ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μάθημα: «ΕΚΠΑΙΔΕΥΤΙΚΟ ΛΟΓΙΣΜΙΚΟ»

Εξάμηνο: 8^{Ο}

Ημερομηνία: 05/07/2021

Καθηγητής: Βίρβου Μαρία,

Σακκόπουλος Ευάγγελος

Φοιτητές: Μπαντάνα Δανάη-Ιωάννα, Π17081

Σπυρόπουλος Γιώργος, Π17127

Θέμα: «Εκπαιδευτικό παιχνίδι για την εκμάθηση της προπαίδειας των Μαθηματικών»

ΤΕΧΝΙΚΟ ΕΓΧΕΙΡΙΔΙΟ

Περιεχόμενα

1. Εισαγωγή 3
   1.1. Ορισμός του προβλήματος προς επίλυση 3
   1.2. Στόχοι της εργασίας 3
2. Έναρξη (Inception) 5
   2.1. Σύλληψη απαιτήσεων 5
   2.2. Ανάλυση-Σχεδιασμός 5
   2.3. Αρχικά Διαγράμματα 6
      2.3.1. Διάγραμμα Περιπτώσεων Χρήσης 6
      2.3.2. Διάγραμμα Τάξεων 7
3. Τελικά Διαγράμματα 8
   3.1. Διάγραμμα Περιπτώσεων Χρήσης 8
   3.2. Διάγραμμα Τάξεων 9
4. Κλάσεις 9
   4.1. Models 9
   4.2. Storage 10
   4.3. Services 10
5. Φόρμες 11
   5.1. LoginForm 11
   5.2. RegisterForm 12
   5.3. MainForm 12
   5.4. TheoryForm 17
   5.5. TestForm 19
   5.6. ProfileForm 23
   5.7. HelpForm 25
   5.8. Γενικά Χαρακτηριστικά 26
      5.8.1. Κουμπί «Πίσω» 26
      5.8.2. Κουμπί «Βοήθεια» 26
      5.8.3. Κουμπί «Πληροφοριών/Βοήθειας» 26

1. Εισαγωγή

Το εκπαιδευτικό παιχνίδι για την εκμάθηση της προπαίδειας των Μαθηματικών «Mathster» αποτελεί μία desktop εφαρμογή που αναπτύχθηκε στο «Visual Studio 2019» με χρήση της «C#» γλώσσας προγραμματισμού. Η αποθήκευση δεδομένων και η εκτέλεση ερωτημάτων σε αυτά πραγματοποιήθηκε σε βάση δεδομένων «SQLite».

1.1. Ορισμός του προβλήματος προς επίλυση

Σήμερα, όσο ποτέ άλλοτε, για την εκμάθηση και την ενίσχυση γνώσεων, παρατηρείται η ένταξη της χρήσης τεχνολογικών μέσων στην εκπαιδευτική διαδικασία. Με την αξιοποίηση των κατάλληλων λογισμικών, οι χρήστες έχουν την δυνατότητα να αντιλαμβάνονται και να κατανοούν καλύτερα το θέμα που διδάσκονται, ευκολότερα και γρηγορότερα, εξοικονομώντας χρόνο.

Ο μαθητής χρειάζεται ένα υποστηρικτικό λογισμικό με δυνατότητα πρόσβασης σε αυτό όποτε επιθυμεί. Έτσι, η δημιουργία μιας desktop εφαρμογής που θα τον υποστηρίζει κρίνεται απαραίτητη.

Συνεπώς, το πρόβλημα που έχουμε να αντιμετωπίσουμε είναι πώς θα γίνει η ανάπτυξη ενός καλού, αξιόπιστου και εύχρηστου προγράμματος που θα χρησιμοποιεί ο μαθητής, ανεξάρτητα αν έχει ή όχι γνώσεις υπολογιστών. Πρέπει οι επιλογές που θα έχει να είναι ευδιάκριτες, στις φόρμες που θα πρέπει να συμπληρώσει να υπάρχουν σχόλια για να ξέρει τι πρέπει να συμπληρώσει και που, καθώς και να υπάρχουν έλεγχοι της σωστής συμπλήρωσης των πεδίων, ώστε να μην γίνονται λάθη.

Έτσι, η δημιουργία του λογισμικού θα βασιστεί στην κάλυψη των προαναφερόμενων αναγκών.

1.2. Στόχοι της εργασίας

Στο πλαίσιο του μαθήματος «Εκπαιδευτικό Λογισμικό» θα αναπτυχθεί μία desktop εφαρμογή εκμάθησης της προπαίδειας των Μαθηματικών με τη χρήση αντικειμενοστραφούς σχεδίασης τεκμηριωμένης με τη γλώσσα περιγραφής UML. Πιο συγκεκριμένα, θα αναπτυχθεί ένα αλληλεπιδραστικό παιχνίδι εκμάθησης της

προπαίδειας, το οποίο θα απευθύνεται σε παιδιά ηλικίας 6 έως 10 ετών για την καλύτερη κατανόηση και την πιο διασκεδαστική εκμάθηση της προπαίδειας.

Βασικός σκοπός της εφαρμογής είναι ο χρήστης, μέσω του καλού σχεδιασμού και της υλοποίησης του εκπαιδευτικού λογισμικού (διδασκαλία, ασκήσεις και αξιολόγηση του μαθητή), να μάθει, να κατανοήσει το θέμα και να μπορεί να απομνημονεύσει και εμπεδώσει την ύλη. Έτσι, ο μαθητής θα μπορεί εύκολα, διασκεδαστικά και αξιόπιστα να ενισχύσει τις γνώσεις του στη προπαίδεια των μαθηματικών.

Θα υπάρχει δυνατότητα εγγραφής και σύνδεσης του χρήστη στην εφαρμογή. Στην συνέχεια θα μπορεί να ξεκλειδώνει τις ενότητες μία μία, αφού έχει ολοκληρώσει το διάβασμα της θεωρίας της εκάστοτε ενότητας και έχει ανταποκριθεί επιτυχώς στο τεστ αυτοαξιολόγησης της συγκεκριμένης ενότητας. Ακόμη, ανά τρεις ενότητες, ο χρήστης θα υποβάλλεται σε ένα επαναληπτικό τεστ που θα περιέχει ερωτήσεις των προς εξέταση προηγούμενων κεφαλαίων, καθώς και θα έχει την επιλογή να κάνει επαναληπτικά τεστ που θα συμπεριλαμβάνουν ερωτήσεις όλων των ολοκληρωμένων κεφαλαίων. Τα τεστ θα περιλαμβάνουν ένα σύνολο μαθηματικών ερωτήσεων σε μορφή πολλαπλής επιλογής, σωστού – λάθους και συμπλήρωσης κενών.

Παράλληλα, εκτός της βασικής λειτουργίας εκμάθησης της προπαίδειας και χρήσης τεστ για την εκτίμηση της γνώσης του μαθητή, το λογισμικό θα προσφέρει και άλλες δυνατότητες με σκοπό την πλήρη αξιολόγηση του. Τέτοιες δυνατότητες είναι η παρουσίαση του εξατομικευμένου προφίλ του χρήστη που θα εκθέτει την πρόοδο του. Η απόδοση του θα βασίζεται στην επίδοσή του μαθητή στα τεστ αυτοαξιολόγησης και στο πόσες φορές διαβάζει τη θεωρία της κάθε ενότητας, καθώς και στα επαναληπτικά τεστ.

