε κάποιο μοντέλο. Αντ' αυτού συμπληρώνει την εξειδικευμένη λειτουργικότητα που απαιτείται για την σωστή προβολή των παραμέτρων ενός Module. Τα *Parameter Models* διατηρούν το **όνομα** της παραμέτρου που εκπροσωπούν, τον **τύπο** της, την **περιγραφή** της, **την τιμή της σε σειριοποιημένη μορφή**, καθώς και **τον *Parameter Converter* με τον οποίο σχετίζεται**, προκειμένου να προσφέρουν ανάδραση και συντακτική καθοδήγηση κατά την επεξεργασία της τιμής της.

  Ως View, τα Parameter Models χρησιμοποιούν ένα Blazor Component ονόματι ***Param View***. Αυτό αποτελείται από μία φόρμα που χρησιμοποιεί το όνομα της παραμέτρου ως τίτλο, μία είσοδο που διατηρεί την τιμή της, ένα pop-up που την περιγράφει σύντομα και ένα μικρό βοηθητικό πλαίσιο που εμφανίζεται μόνο στην περίπτωση συντακτικού λάθους κατά την μεταβολή της τιμής. Όταν ο Parameter Converter της παραμέτρου είναι ένας από τους *Προεπιλεγμένους Μετατροπείς Παραμέτρων*, η είσοδος του View παίρνει μορφή εξειδικευμένη σε αυτόν τον τύπο δεδομένων (πχ: δέχεται μόνο ψηφία,

για `int`, `float` ή `double`, drop-down list για παραμέτρους τύπου απαρίθμησης, checkbox για `Boolean` κλπ.). Σε κάθε άλλη περίπτωση, η εν λόγω είσοδος έχει την μορφή TextArea.



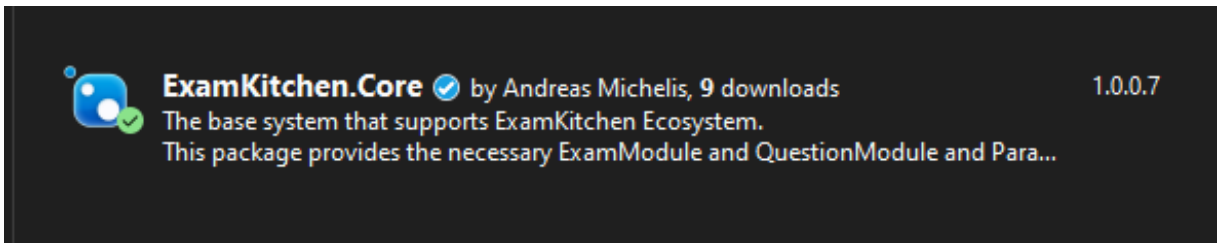
Εικόνα 14: Σχεδιαγραμματική απεικόνιση των Models, Views και ViewModels του ExamKitchen WebApp

Κεφάλαιο 3: Walkthrough

Για μια ενδελεχέστερη παρουσίαση του workflow που προτείνει το ExamKitchen, ακολουθούν βήμα-βήμα ένα παράδειγμα Exam Module και δύο Question Modules που χρησιμοποιήθηκαν για την κατασκευή δοκιμαστικών θεμάτων. Η δημιουργία και αξιοποίηση των δομοστοιχείων κατακερματίζεται σε επιμέρους τμήματα, τα οποία αναλύονται συστηματικά παρακάτω.

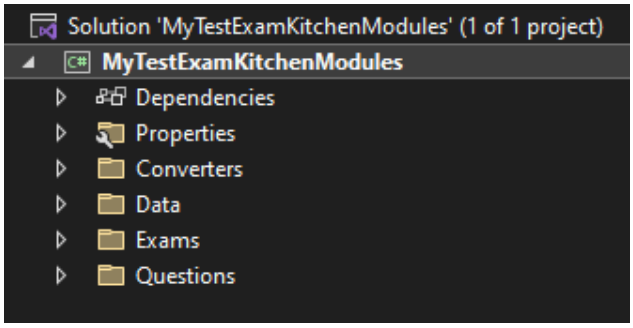
3.1. Κατασκευή module pack

Η επινόηση των Modules ξεκινά στο *Visual Studio* με τη δημιουργία ενός νέου *Class Library Project* βασισμένο σε *.NET 5.0*. Κάθε project που υλοποιεί Modules για το ExamKitchen εξαρτάται λειτουργικά από το Framework. Για την ικανοποίηση αυτής της εξάρτησης, στο project ενσωματώνεται το NuGet Package του ExamKitchen Core το οποίο ανασύρεται από το GitHub Repository, όπου φιλοξενείται το οικοσύστημα του πονήματος.



Εικόνα 15: Ενσωμάτωση του ExamKitchen Core σε καινούριο Project C#

Στην συνέχεια δημιουργούνται τέσσερις υποφακέλοι ονόματι (1) **Exams,** (2) **Questions,** (3) **Converters** και (4) **Data**. Οι φάκελοι Questions και Exams φιλοξενούν τα δομοστοιχεία ερώτησης και εξέτασης που θα δημιουργηθούν, ο Converters τους μετατροπείς των αξιοποιήσιμων παραμέτρων και ο Data όλες τις βοηθητικές δομές που δεν σχετίζονται άμεσα με το ExamKitchen.



Εικόνα 16: Αρχική δομή του Visual Studio Project

Το πρώτο Module που δημιουργήθηκε είναι ένα **Exam Module**. Πρόκειται για μια ιδιαίτερα απλή υλοποίηση που βασίζεται στο CSS πλαίσιο του **Bootstrap** για την δόμηση ενός μινιμαλιστικού πλην λειτουργικού αποτελέσματος. Το εν λόγω Module ονομάστηκε **MyExamTemplate**, με φιλική ονομασία «*Best Exam Module*».

MyExamTemplate.cs

```
[ExamMetadata(
    "Best Exam Module",
    "Andreas Michelis",
    "This is the **best** exam module money can buy.")]
public class MyExamTemplate : ExamModule
{
    protected override void ConfigureExam(ExamConfiguration config)
    {
        throw new System.NotImplementedException();
    }

    protected override ExamSolutionTuple PrepareExam(
        int groupID, QuestionModuleResult[] questions)
    {
        throw new System.NotImplementedException();
    }

    protected override string PrepareQuestion(QuestionModuleResult question)
    {
        throw new System.NotImplementedException();
    }

    protected override string PrepareAnswer(QuestionModuleResult question)
    {
        throw new System.NotImplementedException();
    }
}
```

Εικόνα 17: Το MyExamTemplate πριν την υλοποίηση

Η υλοποίηση του MyExamTemplate προϋποθέτει την περιγραφή των παραμέτρων που ενέχει. Εν προκειμένω, τον τίτλο του μαθήματος, το όνομα του εξεταστή και την χρονιά διεξαγωγής των εξετάσεων. Για την υλοποίηση του MyExamTemplate επιλέχθηκε η αυτούσια αποτύπωση των παραμέτρων σε κάθε θέμα, αλλά ο τρόπος χρήσης των παραμέτρων ενός Exam Module επαφίεται στις ανάγκες της εκάστοτε υλοποίησης.

MyExamTemplate.cs

```cs
[ExamMetadata(
    "Best Exam Module",
    "Andreas Michelis",
    "This is the **best** exam module money can buy.")]
public class MyExamTemplate : ExamModule
{
    [ParameterDefinition(
        typeof(string),
        "The *name* of the **course** to be examined.",
        "Untitled Exam")]
    protected string CourseName { get; set; }


    [ParameterDefinition(
        typeof(string),
        "The name of the person *conducting the evaluation*.",
        "Anonymous")]
    protected string ExaminerName { get; set; }


    [ParameterDefinition(
        typeof(int),
        "Year of the examination.",
        "2022")]
    protected int Year { get; set; }

    // [ ... ]
}
```

Εικόνα 18: Οι παράμετροι του MyExamTemplate

Στη συνέχεια υλοποιήθηκε η παραμετροποίηση του Exam Module. Σε αυτή δηλώνεται η χρήση Bootstrap και Μαθηματικών στο MarkDown Pipeline που χρησιμοποιείται κατά την γένεση θεμάτων, καθώς και ένα μικρό CSS αρχείο που διαμορφώνει κάποιες λεπτομέρειες για την καλύτερη παρουσίαση των θεμάτων.

MyExamTemplate.cs

```
protected override void ConfigureExam(ExamConfiguration config)
{
    config.SetPipeline(x ⇒ {
            x.UseBootstrap();
            x.UseListExtras();
            x.UseMathematics();
        });

    config.SetCssFile("main", @"
        .ek-question {
            margin: 20px;
            width: calc (100% - 40px);
        }

        .ek-header {
            display: flex;
            width: 100%;
        }

        .ek-title {
            flex-grow: 1;
            font-size: 130%;
            font-weight: bold;
        }

        .ek-score {
            flex-grow: 0;
            font-size: 110%;
            font-weight: bold;
            margin: auto auto auto 5px;
            padding-left: 5px;
            border-left: 1px solid white;
        }

        .ek-body {
            font-size: 80%;
            padding: 5px;
        }");
}
```

Εικόνα 19: Μέθοδος παραμετροποίησης του MyExamTemplate

Ακολουθεί η μέθοδος προετοιμασίας των θεμάτων. Εδώ δομείται το Header για τις σελίδες *Εξέτασης* και *Απαντήσεων*, ενώ προς παραγωγή του τελικού αποτελέσματος καλούνται αναδρομικά οι ρουτίνες μορφοποίησης ερωταπαντήσεων.