Επιπλέον, θα πρέπει να εντοπίζονται λάθη του μαθητή που αφορούν την ελλιπή γνώση του σε μία ενότητα. Αν κάτι τέτοιο γίνει αντιληπτό, θα πρέπει να εμφανιστεί η θεωρία και να υποβληθεί ο χρήστης σε ένα τεστ αυτοαξιολόγησης στο οποίο θα πρέπει να επιτύχει για να προχωρήσει παρακάτω. Η αδυναμία αυτή θα καταγράφεται στη πρόοδο του και με την επιτυχή ολοκλήρωση της παραπάνω διαδικασίας θα αφαιρείται.

Στόχος του λογισμικού είναι να εξιδανικεύσει την διαδικασία εκμάθησης της προπαίδειας των μαθηματικών και να την κάνει πιο εύκολη και διασκεδαστική για τον ενδιαφερόμενο χρήστη.

2. Έναρξη (Inception)

2.1. Σύλληψη απαιτήσεων

Στη φάση αυτή γίνεται ο καθορισμός της προοπτικής του έργου. Απαραίτητη προϋπόθεση για να καθοριστεί η λειτουργία του λογισμικού θα αποτελέσει η κάλυψη των αναγκών του χρήστη αλλά και η ευκολία στη χρήση και στη διαχείριση της εφαρμογής.

Σε αυτό το στάδιο καλούμαστε να απαντήσουμε στις εξής ερωτήσεις:

1. Ποιοι θα είναι οι χρήστες της εφαρμογής και τι θα κάνει το σύστημα για αυτούς;
2. Ποια είναι μια αρχική αρχιτεκτονική του συστήματος;
3. Ποιο είναι το πλάνο, ποιες οι απαιτήσεις και ποιοι οι κίνδυνοι;

Για να απαντήσουμε στα παραπάνω ερωτήματα ποιο αναλυτικά θα μελετήσουμε τα παρακάτω διαγράμματα.

2.2. Ανάλυση-Σχεδιασμός

Διενεργήθηκε έρευνα σχετικά με τις ανάγκες του χρήστη για την εφαρμογή «Mathster». Οι πληροφορίες που αποκομήθηκαν οδήγησαν στον καθορισμό της δομής που θα έχει η εφαρμογή, σε ποιες ομάδες χρηστών απευθύνεται, καθώς και ποιες λειτουργίες θα μπορεί να πραγματοποιήσει ο χρήστης.

Οι χρήστες της εφαρμογής θα είναι παιδιά ηλικίας 6 έως 10 ετών που διαθέτουν ηλεκτρονικό υπολογιστή και έχουν ανάγκη ή επιθυμούν να ενισχύσουν και να βελτιώσουν τις γνώσεις τους πάνω στη προπαίδεια των μαθηματικών. Το πρόγραμμα θα χρησιμοποιείται ως υποστήριξη των προσωπικών αλλά και σχολικών τους αναγκών.

Οι ενέργειες που θα μπορούν να πραγματοποιηθούν μέσα από την εφαρμογή θα είναι οι εξής:

- Ο χρήστης θα μπορεί να κάνει εγγραφή στην εφαρμογή, φτιάχνοντας έτσι ένα προσωπικό λογαριασμό.
- Ο χρήστης θα μπορεί να κάνει είσοδο στην εφαρμογή, εισάγωντας το όνομα χρήστη και τον κωδικό που επέλεξε κατά την εγγραφή του στην εφαρμογή.

- Ο χρήστης θα μπορεί να δει όλες τις ενότητες του διδακτικού υλικού αλλά θα μπορεί να επιλέξει μόνο αυτές που έχει ξεκλιδώσει, ξεκινώντας από την ενότητα παρουσίασης και εξέτασης της προπαίδειας του «1».
- Ο χρήστης επιλέγοντας μία ενότητα, θα μπορεί να μελετήσει τη θεωρία και να προσπαθήσει το τεστ αυτοαξιολόγησης της ενότητας, με σκοπό να ξεκλειδώσει το επόμενο στάδιο.
- Ο χρήστης όποτε επιθυμεί, προκειμένου να ενισχύσει τις γνώσεις του, θα μπορεί να κάνει ένα επαναληπτικό τεστ που θα βασίζεται σε γνώσεις των ενοτήτων που έχει ολοκληρώσει.
- Ο χρήστης θα μπορεί να δει τις πληροφορίες του λογαριασμού του καθώς και την γενική και ανά κεφάλαιο απόδοσή του.
- Ο χρήστης σε περίπτωση συνεχόμενων λαθών σε μία ενότητα θα υποβάλλεται να κάνει επανάληψη τη θεωρία και ένα τεστ αυτοαξιολόγησης της ενότητας.
- Ο χρήστης θα μπορεί να κάνει αποσύνδεση από τον λογαριασμό του.

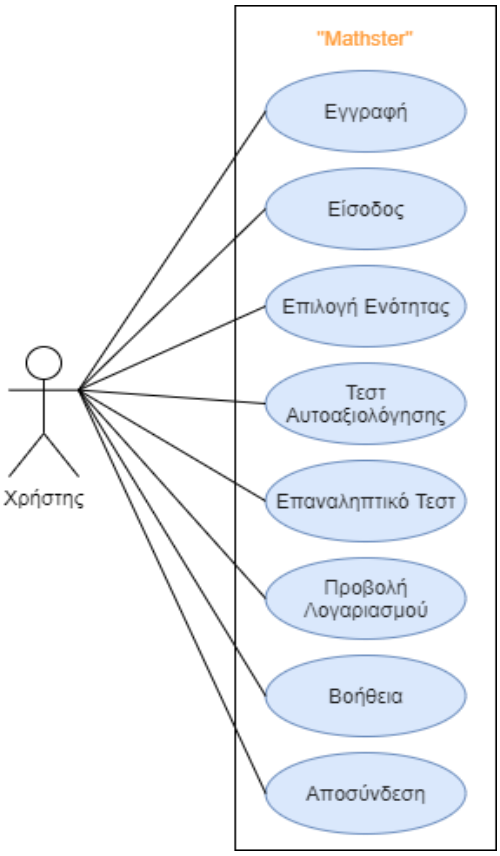
Το Λογισμικό εκτός των παραπάνων:
- Θα πρέπει να ενημερώνει τον χρήστη σε ποιές ενότητες αντιμετωπίζει πρόβλημα.

2.3. Αρχικά Διαγράμματα

2.3.1. Διάγραμμα Περιπτώσεων Χρήσης

Τα διαγράμματα περιπτώσεων χρήσης περιγράφουν την συμπεριφορά ενός συστήματος και αναπαριστούν τις λειτουργίες ενός συστήματος από την οπτική γωνία του χρήστη. Είναι μία εικόνα της λειτουργικότητας ενός συστήματος το οποίο ενεργοποιείται για να ανταποκριθεί σε έναν εξωτερικό ενεργοποιό (actor), ο οποίος στην συγκεκριμένη εφαρμογή είναι οι μαθητές. Μία περίπτωση χρήσης περιέχει μία ροή ενεργειών που αποτελεί την περιγραφή των γεγονότων που χρειάζονται για να επιτύχουν τη λειτουργικότητα της. Δείχνει τι πρέπει να κάνει το σύστημα και όχι το πώς θα το κάνει.

Στο παρακάτω διάγραμμα φαίνονται οι ενέργειες που μπορεί να πραγματοποιήσει ο χρήστης κατά την είσοδό του στην εφαρμογή. Έχουμε τον actor «Χρήστη»(“User”) που είναι ο χειριστής της εφαρμογής.

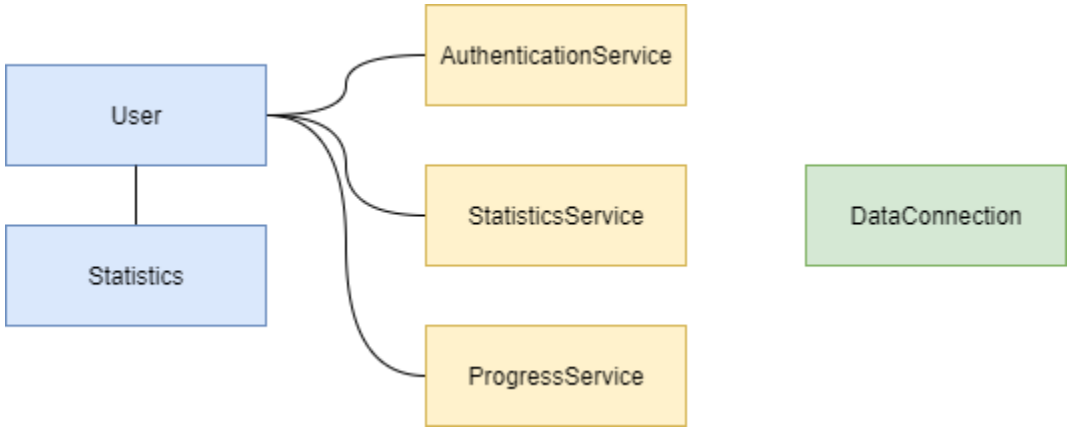


Η πρώτη ενέργεια του χρήστη είναι να κάνει «Είσοδο/Login» στην εφαρμογή μόλις αυτή ανοίξει ή στην περίπτωση που δεν έχει λογαριασμό να κάνει «Εγγραφή/Register».
Στη συνέχεια, μπορεί να επιλέξει μία από τις ενότητες που έχει ξεκλειδώσει, να διαβάσει τη θεωρία και να κάνει ένα τεστ αυτοαξιολόγησης. Ο χρήστης μπορεί να επιλέξει να κάνει κατευθείαν ένα τεστ αυτοαξιολόγησης μίας ενότητας αλλά και επαναληπτικά τεστ των ενοτήτων που έχει ξεκλειδώσει.
Ακόμη, μπορεί να δει πληροφορίες του λογαριασμού του, δηλαδή τα προσωπικά και τα στατιστικά στοιχεία της απόδοσής του.
Επιπλέον, σε κάθε φόρμα μπορεί να ζητήσει βοήθεια ώστε να δει τις οδηγίες χρήσης της εφαρμογής αλλά και να δει αναλυτικά πληροφορίες για κάθε σελίδα της εφαρμογής.
Ως τελευταία λειτουργία μπορέι να αποσυνδεθεί από την εφαρμογή.

Τέλος, έχουμε τις σχέσεις μεταξύ των στοιχείων των διαγραμμάτων περιπτώσεων χρήσης. Εξ ορισμού οι «Actors» πρέπει να αλληλοεπιδράσουν με όλα ή κάποια από τα στοιχεία του «Use Case» ώστε να επιτύχουν έναν στόχο. Στο διάγραμμα παρουσιάζονται με γραμμές ποιες λειτουργίες μπορούν να πραγματοποιηθούν από ποιόν,δείχνοντας έτσι την κάθε συσχέτιση («Association»).

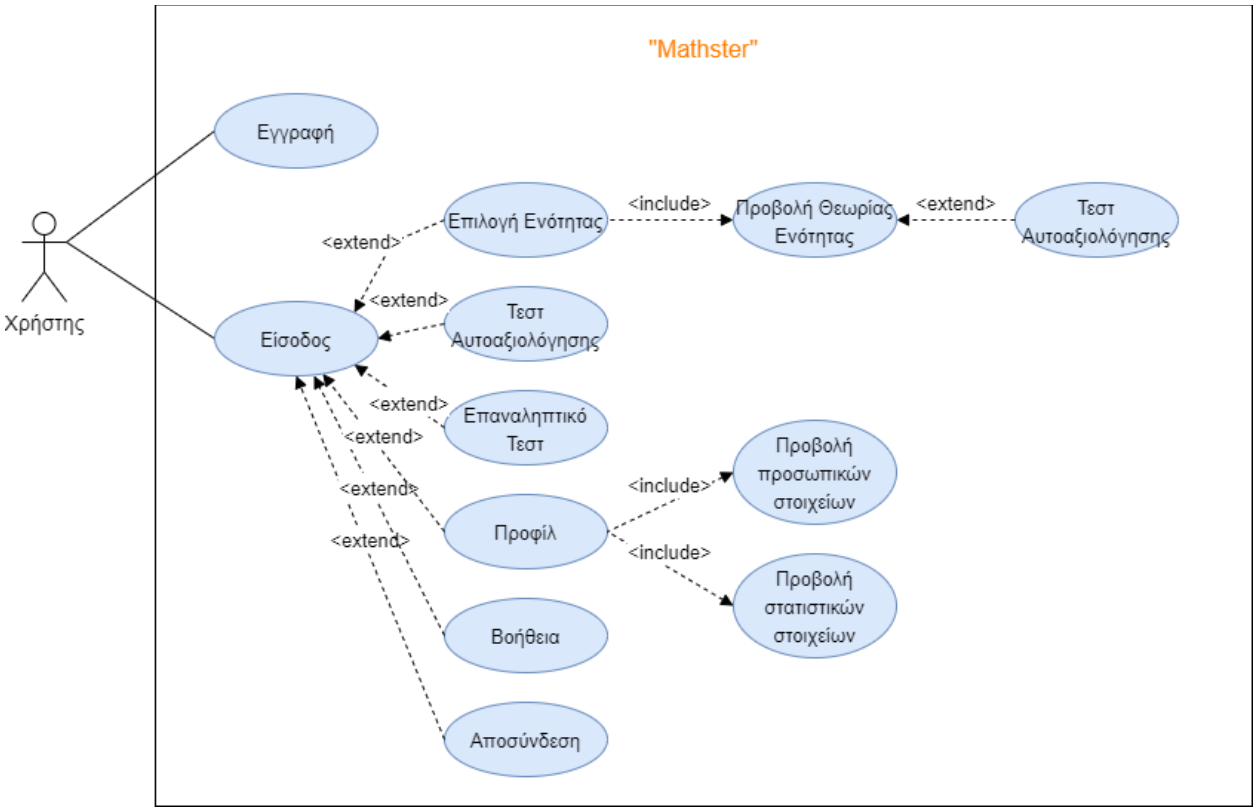
2.3.2. Διάγραμμα Τάξεων

Αναπαριστούν τη στατική δομή της εφαρμογής όσον αφορά τις τάξεις και τις σχέσεις τους. Οι τάξεις αποτελούν το βασικό στοιχείο για την κατασκευή ενός αντικειμενοστραφούς συστήματος.
Στην desktop εφαρμογή «Mathster», με βάση τις απαιτήσεις που έχουμε συλλέξει θα χρησιμοποιήσουμε έξι κλάσεις· την «User» για την περιγραφή των χρηστών της εφαρμογής, την «Statistics» για την αναπαράσταση των στατιστικών μίας ενότητας για έναν συγκεκριμένο χρήστη, την «AuthenticationService» για τη λειτουργία ως σημείο αναφοράς του συνδεδεμένου χρήστη, την «StatisticsService» για την σύλληψη στατιστικών του χρήστη για μία ενότητα, την «ProgressService» για τη λήψη πληροφοριών και στατιστικών που αφορούν την πρόοδο του χρήστη και τέλος την «DataConnection» για τη επικοινωνία με τη τοπική βάση δεδομένων “SQLite”.