MyExamTemplate.cs

```
protected override ExamSolutionTuple PrepareExam(
    int groupID, QuestionModuleResult[] questions)
{
    var header = @$"
        <div class='card ml-2 ms-2 mr-2 me-2'
             style='margin-bottom: 30px;'>
            <div class='card-header'>
                <h1 class='m-auto text-center'>
                    {CourseName}
                </h1>
            </div>
            <div class='card-body'>
                <div><b>Author: </b>{ExaminerName}</div>
                <div><b>Year: </b>{Year}</div>
            </div>
        </div>
        ";

    var exam = new StringBuilder(header);
    var solutions =new StringBuilder(header);

    foreach (var question in questions) {
        exam.AppendLine(PrepareQuestion(question));
        solutions.AppendLine(PrepareAnswer(question));
    }

    return new ExamSolutionTuple(
        exam.ToString(),
        solutions.ToString());
}
```

Εικόνα 20: Μέθοδος προετοιμασίας θεμάτων του MyExamTemplate

Οι μέθοδοι προετοιμασίας Ερώτησης και Απάντησης παράγουν δομικά πανομοιότυπο αποτέλεσμα χρησιμοποιώντας τα κατάλληλα πεδία (*Question Pre/Post text* και *Answer Pre/Post text* αντίστοιχα).

MyExamTemplate.cs

```
protected override string PrepareQuestion(QuestionModuleResult question)
{
    var output = new StringBuilder(@$"
        <div class='ek-question card'>
            <div class='card-header'>
                <div class='ek-header'>
                    <span class='ek-title'>{question.Title}</span>
        ");

    if (question.HasScore) {
        output.AppendLine($"<span class='ek-score'>{question.Score}</span>");
    }

    output.AppendLine(@$"
            </div>
        </div>
        <div class='card-body ek-body'>
            {(question.HasQuestionPreText ? question.QuestionPreText : "")}
        ");

    if (question.HasSubQuestions){
        foreach (var sub in question.SubQuestions)
            output.AppendLine(PrepareQuestion(sub));
    }

    output.AppendLine(@$"
                {(question.HasQuestionPostText ? question.QuestionPostText : "")}
            </div>
        </div>
        ");
}
```

Εικόνα 21: Μέθοδος προετοιμασίας ερώτησης του MyExamTemplate

MyExamTemplate.cs

```
protected override string PrepareAnswer(QuestionModuleResult question)
{
    var output = new StringBuilder(@$"
        <div class='ek-question card'>
            <div class='card-header'>
                <div class='ek-header'>
                    <span class='ek-title'>{question.Title}</span>
        ");

    if (question.HasScore) {
        output.AppendLine($"<span class='ek-score'>{question.Score}</span>");
    }

    output.AppendLine(@$"
            </div>
        </div>
        <div class='card-body ek-body'>
            {(question.HasAnswerPreText ? question.AnswerPreText : "")}
        ");

    if (question.HasSubQuestions){
        foreach (var sub in question.SubQuestions)
            output.AppendLine(PrepareQuestion(sub));
    }

    output.AppendLine(@$"
                {(question.HasAnswerPostText ? question.AnswerPostText : "")}
            </div>
        </div>
        ");
}
```

Εικόνα 22: Μέθοδος προετοιμασίας απάντησης του MyExamTemplate

Στη συνέχεια σχηματίστηκαν τα Question Modules. Για τους σκοπούς του παραδείγματος σχηματίστηκαν δύο δομοστοιχεία ερωτήσεων. Το πρώτο είναι ένα Module *Τυχαιότητας* που παράγει τουλάχιστον μία μαθηματική πρόσθεση. Οι παράμετροι που ενέχει παραχωρούν στον χρήστη τον έλεγχο του **πλήθους των προσθέσεων**, του **αριθμού των προσθετέων** κάθε πρόσθεσης, καθώς και το **πλήθος των ψηφίων κάθε παράγοντα**.

MySumsQuestion.cs

```cs
[QuestionMetadata(
    "Batch sum calculations module",
    "Andreas Michelis",
    "This module generates random sums to be calculated by the examinee.")]
public class MySumsQuestion : QuestionModule
{
    // Provides the necessary randomness
    private static readonly Random Rng = new Random();

    // ------------------------------------------------------------------------
    [ParameterDefinition( typeof(int),
                          "The number individual sums to be calculated.\n" +
                                   "\nMust be a **Positive** integer.",
                          "1"
        )]
    public int Count { get; set; }
    // ------------------------------------------------------------------------
    [ParameterDefinition( typeof(int),
                          "The number of factors to be used.\n" +
                                   "\nMust be a **Positive** integer over *2*.",
                          "2"
        )]
    public int FactorCount { get; set; }
    // ------------------------------------------------------------------------
    [ParameterDefinition( typeof(int),
                          "The number of digits each factor should contain\n" +
                                   "\nMust be a **Positive** integer over *1*.",
                          "1")]
    public int DigitsCount { get; set; }
    // ------------------------------------------------------------------------

    protected override void Prepare(QuestionModuleResult result, int groupId)
    {
        result.SetTitle("Calculate the Following Sum(s)");

        var sumsQ = new StringBuilder();                // Question's Body
        var sumsA = new StringBuilder();                // Answer's Body
        var min = (int)Math.Pow(10, DigitsCount);       // Minimum factor Value
        var max = (int)Math.Pow(10, DigitsCount + 1);   // Maximum factor Value

        for (var c = 0; c < Count; c++) { // For each sum to be generated
            var factors = new int[FactorCount];  // The factor list of a sum
            // Generation of random numbers to fill the factor list
            for (var i = 0; i < FactorCount; i++) factors[i] = Rng.Next(min, max);
            var sum = factors.Sum(); // The sum of the generated factors

            sumsQ.AppendLine($"- $$ {string.Join(" + ", factors)} = ? $$");
            sumsA.AppendLine( $"- $$ {string.Join(" + ", factors)} = {sum} $$");
        }

        result.SetQuestion(sumsQ.ToString(), "");
        result.SetAnswer(sumsA.ToString(), "");
    }
}
```

Εικόνα 23: Δομή MySumsQuestion

Το δεύτερο Question Module που υλοποιήθηκε ειδικεύεται στην παραγωγή ερωτήσεων *πολλαπλής επιλογής*. Το εν λόγω δομοστοιχείο περιέχει μονάχα μία παράμετρο χειροποίητου τύπου ονόματι *MulChoiceDataContainer*.

MulChoiceDataContainer.cs

```
public class MulChoiceDataContainer
{
    public string Question { get; set; }
    public int ChoiceCount { get; set; }
    public int CorrectCount { get; set; }
    public string[] WrongAnswers { get; set; }
    public string[] CorrectAnswers { get; set; }
}
```

Εικόνα 24: Κλάση MulChoiceDataContainer

Για την μετατροπή της παραμέτρου δημιουργήθηκε ένας Μετατροπέας Παραμέτρου που ειδικεύεται στον νεοσύστατο τύπο αντικειμένου ονόματι **MulChoiceConverter**.

MulChoiceConverter.cs

```
public class MulChoiceConverter : ParameterConverter
{
    public MulChoiceConverter()
        : base(typeof(MulChoiceDataContainer)) // parameter type abstraction
    { }

    private static string defaultString
        ⇒  "No Question" +
           "\n——\n" +
           "4:1" +
           "\n——\n" +
           "Correct1\nCorrect2\nCorrect3\nCorrect4" +
           "\n——\n" +
           "Wrong1\nWrong2\nWrong3\nWrong4\nWrong5\nWrong6\nWrong7\nWrong8";


    private static MulChoiceDataContainer defaultObject
        ⇒  new MulChoiceDataContainer() {
               Question = "No Question",
               ChoiceCount = 4,
               CorrectCount = 1,
               CorrectAnswers = new[] {
                   "Correct1", "Correct2", "Correct3", "Correct4"
               },
               WrongAnswers = new[] {
                   "Wrong1", "Wrong2", "Wrong3", "Wrong4",
                   "Wrong5", "Wrong6", "Wrong7", "Wrong8"
               }
           };

    // [ ... ]
}
```

Εικόνα 25: Μετατροπέας MulChoiceConverter

MulChoiceConverter.cs

```
public override string Serialize(object value)
{
    if (value is not MulChoiceDataContainer val) return defaultString;

    return $"{val.Question}\n——\n" +
           $"{val.ChoiceCount}:{val.CorrectCount}\n——\n" +
           $"{string.Join("\n", val.CorrectAnswers)}\n——\n" +
           $"{string.Join("\n", val.WrongAnswers)}";
}


public override object Parse(string value)
{
    if (string.IsNullOrWhiteSpace(value) || !value.Contains("\n——\n"))
        return defaultObject;

    var sections = value.Split("\n——\n");
    if (sections.Length ≠ 4 || sections.Any(string.IsNullOrWhiteSpace))
        return defaultObject;

    var data = new MulChoiceDataContainer { Question = sections[0].Trim() };

    var counts = sections[1].Split(":");
    if (counts.Length ≠ 2 || counts.Any(x ⇒ !int.TryParse(x, out _)))
        return defaultObject;

    (data.ChoiceCount, data.CorrectCount) =
        (int.Parse(counts[0]), int.Parse(counts[1]));

    data.CorrectAnswers = sections[2]
        .Split("\n", StringSplitOptions.RemoveEmptyEntries);

    data.WrongAnswers = sections[3]
        .Split("\n", StringSplitOptions.RemoveEmptyEntries);

    return data;
}
```

Εικόνα 26: Μετατροπέας MulChoiceConverter - Σειριοποίηση και αποσειριοποίηση

MulChoiceConverter.cs

```
public override bool CanParse(string value, out string errorMessage)
{

    errorMessage = null;
    if (string.IsNullOrWhiteSpace(value)) {
        errorMessage = "Empty input detected";
        return false;
    }

    if (!value.Contains("\n——\n")) {
        errorMessage = "Check your syntax. No '——' sections found.";
        return false;
    };

    var sections = value.Split("\n——\n");
    if (sections.Length ≠ 4 || sections.Any(string.IsNullOrWhiteSpace)) {
        errorMessage = "All for sections are needed";
        return false;
    }

    var counts = sections[1].Split(":");
    if (counts.Length ≠ 2 || counts.Any(x ⇒ !int.TryParse(x, out _))) {
        errorMessage = "The counts section has no valid syntax " +
                       "(choice_count:correct_count)";
        return false;
    };

    var (choiceCount, correctCount) =
        (int.Parse(counts[0]), int.Parse(counts[1]));
    if (choiceCount < correctCount) {
        errorMessage = "Asking for more correct answers than the number of choices";
        return false;
    }

    var cLen = sections[2]
        .Split("\n", StringSplitOptions.RemoveEmptyEntries)
        .Length;
    var wLen = sections[3]
        .Split("\n", StringSplitOptions.RemoveEmptyEntries)
        .Length;

    if (cLen < correctCount) {
        errorMessage = "Insufficient number of Correct Answers";
        return false;
    }

    if (wLen < choiceCount - correctCount) {
        errorMessage = "Insufficient number of Wrong Answers";
        return false;
    }

    return true;
}
```

Εικόνα 27: Μετατροπέας MulChoiceConverter - Συντακτική ανάλυση

Προμηθεύοντας την μοναδική παράμετρο που περιέχει το δομοστοιχείο με την **ερώτηση**, το **πλήθος των διαθέσιμων επιλογών**, τον **αριθμό των ορθών απαντήσεων**, μία λίστα με **σωστές επιλογές αυτές καθαυτές** και μία λίστα με τις **λάθος απαντήσεις**, το Module παράγει ντετερμινιστικά μία ερώτηση πολλαπλής επιλογής κάθε φορά που καλείται.