3. Τελικά Διαγράμματα

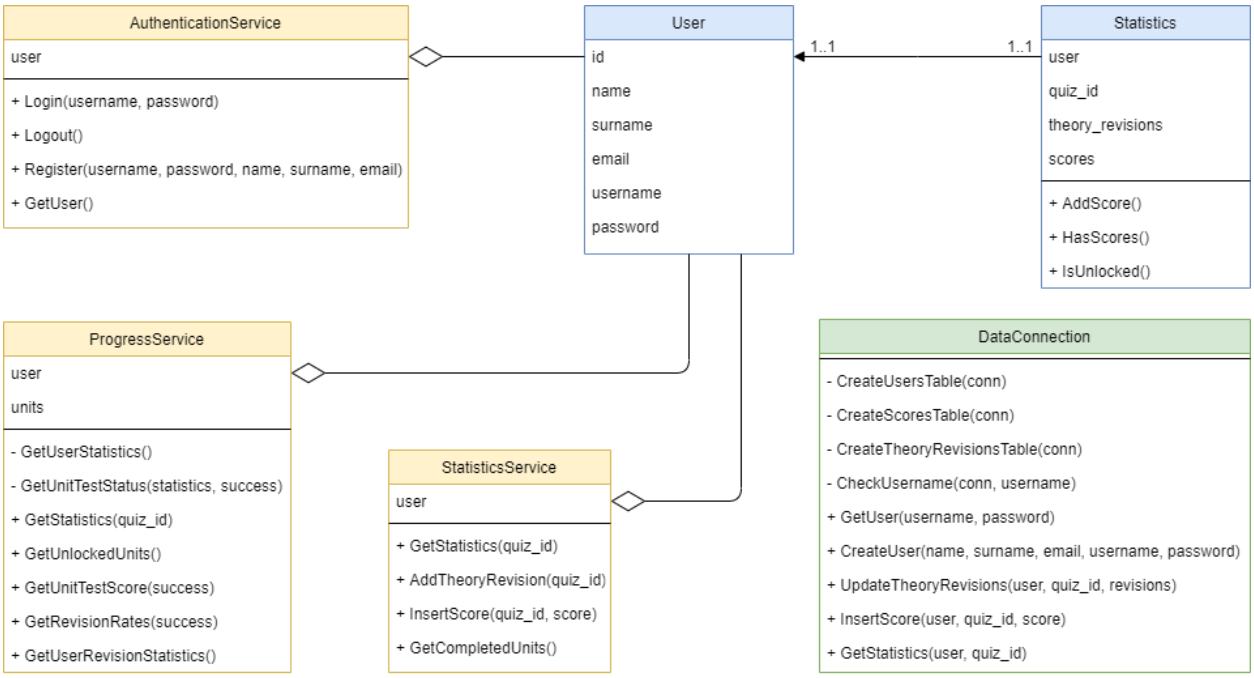
3.1. Διάγραμμα Περιπτώσεων Χρήσης



Στο παραπάνω διάγραμμα φαίνονται οι ενέργειες και η ροή αυτών των ενεργειών που μπορεί να πραγματοποιήσει ο χρήστης κατά την είσοδό του στην εφαρμογή.

3.2. Διάγραμμα Τάξεων

Στο διάγραμμα αυτό παρουσιάζονται αναλυτικά τα χαρακτηριστικά και οι συναρτήσεις της κάθε κλάσης καθώς και οι σχέσεις μεταξύ τους. Φαίνεται ότι οι κλάσεις «AuthenticationService», «ProgressService» και «StatisticsService» χρησιμοποιούν και εμπεριέχουν, μεταξύ άλλων, ένα αντικείμενο τύπου «User». Κάθε αντικείμενο της τάξης «Statistics» θα στηρίζεται σε ένα αντικείμενο τύπου «User».



4. Κλάσεις

4.1. Models

Τα μοντέλα αποτελούν τις κλάσεις για την αναπαράσταση των οντοτήτων της εφαρμογής ώστε να διευκολύνουν την διαχείρισή τους.

Συγκεκριμένα, η κλάση «**User**» αναπαριστά τους χρήστες της εφαρμογής (μαθητές) και διευκολύνει την πρόσβαση στα στοιχεία του συνδεδεμένου χρήστη όπως ονοματεπώνυμο, email, όνομα χρήστη και κωδικό.

Αντίστοιχα, η κλάση «**Statistics**» αναπαριστά τα στοιχεία μίας ενότητας για έναν συγκεκριμένο χρήστη, διευκολύνοντας την διαχείριση χαρακτηριστικών όπως τον αριθμό των επαναλήψεων της θεωρίας καθώς και τις βαθμολογίες του χρήστη στα αντίστοιχα τεστ αυτοαξιολόγησης της ενότητας.

4.2. Storage

Στην κατηγορία της αποθήκευσης ανήκει η κλάση «**DataConnection**» η οποία αποτελεί μια διεπαφή υψηλού επιπέδου για τις λειτουργίες εγγραφής, ανάγνωσης και ανανέωσης στην τοπική βάση δεδομένων SQLite. Σκοπός της είναι η αφαίρεση των στοιχείων χαμηλού επιπέδου της βάσης δεδομένων από το επίπεδο λογικής της εφαρμογής. Έτσι, επιτυγχάνεται η επαναχρησιμοποίηση των διαδικασιών αποθήκευσης ενώ παράλληλα δίνεται η δυνατότητα αλλαγής της υποδομής αποθήκευσης στο μέλλον χωρίς να επηρεαστεί η λογική της εφαρμογής.

4.3. Services

Οι υπηρεσίες αποτελούν κλάσεις που αποσκοπούν στην ενθυλάκωση της λογικής της εφαρμογής. Χρησιμοποιούν την κλάση «DataConnection» για να πάρουν τα δεδομένα που θέλουν από την τοπική βάση δεδομένων «SQLite» και στη συνέχεια με διάφορες μεθόδους τα επεξεργάζονται για να χρησιμοποιηθούν μετέπειτα από τις φόρμες της εφαρμογής.

Συγκεκριμένα, η κλάση «**AuthenticationService»** παρέχει τις δυνατότητες για την εγγραφή, είσοδο και αποσύνδεση ενός χρήστη ενώ παράλληλα λειτουργεί ως σημείο αναφοράς στον συνδεδεμένο χρήστη.

Η κλάση «**StatisticsService**» διευκολύνει την σύλληψη στατιστικών για έναν χρήστη για κάθε μία από τις ενότητες της εφαρμογής.

Τέλος, η κλάση «**ProgressService**» δίνει την δυνατότητα λήψης και οργάνωσης πληροφοριών και στατιστικών που αφορούν τη γενική πρόοδο του μαθητή στην εφαρμογή.

5. Φόρμες

Για το UI χρησιμοποιήθηκαν αντικείμενα από το toolbox του Visual Studio και δόθηκε έμφαση στη καλή εμφάνιση και διεπαφή του χρήστη.

5.1. LoginForm

Η φόρμα αυτή αποτελεί το σημείο εισόδου της εφαρμογής. Χρησιμοποιεί ένα αντικείμενο «auth» της κλάσης «AuthenticationService» και μέσω της μεθόδου «auth.Login» ελέγχει εάν τα συμπληρωμένα στοιχεία του χρήστη υπάρχουν σαν εγγραφή στη τοπική βάση δεδομένων. Εάν υπάρχει εγγραφή με τα δοθέντα στοιχεία τότε γίνεται μεταφορά στην κεντρική σελίδα «MainForm», αλλιώς εμφανίζεται αντίστοιχο μήνυμα.

```
public partial class LoginForm : Form
{
    private AuthenticationService auth;

    2 references
    public LoginForm()
    {
        InitializeComponent();
        auth = new AuthenticationService();
    }

    1 reference
    private void button_login_Click(object sender, EventArgs e)
    {
        try
        {
            User u = auth.Login(textBox_username.Text, textBox_password.Text);
            this.Hide();
            MainForm main = new MainForm(auth);
            main.ShowDialog();
            this.Close();
        } catch (InvalidCredentialsException)
        {
            MessageBox.Show("Ο λογαριασμός δεν βρέθηκε!");
            textBox_username.Clear();
            textBox_password.Clear();
        }
    }
```

Για την εγγραφή στην εφαρμογή καλείται η σελίδα εγγραφής «RegisterForm».

```
1 reference
private void linkLabel_register_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
{
    this.Hide();
    RegisterForm rf = new RegisterForm(auth);
    rf.ShowDialog();
    this.Show();
}
```

5.2. RegisterForm

Η φόρμα αποτελεί τη φόρμα εγγραφής ενός χρήστη στην εφαρμογή. Αρχικοποιεί ένα αντικείμενο «auth» της κλάσης «AuthenticationService» και μέσω της μεθόδου «auth.Register» πραγματοποιείται η εγγραφή στη τοπική βάση δεδομένων. Σε περίπτωση· μη συμπλήρωσης κάποιων πεδίων της φόρμας, δήλωσης ονόματος χρήστη που υπάρχει ήδη και οποιουδήποτε άλλου σφάλματος εμφανίζονται τα αντίστοιχα μηνύματα.

```
5 references
public partial class RegisterForm : Form
{
    AuthenticationService auth;

    1 reference
    public RegisterForm(AuthenticationService auth)
    {
        InitializeComponent();
        this.auth = auth;
    }

    1 reference
    private void button_register_Click(object sender, EventArgs e)
    {
        if (String.IsNullOrEmpty(textBox_name.Text) || String.IsNullOrEmpty(textBox_surname.Text)
            || String.IsNullOrEmpty(textBox_email.Text) || String.IsNullOrEmpty(textBox_username.Text)
            || String.IsNullOrEmpty(textBox_password.Text))
        {
            MessageBox.Show("Πρέπει να συμπληρώσεις όλα τα πεδία!");
        }
        else
        {
            try
            {
                auth.Register(textBox_name.Text, textBox_surname.Text, textBox_email.Text, textBox_username.Text, textBox_password.Text);
                MessageBox.Show("Η εγγραφή έγινε με επιτυχία!");
                this.Close();
            } catch (UserExistsException uex)
            {
                MessageBox.Show(String.Format("Ο χρήστης με όνομα \"{0}\" υπάρχει ήδη.Επέλεξε κάτι διαφορετικό", uex.GetUsername()));
            } catch (Exception)
            {
                MessageBox.Show("Ξαφνικό σφάλμα! Προσπάθησε ξανα!");
            }
        }
    }
```

5.3. MainForm

Αποτελεί την κεντρική σελίδα της εφαρμογής και περιέχει το βασικό μενού περιήγησης, τις ξεκλειδωμένες ενότητες και την πρόσβαση στα επαναληπτικά τεστ. Κατά την δημιουργία της αρχικοποιείται· ένα αντικείμενο «auth» της κλάσης «AuthenticationService», ένα αντικείμενο «statisticsService» της κλάσης «StatisticsService» και ένα αντικείμενο «progressService» της κλάσης «ProgressService».

```
public partial class MainForm : Form
{
    private AuthenticationService auth;
    private StatisticsService statisticsService;
    private int[] units;
    private ProgressService progressService;
    private List<double> unitsTestsSuccess;
    private List<double> unitsTestsFailure;

    3 references
    public MainForm(AuthenticationService auth)
    {
        InitializeComponent();
        this.auth = auth;
        this.statisticsService = new StatisticsService(auth.GetUser());
        this.progressService = new ProgressService(auth.GetUser());
    }
```

Κατά τη φόρτωση της φόρμας, παραμετροποιούνται στοιχεία της για να φαίνεται το ονοματεπώνυμο του χρήστη, με τη βοήθεια του αντικειμένου «progressService» ορίζονται δύο λίστες που περιέχουν τα ποσοστά επιτυχίας (true) και αποτυχίας (false) των τεστ αυτοαξιολόγησης ανά ενότητα και μέσω της «LoadUnlockedUnits()» γίνεται η εμφάνιση των κεφαλαίων που ο χρήστης έχει ξεκλειδώσει.

```
private void MainForm_Load(object sender, EventArgs e)
{
    User user = auth.GetUser();
    label_username.Text = user.GetName() + " " + user.GetSurname();
    unitsTestsSuccess = progressService.GetUnitTestScore(true);
    unitsTestsFailure = progressService.GetUnitTestScore(false);
    LoadUnlockedUnits();
}
```

Η «LoadUnlockedUnits()» ορίζει μία λίστα ακεραίων που περιέχει τους αριθμούς των κεφαλαίων που ο χρήστης έχει ολοκληρώσει. Προστίθεται στη λίστα αυτή το επόμενο κεφάλαιο από την τελευταία ολοκληρωμένη ενότητα. Βάση της παραπάνω λίστας εμφανίζονται στην κεντρική οθόνη οι αντίστοιχες ενότητες με τα ζωάκια τους. Ακόμη, βασιζόμενοι στις ενότητες που ο χρήστης έχει επιτύχει τουλάχιστον μία φορά, εάν το ποσοστό αποτυχίας είναι μεγαλύτερο από το ποσοστό επιτυχίας της ενότητας αυτό δείχνει μία αδυναμία του χρήστη στη συγκεκριμένη ενότητα και εμφανίζεται το αντίστοιχο συμβολο κοντά στο ζωάκι της ενότητας.

```
private void LoadUnlockedUnits()
{
    List<int> unitList = statisticsService.GetCompletedUnits();
    units = unitList.ToArray();
    if (!unitList.Count.Equals(9) && statisticsService.GetStatistics(unitList.Count.ToString()).IsUnlocked())
    {
        unitList.Add(unitList.Count + 1);
    }
    if (unitList.Count.Equals(0))
    {
        unitList.Add(unitList.Count + 1);
    }
    var pictureBoxUnits = panel_units.Controls.OfType<PictureBox>();
    foreach (int unit in unitList)
    {
        foreach (PictureBox pictureBox in pictureBoxUnits)
        {
            if (pictureBox.Name.Contains(unit.ToString()) && !pictureBox.Name.Contains("problem"))
            {
                pictureBox.Visible = true;
                pictureBox.Enabled = true;
            }
        }
        foreach (Label label in panel_units.Controls.OfType<Label>().Where(n => n.Text.Contains(unit.ToString())))
        {
            label.Visible = true;
        }
    }
    foreach(int unit in progressService.GetUnlockedUnits())
    {
        //Unit problem
        if(unitsTestsFailure.ElementAt(unit - 1) > unitsTestsSuccess.ElementAt(unit - 1))
        {
            string pbName = "pictureBox_problem_" + unit.ToString();
            foreach (PictureBox pb in pictureBoxUnits.Where(p => p.Name.Equals(pbName)))
            {
                pb.Visible = true;
                pb.Enabled = true;
            }
        }
    }
}
```