MyMulChoiceQuestion.cs

```
[QuestionMetadata("My Multiple Choice Module",
                  "Andreas Michelis",
                  "This module is capable of creating multiple choice questions.")]
public class MyMulChoiceQuestion : QuestionModule
{
    // ------------------------------------------------------------------------
    [ParameterDefinition(
        typeof(MulChoiceDataContainer),
        "Contains all data regarding a **multiple choice** question.\n\n" +
        "This field should be filled with the following syntax:\n\n" +
        "```\n{Question}" +
        "\n--\n" +
        "{Number of choices}:{Number of correct answers}" +
        "\n--\n" +
        "{Correct Answer 1}\n{Correct Answer 2}\n[ ... ]" +
        "\n--\n" +
        "{Wrong Answer 1}\n{Wrong Answer 2}\n[ ... ]\n" +
        "```",
        "No Question" +
        "\n--\n" +
        "4:1" +
        "\n--\n" +
        "Correct1\nCorrect2\nCorrect3\nCorrect" +
        "4\n--\n" +
        "Wrong1\nWrong2\nWrong3\nWrong4\nWrong5\nWrong6\nWrong7\nWrong8",
        typeof(MulChoiceConverter))]
    public MulChoiceDataContainer QuestionData { get; set; }
    // ------------------------------------------------------------------------

    // [ ... ]
}
```

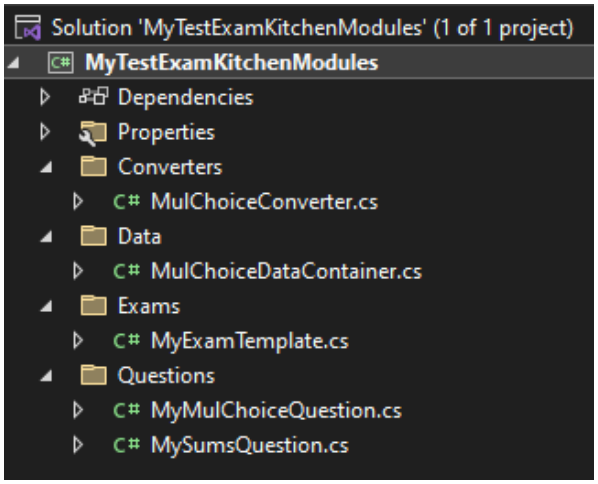
Εικόνα 28: Κλάση MyMulChoiceQuestion - Δήλωση παραμέτρων

MyMulChoiceQuestion.cs

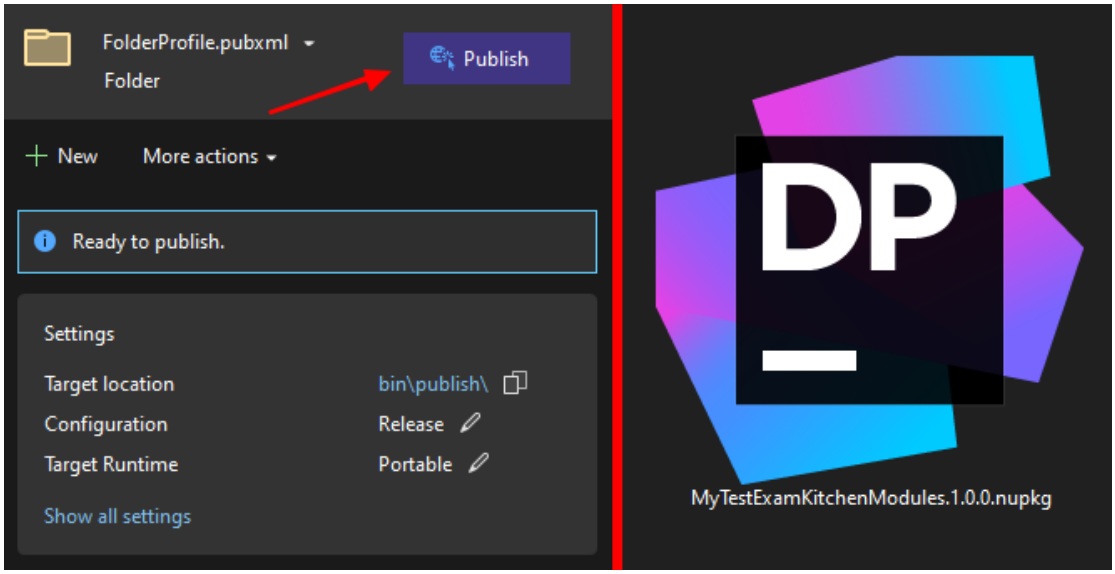
```
protected override void Prepare(QuestionModuleResult result, int groupId)
{
    result.SetTitle("Select the correct answer(s)");
    var rng = new Random(groupId); // Deterministic Randomness, based on group's ID

    // Convert Correct and Wrong choices to lists
    var correctList = new List<string>(QuestionData.CorrectAnswers);
    var wrongList = new List<string>(QuestionData.WrongAnswers);

    var choices = new List<(string, bool)>(); // Will contain the selected choices

    // Randomly select Correct Choices and store them in random order
    for (var i = 0; i < QuestionData.CorrectCount; i++) {
        var j = rng.Next(correctList.Count);
        choices.Insert(rng.Next(choices.Count),(correctList[j], true));
        correctList.RemoveAt(j);
    }

    // Randomly select Wrong Choices and store them in random order
    for (var i = 0; i < QuestionData.ChoiceCount - QuestionData.CorrectCount; i++) {
        var j = rng.Next(wrongList.Count);
        choices.Insert(rng.Next(choices.Count),(wrongList[j], true));
        wrongList.RemoveAt(j);
    }

    // Extract the correct choices
    var correct = choices.Where(x => x.Item2);

    var q = $"{QuestionData.Question}\n\n" +
            $"{string.Join("\n", choices.Select(x => "- " + x.Item1))}";

    var a = $"{QuestionData.Question}\n\n" +
            $"{string.Join("\n", correct.Select(x => "- " + x.Item1))}";

    result.SetQuestion(q, "");
    result.SetAnswer(a, "");
}
```

Εικόνα 29: Κλάση MyMulChoiceQuestion - Υλοποίηση της μεθόδου γένεσης θεμάτων

Όταν η βιβλιοθήκη ολοκληρωθεί και το project αποκτήσει δομή αντίστοιχη της Εικόνας 30, δημοσιοποιείται (**publish**) σε αρχείο **NuGet** με ρυθμίσεις (1) Configuration: **Release** και (2) Target Runtime: **Portable**. Οι παραπάνω ρυθμίσεις επιτρέπουν στον Compiler της C# να βελτιστοποιήσει το module pack αυξάνοντας απόδοσή του και οδηγούν στην δημιουργία *Platform/Architecture Agnostic* αποτελέσματος.