Όταν κάποια ενότητα επιλεχθεί, πατώντας ο χρήστης το ζωάκι της ενότητας, καλείται η μέθοδος «SelectUnit». Απομονώνοντας τον αριθμό της ενότητας, καλείται η φόρμα της θεωρίας με παράμετρο την επιλεγμένη ενότητα και την τιμή «false» στην παράμετρο για το υποχρεωτικό τεστ αυτοαξιολόγησης.

```
private void SelectUnit(object sender, EventArgs e)
{
    PictureBox clickedPictureBox = sender as PictureBox;
    //Get the selected unit
    string[] pictureBoxName = clickedPictureBox.Name.Split('_');
    int unit = Int32.Parse(pictureBoxName[1]);

    this.Hide();
    TheoryForm theoryForm = new TheoryForm(auth, statisticsService, unit, false);
    theoryForm.ShowDialog();
    this.Close();
}
```

Στο κεντρικό μενού ο χρήστης έχει πρόσβαση στην σελίδα του προφίλ, της βοήθειας και έχει τη δυνατότητα να αποσυνδεθεί.

```
private void profileToolStripMenuItem_Click(object sender, EventArgs e)
{
    this.Hide();
    ProfileForm profileForm = new ProfileForm(auth);
    profileForm.ShowDialog();
    this.Show();
}

1 reference
private void helpToolStripMenuItem_Click(object sender, EventArgs e)
{
    this.Hide();
    HelpForm helpForm = new HelpForm();
    helpForm.ShowDialog();
    this.Show();
}

1 reference
private void logoutToolStripMenuItem_Click(object sender, EventArgs e)
{
    this.Hide();
    auth.Logout();
    LoginForm loginForm = new LoginForm();
    loginForm.ShowDialog();
    this.Close();
}
```

Όταν ο χρήστης επιλέξει να κάνει ένα επαναληπτικό τεστ καλείται η φόρμα «TestForm» με παράμετρο ως ενότητα το string «revision» και ως unit[] τον πίνακα των ολοκληρωμένων ενοτήτων.

```
private void button_revisionTest_Click(object sender, EventArgs e)
{
    this.Hide();
    TestForm testForm = new TestForm(auth, statisticsService, "revision", units, false);
    testForm.ShowDialog();
    this.Close();
}
```

Επίσης, ο χρήστης όταν θέλει να κάνει ένα τεστ αυτοαξιολόγης απευθείας από την κεντρική σελίδα καλείται η μέθοδος «TakeUnitSelfAssessmentTest()». Απομονώνοντας τον αριθμό της ενότητας, καλείται η φόρμα των τεστ με παράμετρο την επιλεγμένη ενότητα και την τιμή «false» στην παράμετρο για το υποχρεωτικό τεστ αυτοαξιολόγησης.

```
private void TakeUnitSelfAssessmentTest(object sender, EventArgs e)
{
    PictureBox clickedPictureBox = sender as PictureBox;
    //Get the selected unit
    string[] pictureBoxName = clickedPictureBox.Name.Split('_');
    int unit = Int32.Parse(pictureBoxName[2]);

    this.Hide();
    TestForm testForm = new TestForm(auth, statisticsService, unit.ToString(), new int[] { unit }, false);
    testForm.ShowDialog();
    this.Close();
}
```

Εάν ο χρήστης εμφανίζει πρόβλημα σε μία ενότητα θα εμφανίζεται ένα συμβολο προσοχής δίπλα από την αντίστοιχη ενότητα. Πατώντας το εκτελείται η «ReadTheory», η οποία απομονώνει τον αριθμό της ενότητας, εμφανίζει μήνυμα επανάληψης της θεωρίας και καλεί τη φόρμα θεωρίας με παράμετρους την ενότητα και την τιμή του υποχρεωτικού τεστ αυτοαξιολόγησης ίση με «true».

```
private void ReadTheory(object sender, EventArgs e)
{
    PictureBox clickedPictureBox = sender as PictureBox;
    //Get the selected unit
    string[] pictureBoxName = clickedPictureBox.Name.Split('_');
    int unit = Int32.Parse(pictureBoxName[2]);

    MessageBox.Show("Πρέπει να κάνεις επανάληψη τη θεωρία του κεφαλαίου "+unit.ToString() +" και να κάνεις ένα 'Τεστ Αυτοαξιολόγησης'.");
    this.Hide();
    TheoryForm theoryForm = new TheoryForm(auth, statisticsService, unit, true);
    theoryForm.ShowDialog();
    this.Close();
}
```

5.4. TheoryForm

Αποτελεί την φόρμα παρουσίασης της θεωρίας της εφαρμογής. Κατά την δημιουργία της αρχικοποιείται· ένα αντικείμενο «auth» της κλάσης «AuthenticationService», ένα αντικείμενο «statisticsService» της κλάσης «StatisticsService», μία ακέραια μεταβλητή «unit» και μία λογική μεταβλητή «mandatoryTest». Οι δύο τελευταίες μεταβλητές περιγράφουν αντίστοιχα τον αριθμό της ενότητας που επέλεξε ο χρήστης και εάν είναι υποχρεωτικό ή όχι ο χρήστης να υποβληθεί σε τεστ αυτοαξιολόγησης.

```
9 references
public partial class TheoryForm : Form
{
    private AuthenticationService auth;
    private StatisticsService statisticsService;
    private int unit;
    private bool mandatoryTest;
    3 references
    public TheoryForm(AuthenticationService _auth, StatisticsService _statisticsService,  int _unit, bool _mandatoryTest)
    {
        InitializeComponent();
        this.auth = _auth;
        this.statisticsService = _statisticsService;
        this.unit = _unit;
        this.mandatoryTest = _mandatoryTest;
    }
```