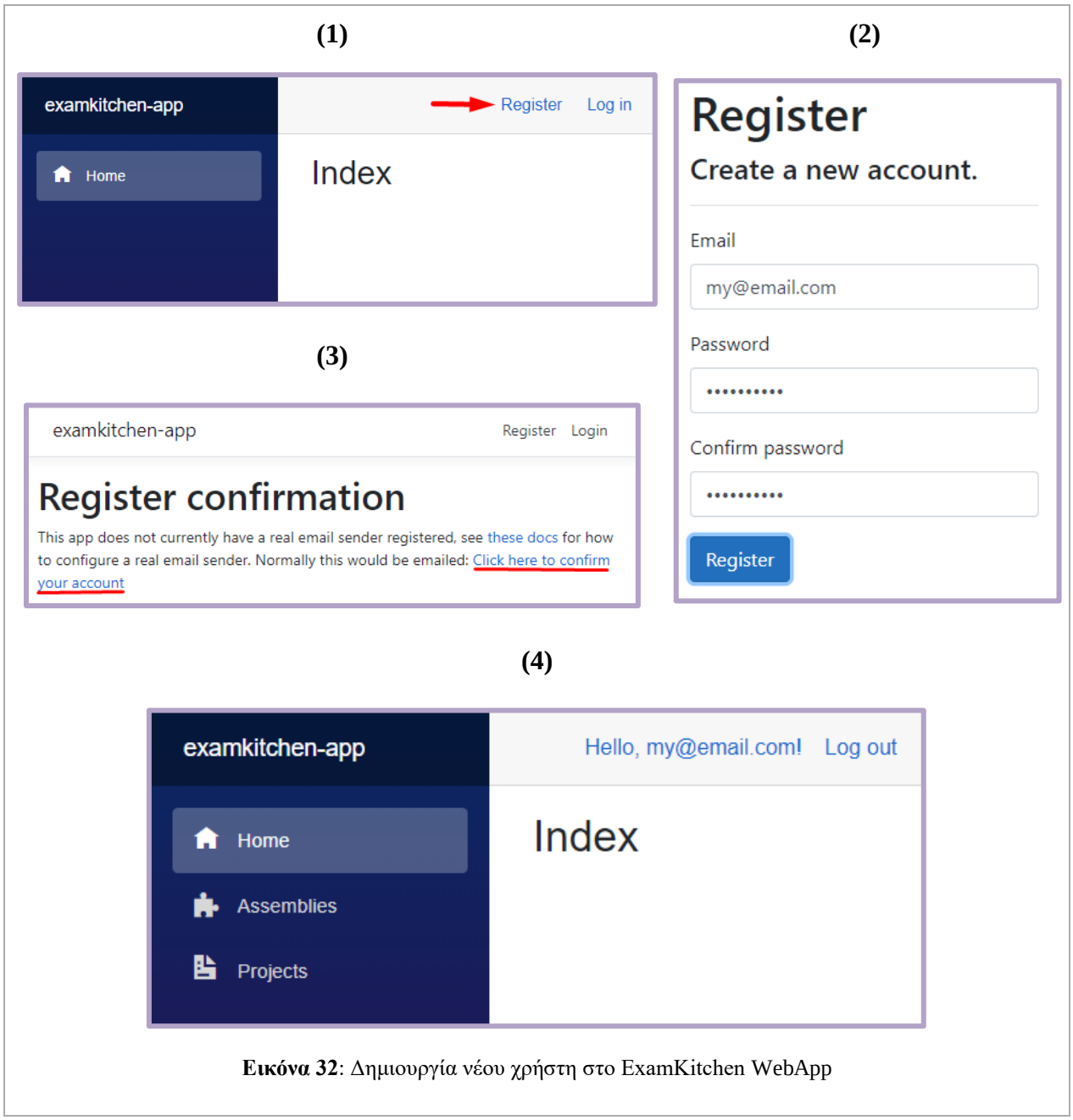
Εικόνα 30: Τελική δομή του Project



Εικόνα 31: Διαδικασία δημοσίευσης του module pack

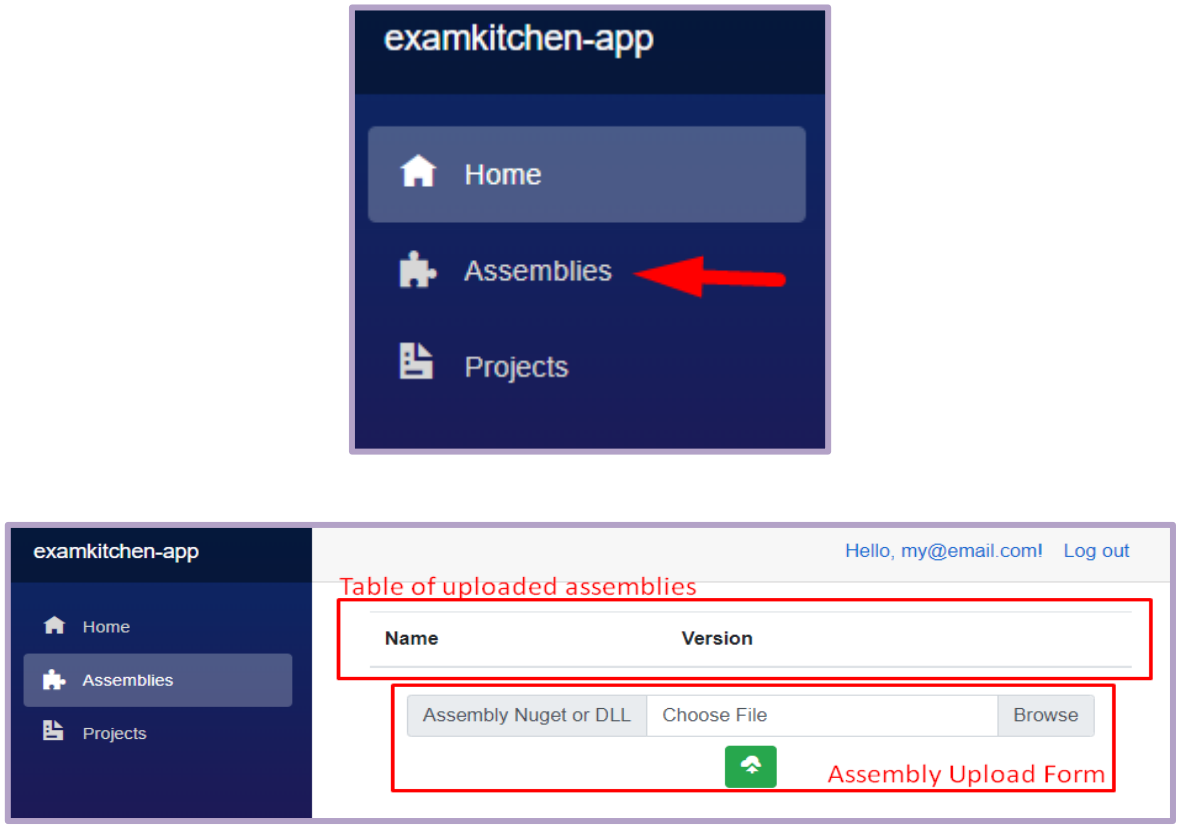
3.2. Χρήση του ExamKitchen WebApp

Όταν ολοκληρωθεί η δημιουργία ενός module pack, αυτό χρησιμοποιείται για το σχεδιασμό ενός project αξιολόγησης που βασίζεται σε ομάδες. Η αξιοποίηση των module packs του ExamKitchen προϋποθέτει τη gratis δημιουργία ενός λογαριασμού στο WebApp.



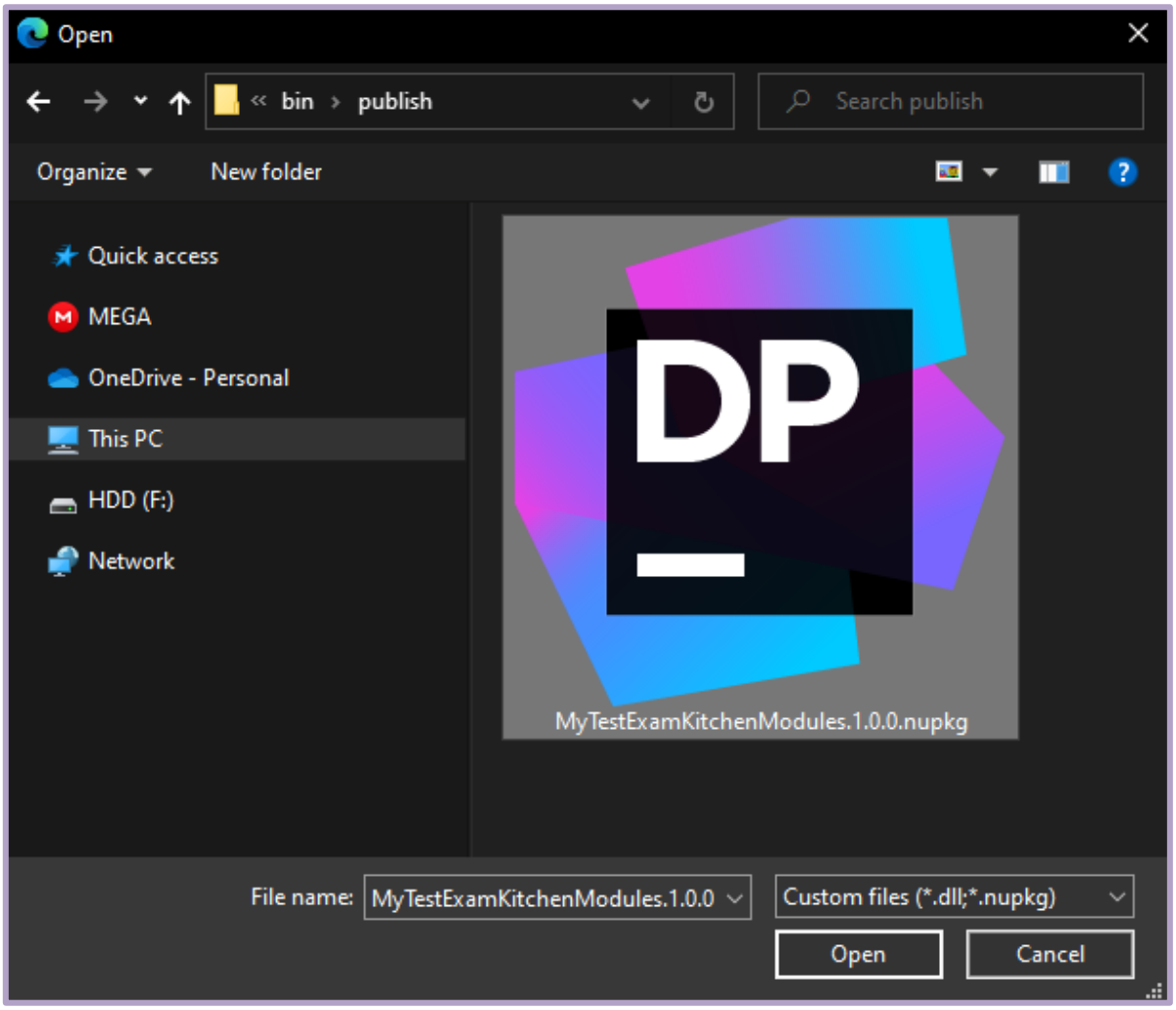
Εικόνα 32: Δημιουργία νέου χρήστη στο ExamKitchen WebApp

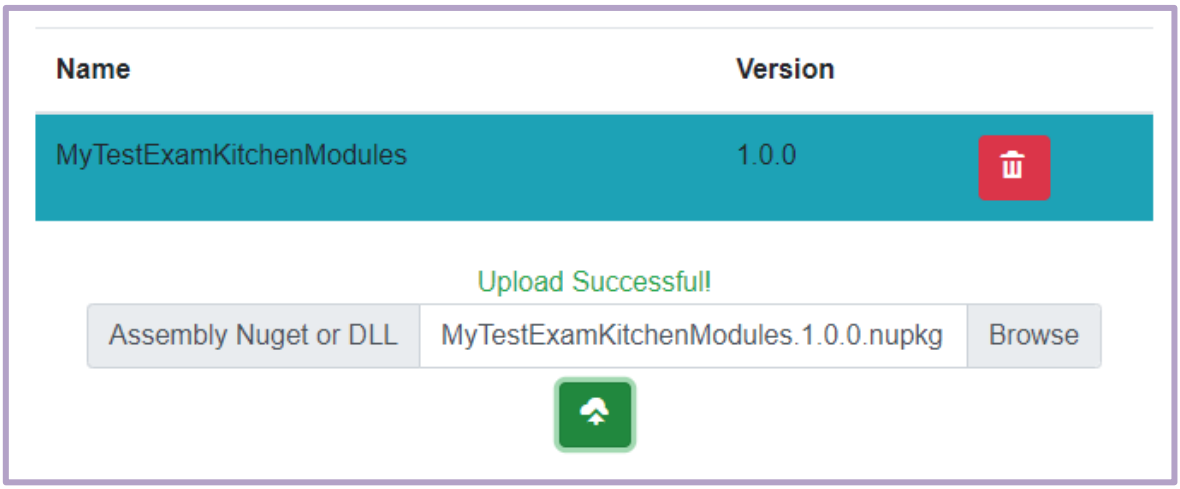
Αφού ο χρήστης συνδεθεί επιτυχώς, το Menu της εφαρμογής εμφανίζει δύο επιπλέον διαδρομές: Τα Assemblies και τα Projects. Αρχικά, ο χρήστης πρέπει να εισάγει στο WebApp το module pack που τον ενδιαφέρει. Κάθε χρήστης μπορεί να χρησιμοποιήσει μόνο τα module packs που έχει φορτώσει ο ίδιος στην εφαρμογή.



Εικόνα 33: Νέο Menu κατόπιν σύνδεσης του χρήστη και η αρχική οθόνη των Assemblies

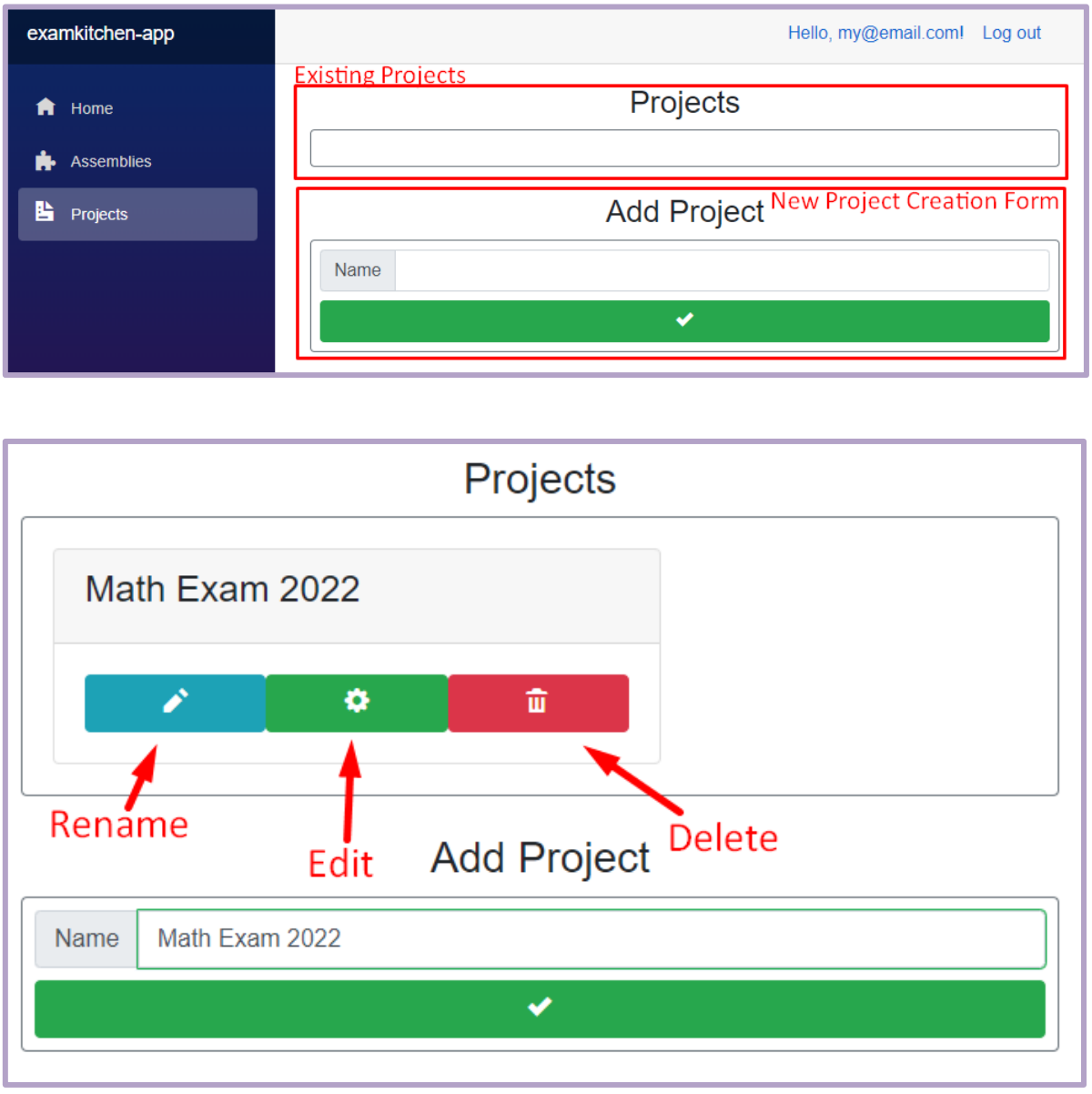
Στη σελίδα *Assemblies* εμφανίζεται ο πίνακας των ήδη ανεβασμένων Modules, και μία φόρμα φόρτωσης ενός καινούριου Assembly. Στην υπό μελέτη περίπτωση ο πίνακας είναι κενός, καθώς ο νεοσύστατος χρήστης δεν έχει φορτώσει ακόμα κάποιο module pack. Η φόρμα χρησιμοποιείται για το ανέβασμα του module pack.





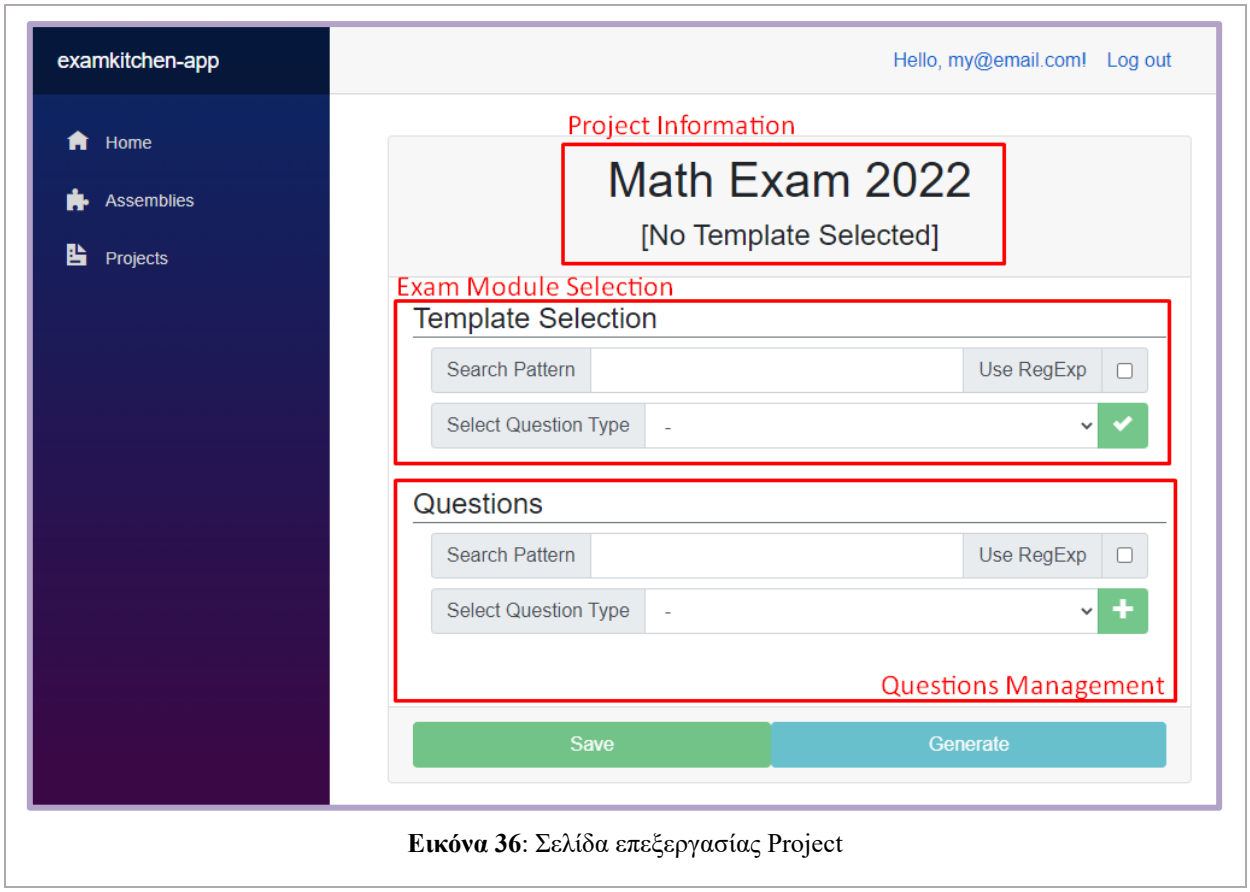
Εικόνα 34: Φόρτωση του νεοσύστατου module pack στο WebApp

Η πλοήγηση στην σελίδα *Projects* φέρνει στο προσκήνιο το panel διαχείρισης των Projects. Το panel συναποτελείται από τον πίνακα των υπαρχόντων Projects και την φόρμα δημιουργίας ενός νέου. Κατόπιν χρήσης της εν φόρμας, μία νέα καταχώρηση προστίθεται στον πίνακα και επιτρέπει *Μετονομασία*, *Επεξεργασία* και *Διαγραφή* της.