Κατά τη φόρτωση της φόρμας, παραμετροποιούνται στοιχεία της για να αντιπροσωπεύουν την ενότητα που επιλέχθηκε από τον χρήστη. Πιο συγκεκριμένα ο αριθμός, το ζωάκι καθώς και ο πίνακας της προπαίδειας της ενότητας παίρνουν τις αντίστοιχες τιμές. Μέσω της μεθόδου «AddTheoryRevision» του «statisticsService» αυξάνεται η επισκεψιμότητα της ενότητας κατά ένα. Εάν η τιμή της μεταβλητής «mandatoryTest» είναι «true» το κουμπί επιστροφής στην αρχική σελίδα απενεργοποιείται και δεν είναι εμφανή στον χρήστη. Αυτό γίνεται για να υποχρεούται ο χρήστης να κάνει τεστ αυτοαξιολόγησης.

```
1 reference
private void TheoryForm_Load(object sender, EventArgs e)
{
    label_unit.Text = unit.ToString();
    label_multiTableNum.Text = unit.ToString();
    pictureBox_helper.Image = (Image)Properties.Resources.ResourceManager.GetObject(unit.ToString());
    Load_Unit_Theory(unit);

    // Keep track of the theory revision
    statisticsService.AddTheoryRevision(unit.ToString());

    //If Theory page is loaded after 3 failed attempts from TestForm or from pressing the problem button in MainForm
    //-> Self-assessment test has to be mandatory.
    if (mandatoryTest)
    {
        backToolStripMenuItem.Visible = false;
        backToolStripMenuItem.Enabled = false;
    }
}
```

Για τη φόρτωση του πίνακα της προπαίδειας χρησιμοποιείται η μέθοδος «Load_Unit_Theory(int unit)». Ανάλογα με τα ονόματα που έχουν δοθεί στα χαρακτηριστικά «Label» του πάνελ «panel_mutiplicationTable» της φόρμας κάποια παίρνουν την τιμή του αριθμού της ενότητας και άλλα παίρνουν την τιμή του αποτελέσματος του πολλαπλασιασμού του αριθμού της ενότητας με τους αριθμούς 1 εώς 10. Επίσης μέσω μίας λίστας χρωμάτων επιλέγεται τυχαία κάθε φορά ένα χρώμα για το πάνελ.

```
private void Load_Unit_Theory(int unit)
{
    //Load Multiplication number and results.
    var multiLabels = panel_multiplicationTable.Controls.OfType<Label>().Where(n => n.Name.Contains("label_multiNum"));
    foreach (Label l in multiLabels)
    {
        l.Text = unit.ToString();
    }

    List<int> results = new List<int>(){1*unit, 2*unit, 3*unit, 4*unit, 5*unit, 6*unit, 7*unit, 8*unit, 9*unit, 10*unit};
    var resultsLabels = panel_multiplicationTable.Controls.OfType<Label>().Where(n => n.Name.Contains("label_result")).OrderBy(t => t.TabIndex);
    int count = 0;
    foreach (Label l in resultsLabels)
    {
        l.Text = results.ElementAt(count).ToString();
        count++;
    }

    //Set Multiplication table colour.
    List<Color> colourList = new List<Color>()
    {
        Color.Salmon, Color.Coral, Color.Gold, Color.DarkKhaki, Color.YellowGreen, Color.LightGreen,
        Color.SpringGreen, Color.Turquoise, Color.Aquamarine, Color.LightBlue, Color.LightSteelBlue,
        Color.Plum, Color.Violet, Color.PaleVioletRed, Color.LightPink
    };
    Random random = new Random();
    int index = random.Next(colourList.Count-1);
    panel_multiplicationTable.BackColor = colourList.ElementAt(index);
}
```

Τέλος, με το πάτημα του κουμπιού «Τεστ Αυτοαξιολόγησης» της φόρμας καλείται η φόρμα «TestForm», δίνοντας τιμές στις παραμέτρους.

```
1 reference
private void button_test_Click(object sender, EventArgs e)
{
    this.Hide();
    TestForm testForm = new TestForm(auth, statisticsService, unit.ToString(), new int[]{ unit }, mandatoryTest);
    testForm.ShowDialog();
    this.Close();
}
```

5.5. TestForm

Αποτελεί την φόρμα εκτέλεσης των τεστ αυτοαξιολόγησης αλλά και του επαναληπτικού τεστ. Κατά την δημιουργία της αρχικοποιείται· ένα αντικείμενο «auth» της κλάσης «AuthenticationService» και ένα αντικείμενο «statisticsService» της κλάσης «StatisticsService», μία μεταβλητή «quiz_id» που χαρακτηρίζει τον τύπο του τεστ (1-9 για τα τεστ αυτοαξιολόγησης και revision για τα επαναληπτικά τεστ), ένας ακέραιος πίνακας «unit» που περιέχει τις ενότητες που θα συμπεριληφθούν στο τεστ, μία λογική μεταβλητή «mandatoryTest» που δηλώνει αν το τεστ πρέπει υποχρεωτικά να γίνει, μία λίστα από τα πάνελ των τριών ειδών ερωτήσεων που έχουμε στα τεστ, ένα αντικείμενο «random» της κλάσης «Random», μία λογική λίστα «answerList» που θα περιέχει εάν ο χρήστης απάντησε σωστά σε κάθε ερώτηση και τίθεται ο αριθμός των ερωτήσεων ανάλογα με τον τύπο του τεστ.

```
public partial class TestForm : Form
{
    private AuthenticationService auth;
    private StatisticsService statisticsService;
    private string quiz_id;
    private int[] unit;
    private List<Panel> panelQuestionList;
    private Random random;
    private List<bool> answerList;
    private int questionCount, maxNumberOfQuestions, failures = 0;
    private List<int> randomNumberMultiplier;
    private bool mandatoryTest;

    3 references
    public TestForm(AuthenticationService _auth, StatisticsService _statisticsService, string _quiz_id, int[] _unit, bool _mandatoryTest)
    {
        InitializeComponent();
        this.auth = _auth;
        this.statisticsService = _statisticsService;
        this.quiz_id = _quiz_id;
        this.unit = _unit;
        this.mandatoryTest = _mandatoryTest;
        panelQuestionList = new List<Panel>() { panel_fillTheBlank, panel_trueOrFalse, panel_multipleChoice };
        random = new Random();
        answerList = new List<bool>();
        if (quiz_id.Equals("revision"))
        {
            this.maxNumberOfQuestions = 10;
            label_Test.Text = "Επαναληπτικό Τεστ";
        }
        else
        {
            this.maxNumberOfQuestions = 5;
        }
    }
```

Κατά τη φόρτωση της φόρμας, παραμετροποιούνται στοιχεία της για να αντιπροσωπεύουν την επιλεγμένη ενότητα. Πιο συγκεκριμένα τίθεται το ζωάκι της ενότητας ή του επαναληπτικού τεστ και εάν το τεστ είναι υποχρεωτικό αφαιρείται η δυνατότητα επιστροφής στην κεντρική σελίδα.

```
private void TestForm_Load(object sender, EventArgs e)
{
    if (quiz_id.Equals("revision"))
    {
        pictureBox_helper.Image = (Image)Properties.Resources.ResourceManager.GetObject("revision");
    }
    else
    {
        pictureBox_helper.Image = (Image)Properties.Resources.ResourceManager.GetObject(unit[0].ToString());
    }
    if (mandatoryTest)
    {
        backToolStripMenuItem.Visible = false;
        backToolStripMenuItem.Enabled = false;
    }
}
```