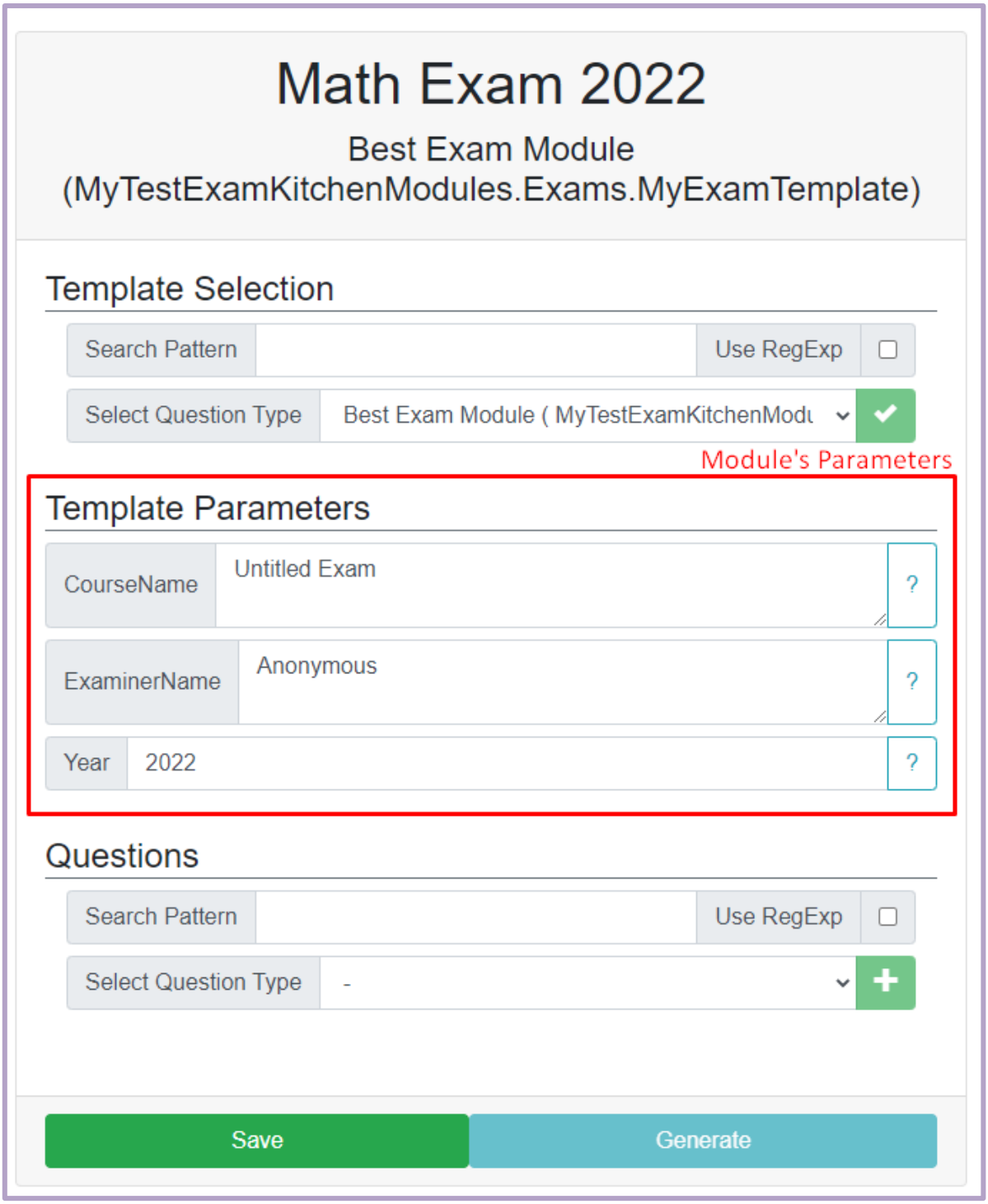
Εικόνα 35: Σελίδα Projects και δημιουργία ενός καινούριου Project

Επιλέγοντας την επεξεργασία του νεοσύστατου Project, εμφανίζεται η σελίδα *Projects/{ProjectId}*. Εδώ καταγράφονται πληροφορίες σχετικά με το Project, η δομή επιλογής Exam Module που το πλαισιώνει, καθώς και η υπεύθυνη για την διαχείριση των ερωτήσεων που αυτό περιλαμβάνει δομή.



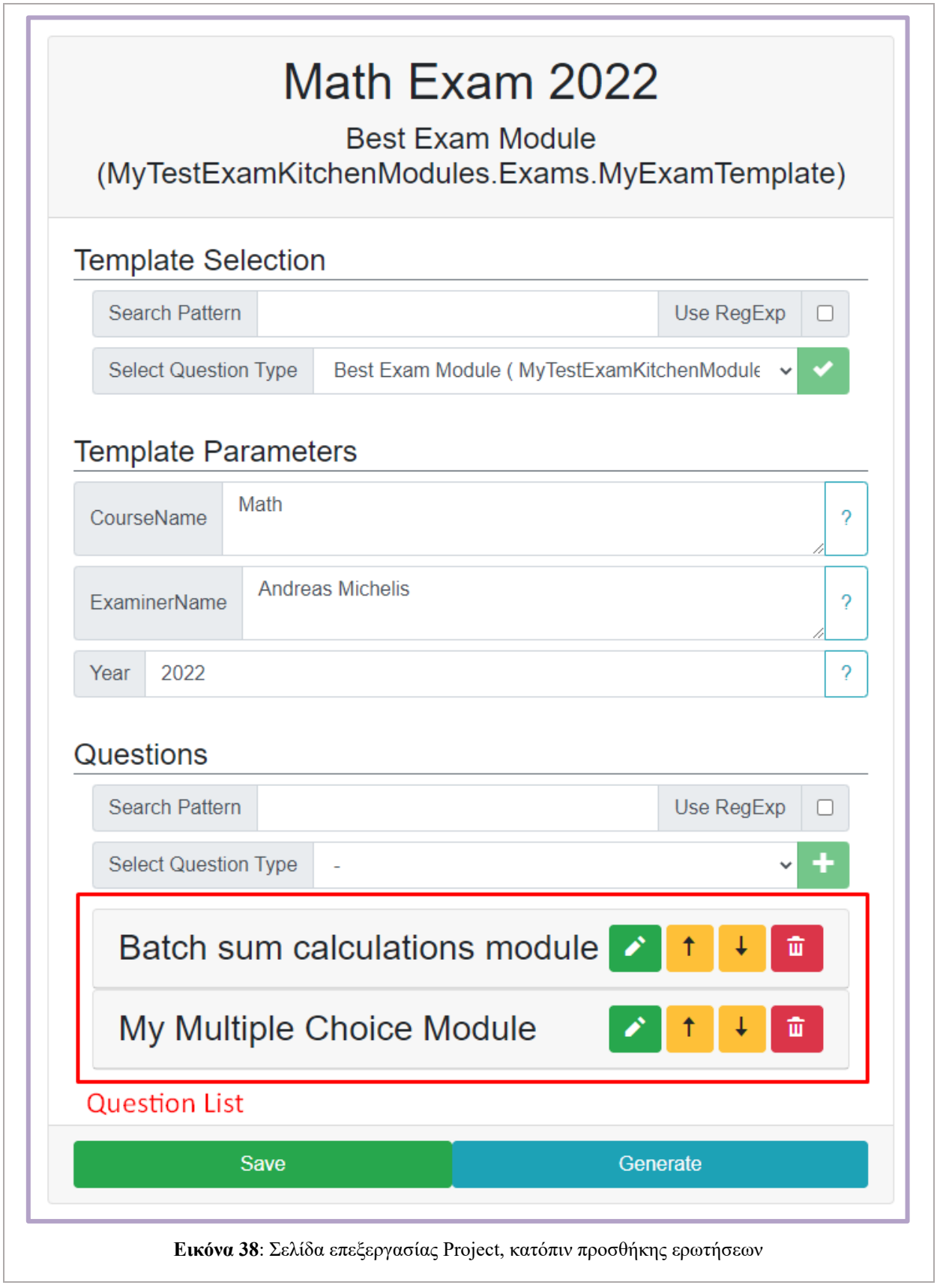
Εικόνα 36: Σελίδα επεξεργασίας Project

Όταν το *MyExamTemplate* επιλεχθεί ως το Exam Module του συγκεκριμένου project, παρουσιάζεται μια καινούρια δομή που περιλαμβάνει τις παραμέτρους του Module.



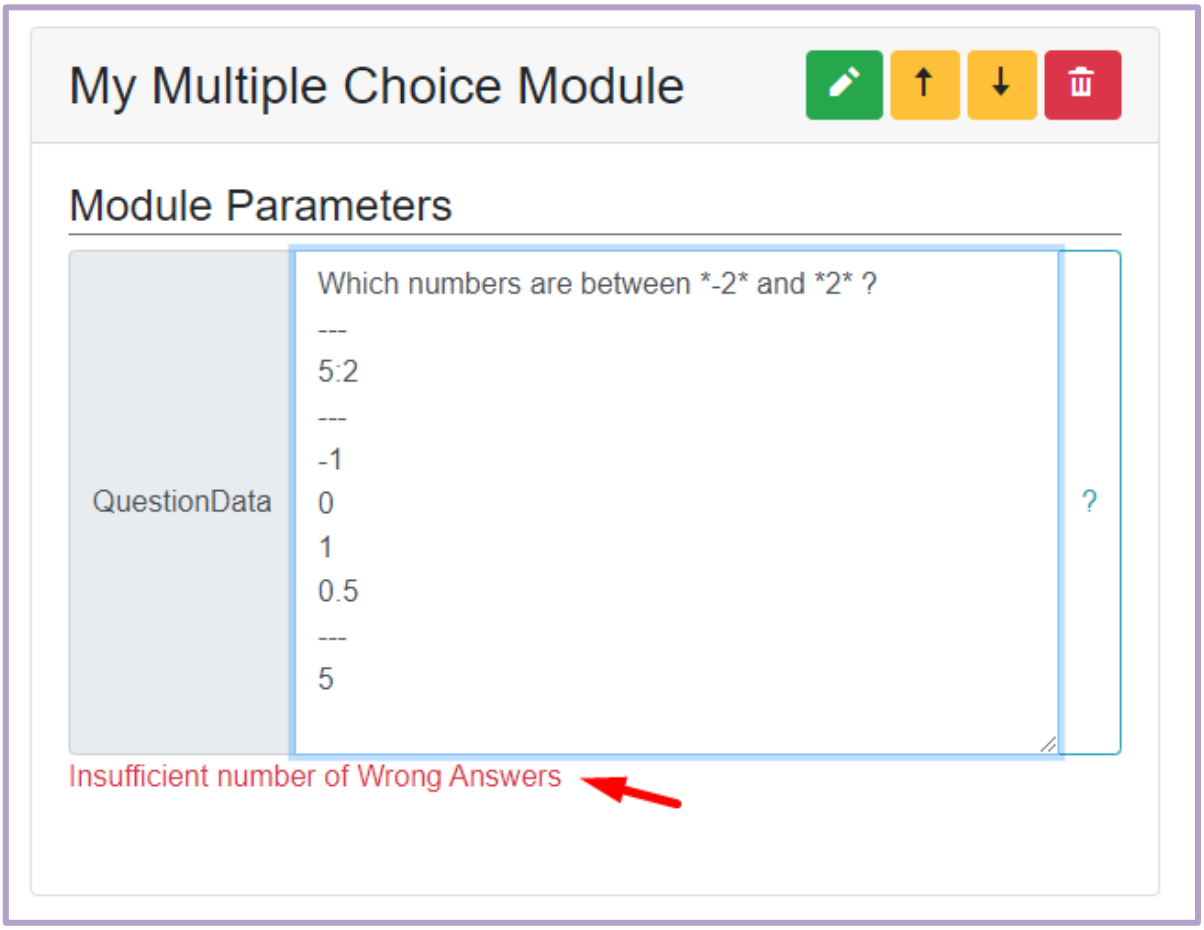
Εικόνα 37: Σελίδα επεξεργασίας Project μετά την επιλογή Exam Module

Ακολούθως, χρησιμοποιώντας τον επιλογέα Question Module που προσφέρεται από το IDE, προστέθηκαν δύο ερωτήσεις - μία για κάθε Question Module που δημιουργήθηκε.



Εικόνα 38: Σελίδα επεξεργασίας Project, κατόπιν προσθήκης ερωτήσεων

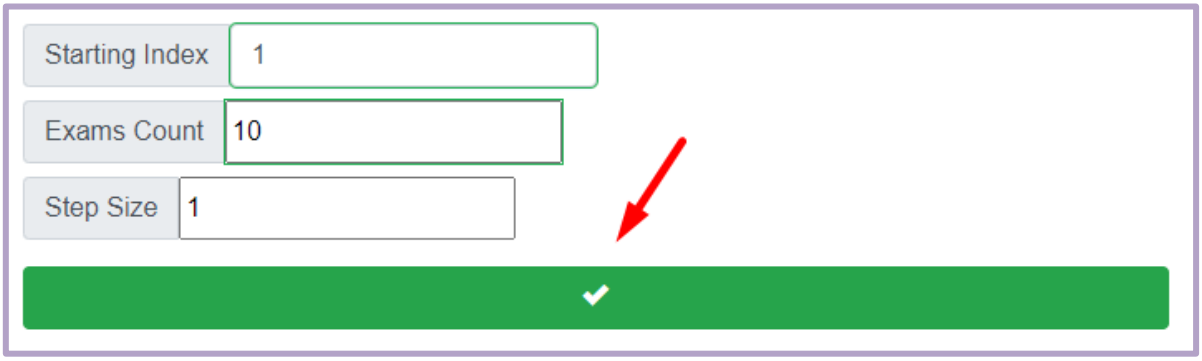
Χρησιμοποιώντας τα πλήκτρα επεξεργασίας των ερωτήσεων (), παραμετροποιείται κάθε ερώτηση. Αξίζει να σημειωθεί πως η ανατροφοδότηση που λαμβάνει ο χρήστης σχετικά με την ορθή σύνταξη των παραμέτρων που εισάγει ανανεώνεται με κάθε πλήκτρο που πατάει.



Εικόνα 39: Παράδειγμα ανάδρασης σε περίπτωση λάθους των παραμέτρων

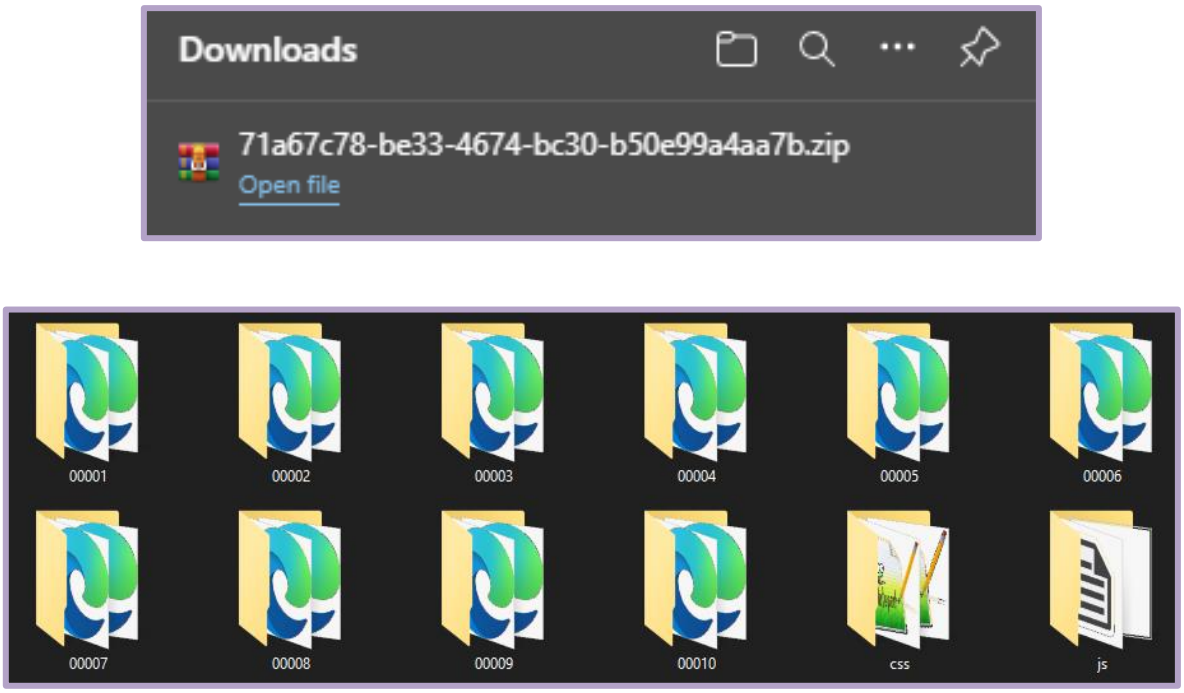
Την ολοκλήρωση της επεξεργασίας του Project ακολουθεί η αποθήκευσή του και, κατόπιν, η πλοήγηση στην σελίδα γένεσης θεμάτων. Η σελίδα */Projects/{ProjectId}/Print* είναι ένα λιτό περιβάλλον όπου ο χρήστης παραμετροποιεί τα ορίσματα της μεθόδου *Generate* - μέθοδος που βρίσκεται στην καρδιά κάθε Exam Module για να παράξει τα επιθυμητά θέματα.

Εικόνα 40: Αποθήκευση των αλλαγών του Project και η είσοδος στην σελίδα γένεσης θεμάτων



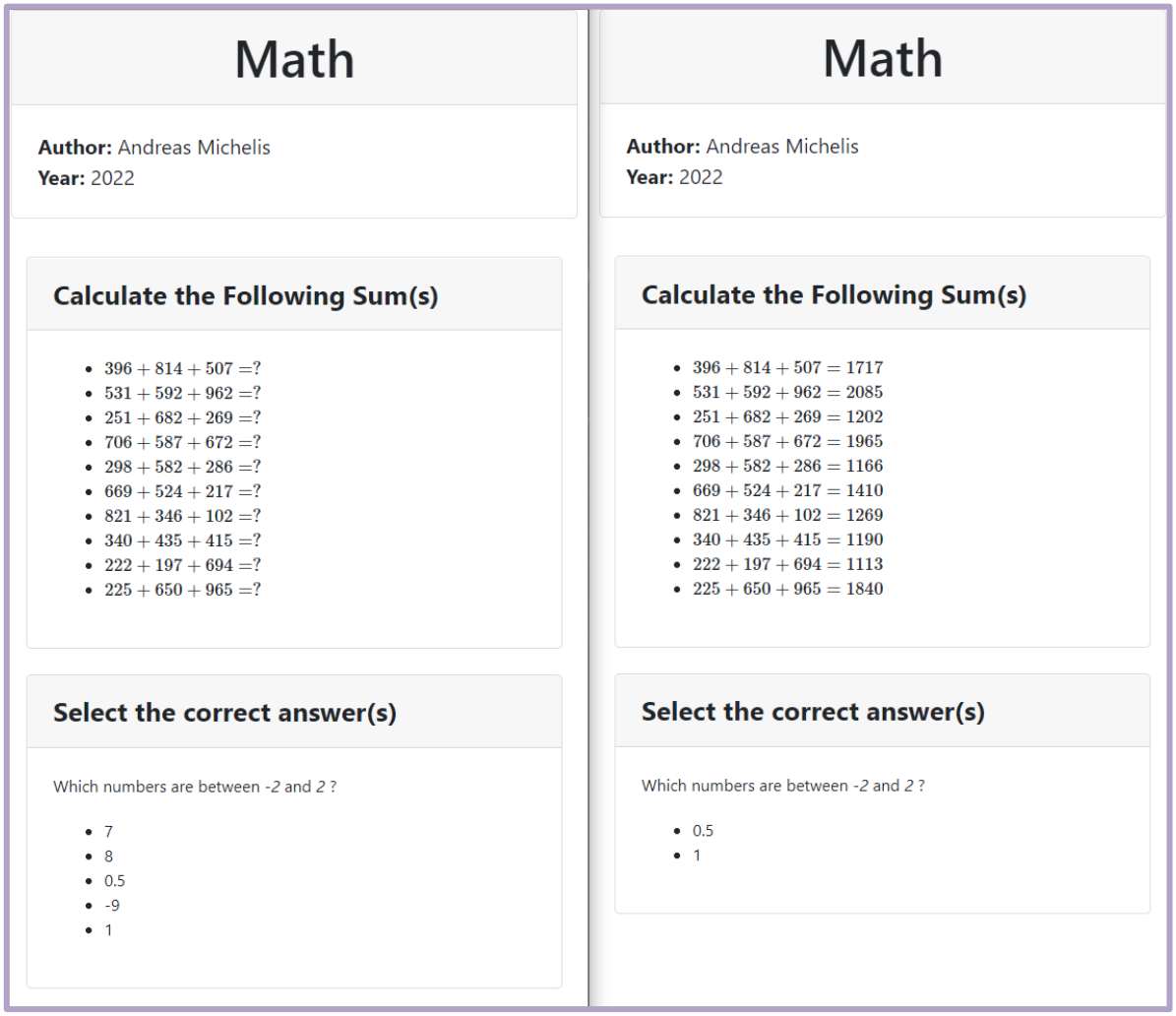
Εικόνα 41: Σελίδα Γένεσης Θεμάτων

Τροφοδοτώντας τα πεδία *Starting Index*, *Exams Count* και *Step Size* με τις κατάλληλες τιμές, η υποβολή του αιτήματος γένεσης επιφέρει την αυτόματη λήψη zip αρχείου. Κατόπιν αποσυμπίεσης, το αρχείο φανερώνει τους φακέλους js, css και πληθώρα φακέλων ονοματισμένων σύμφωνα με το group θεμάτων που διατηρούν.

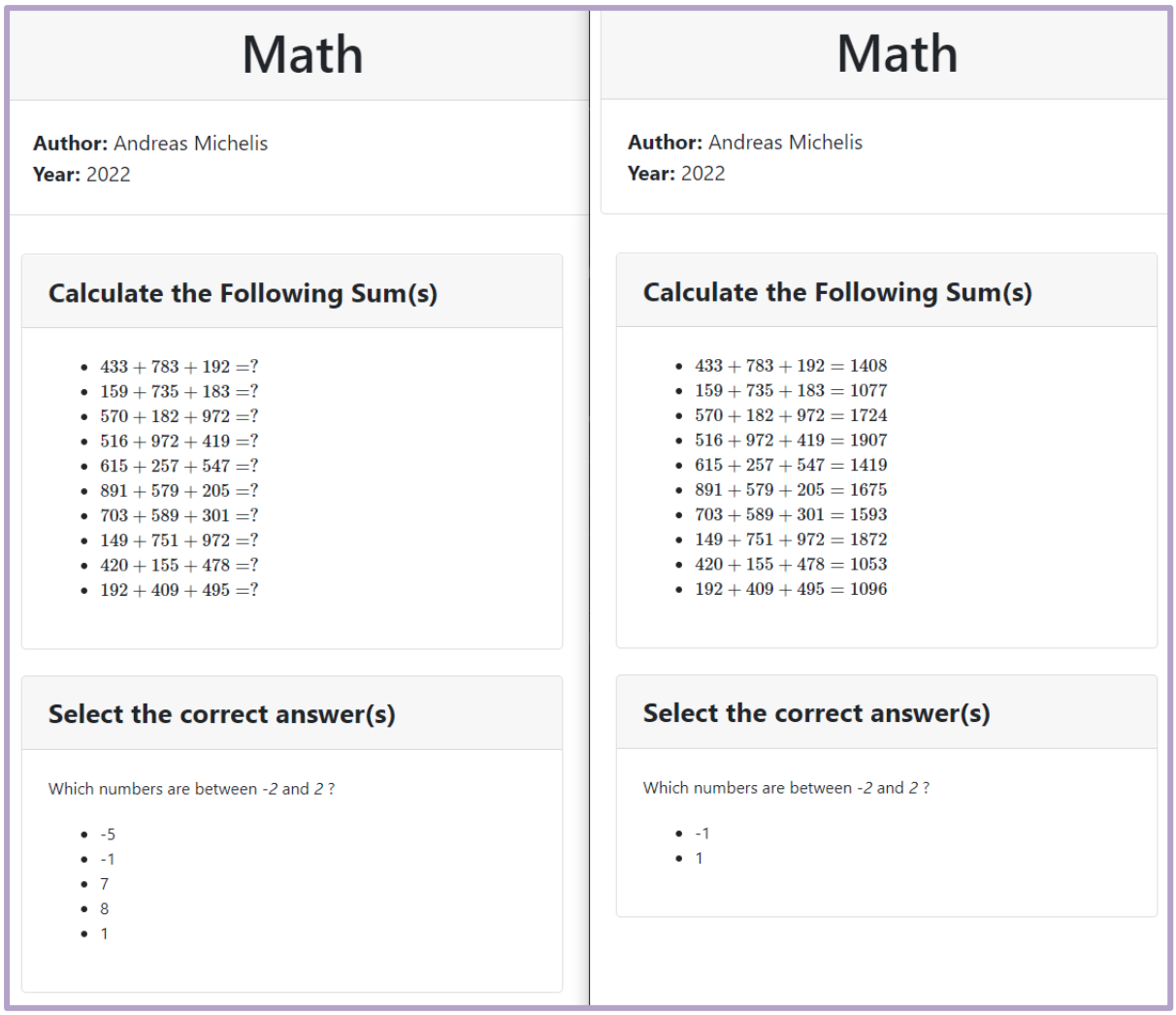


Εικόνα 42: Λήψη και αποσυμπίεση των νεόδμητων θεμάτων

Ανοίγοντας ένα από τα θέματα διαπιστώνεται πως η γένεση και, συνεπώς, ο σκοπός της αλληλεπίδρασης με το ExamKitchen WebApp ολοκληρώθηκε επιτυχώς.



Εικόνα 43: Αποτελέσματα γένεσης θεμάτων με την βοήθεια του ExamKitchen (Group 1)



Εικόνα 44: Αποτελέσματα γένεσης θεμάτων με την βοήθεια του ExamKitchen (Group 2)

Κεφάλαιο 4: Πορίσματα

Οι γεωμετρικοί ρυθμοί της τεχνολογικής ανάπτυξης και οι εφευρέσεις που αυτή συνεπάγεται αλλάζουν με ιλιγγιώδεις ρυθμούς τα δεδομένα στην εκπαίδευση. Η βαθμοθηρία που διαπνέει τον ακαδημαϊκό χώρο και η φετιχοποιηση του πτυχίου παραγκωνίζουν την αγάπη για μάθηση και ευνοούν δόλιες πρακτικές στο όνομα ενός τίτλου. Ελλείψει υποδομών οι εκπαιδευτικοί φορείς αδυνατούν να προσαρμοστούν στις συνεχείς αλλαγές και στερούνται μηχανισμούς αποτροπής ακαδημαϊκής απάτης. Εν γνώση μου πρωτότυπο, το παρόν πόνημα προδιαγράφει με σαφήνεια τις βάσεις για έναν μακροπρόθεσμο στόχο: την εγκαθίδρυση μίας ευέλικτης διαδικασίας αξιολόγησης που προσφέρει ευκολία σχεδιασμού, αμεροληψία και απαραβίαστη ακεραιότητα.

Η ύπαρξη δωρεάν υποδομών για τους κατοίκους ενός δήμου, έχει μικρή αξία για τους δημότες, όταν οι υποδομές δεν στελεχώνονται σωστά ή κατά τ' άλλα είναι παραμελημένες. Παρόμοια, η δωρεάν δημόσια εκπαίδευση μειώνεται ως προς τη συμβολή της, όταν στερείται ενός συστηματικού και ομοιογενούς τρόπου αξιολόγησης της ποιότητάς της. Η υλοποίηση ενός κοινού οικοσυστήματος για τον σχεδιασμό και τη δημιουργία εξετάσεων αποτέλεσε μία απαιτητική και, ως εκ τούτου, αδιαμφισβήτητα διδακτική εμπειρία. Παρότι το τελικό αποτέλεσμα στερείται κάποιων δυνατοτήτων και υιοθετεί αφελείς προγραμματιστικές τακτικές, όπως η φόρτωση assemblies κατευθείαν στο runtime του WebApp, η υλοποίησή του στηρίχθηκε στην διερεύνηση ενός βαθιά ετερογενούς φάσματος γνωστικών πεδίων. Συγκεράζοντας ένα framework που αφορά τον προγραμματιστή, με μία διαδικτυακή εφαρμογή που προορίζεται για χρήση από εξεταστικούς φορείς δίχως γνώσεις προγραμματισμού, το ExamKitchen φιλοδοξεί να χαράξει μια οδό συνεργασίας προγραμματιστών και εκπαιδευτικών ιδρυμάτων για την ανάπτυξη ενός καλύτερου εκπαιδευτικού συστήματος.