Όταν ο χρήστης πατήσει το κουμπί «Έναρξη» καλείται η «button_start_Click()». Εμφανίζονται και εξαφανίζονται τα απαραίτητα στοιχεία στη φόρμα, αρχικοποιούνται κάποιες μεταβλητές και καλείται η μέθοδος «GenerateQuestion(int[] unit)».

```
private void button_start_Click(object sender, EventArgs e)
{
    pictureBox_message.Image = (Image)Properties.Resources.ResourceManager.GetObject("messageCloud");
    button_start.Visible = false;
    button_next_end.Visible = true;
    if (!quiz_id.Equals("revision"))
    {
        randomNumberMultiplier = new List<int>() { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
    }
    questionCount = 1;
    label_correctLabel.Visible = false;
    label_correct.Visible = false;
    answerList.Clear();
    GenerateQuestion(unit);
}
```

Η «GenerateQuestion(int[] unit)» επιλέγει τυχαία μέσα από τη λίστα ερωτήσεων των πάνελ ένα τύπο ερώτησης και τον εμφανίζει στην οθόνη. Ανάλογα με τον τύπο του τεστ ορίζει τον αριθμό της ενότητας που θέλουμε να πολλαπλασιαστεί με έναν τυχαίο αριθμό μεταξύ του 1-10 και υπολογίζει το αποτέλεσμα. Στην συνέχεια καλεί την «FillQuestion()».

```
private void GenerateQuestion(int[] unit)
{
    //Randomly pick a type of Question through the panelQuestionList
    int index = random.Next(panelQuestionList.Count);
    Panel panel = panelQuestionList.ElementAt(index);
    panel.Visible = true;
    int unitNumber = 0;
    int randomNumber = 0;

    //Pick the unitNumber
    if (quiz_id.Equals("revision"))
    {
        index = random.Next(unit.Length);
        unitNumber = unit[index];
        randomNumber = random.Next(1, 11);
    }
    else
    {
        unitNumber = unit[0];
        index = random.Next(randomNumberMultiplier.Count);
        randomNumber = randomNumberMultiplier.ElementAt(index);
        randomNumberMultiplier.RemoveAt(index); // TODO This fails after 2 failures. Needs fixing.
    }

    //Randomly pick the number that will get multiplied with the unit number

    int resultNumber = unitNumber * randomNumber;
    FillQuestion(panel, unitNumber, randomNumber, resultNumber);
}
```

Η μέθοδος «FillQuestion(Panel panel, int unitNumber, int randomNumber, int resultNumber)» τοποθετεί τις τιμές των παραμέτρων της στις σωστές θέσεις ανάλογα με τον τύπο της ερώτησης. Έχουμε τρεις τύπους ερωτήσεων· συμπλήρωση απάντησης, σωστό ή λάθος και πολλαπλής επιλογής. Έτσι παράγεται και εμφανίζεται διαφορετική ερώτηση κάθε φορά.

```
private void FillQuestion(Panel panel, int unitNumber, int randomNumber, int resultNumber)
{
    var panelLabels = panel.Controls.OfType<Label>().Where(n => n.Name.Contains("label_number")).OrderBy(t => t.TabIndex);
    panelLabels.ElementAt(0).Text = unitNumber.ToString();
    panelLabels.ElementAt(1).Text = randomNumber.ToString();
    if (panel.Name.Equals("panel_trueOrFalse"))
    {
        int setTrueOrFalse = random.Next(2);
        if (setTrueOrFalse.Equals(0))
        {
            panelLabels.ElementAt(2).Text = resultNumber.ToString();
        }
        else
        {
            panelLabels.ElementAt(2).Text = random.Next(unitNumber, unitNumber * 10).ToString();
        }
    } else if (panel.Name.Equals("panel_multipleChoice"))
    {
        var radioButtons = panel.Controls.OfType<RadioButton>();
        int index = random.Next(3);
        radioButtons.ElementAt(index).Text = resultNumber.ToString();

        int randomNumber1 = random.Next(unitNumber, (unitNumber * 10) + 1);
        int randomNumber2 = random.Next(unitNumber, (unitNumber * 10) + 1);
        if (radioButton_choice1.Text.Equals("_") && radioButton_choice2.Text.Equals("_"))
        {
            radioButton_choice1.Text = randomNumber1.ToString();
            radioButton_choice2.Text = randomNumber2.ToString();
        }
        else if (radioButton_choice2.Text.Equals("_") && radioButton_choice3.Text.Equals("_"))
        {
            radioButton_choice2.Text = randomNumber2.ToString();
            radioButton_choice3.Text = randomNumber1.ToString();
        } else if (radioButton_choice1.Text.Equals("_") && radioButton_choice3.Text.Equals("_"))
        {
            radioButton_choice1.Text = randomNumber1.ToString();
            radioButton_choice3.Text = randomNumber2.ToString();
        }
    }
}
```

Όταν ο χρήστης πατήσει το κουμπί «Επόμενο» τότε καλείται η «button_next_end_Click()». Αρχικά ελέγχεται η ερώτηση με τη βοήθεια της μεθόδου «CheckQuestion()». Η μέθοδος αυτή ανάλογα με τον τύπο της ερώτησης, ελέγχει εάν ο χρήστης έχει απαντήσει στην ερώτηση και μετά ελέγχει την απάντησή του. Στη «answerList» λίστα προστίθεται «true» εάν ο χρήστης έχει απαντήσει σωστά και «false» εάν έχει απαντήσει λάθος. Αν ο χρήστης δεν έχει απαντήσει τον ενημερώνει με κατάλληλο μήνυμα.

```
private void button_next_end_Click(object sender, EventArgs e)
{
    bool answerChecked = CheckQuestion();
    if (answerChecked)
    {
        //See if it's the end of the Test.
        if (questionCount.Equals(maxNumberOfQuestions))
        {
            // Count the correct answers
            int correctAnswers = answerList.Where(answer => answer.Equals(true)).Count();

            //Calculate success percentage for Test
            double successPer = ((double)correctAnswers / (double)maxNumberOfQuestions) * 100;
            if (successPer <= 50)
            {
                pictureBox_message.Image = (Image)Properties.Resources.ResourceManager.GetObject("messageCloudFailed");
                button_start.Visible = true;
                ClearQuestions();

                // If the student fails 3 times, suggest them to revise the theory of the unit
                if (++failures >= 3 && !quiz_id.Equals("revision"))
                {
                    MessageBox.Show("Μια επανάληψη στην προπαίδεια του " + unit[0].ToString() + " θα σε βοηθήσει να τα πας καλύτερα! \nΜετά την επανάληψη πρέπει να ξανακάνεις ένα 'Τεστ Αυτοαξιολ
                    this.Hide();
                    TheoryForm theoryForm = new TheoryForm(auth, statisticsService, unit[0], true);
                    theoryForm.ShowDialog();
                    this.Close();
                }
            }
            else
            {
                pictureBox_message.Image = successPer <= 70 ? (Image)Properties.Resources.ResourceManager.GetObject("messageCloudBravo") :
                    (Image)Properties.Resources.ResourceManager.GetObject("messageCloudExcellent");
                button_start.Visible = true;
                button_start.Text = "ΤΕΛΟΣ";
                button_start.BackColor = Color.BurlyWood;
                button_start.Enabled = false;
                backToolStripMenuItem.Visible = true;
                backToolStripMenuItem.Enabled = true;
            }
```

Στη συνέχεια, στην «button_next_end_Click()» εάν έχει απαντηθεί η ερώτηση τότε ελέγχεται αν η ερώτηση είναι η τελευταία του τεστ. Αν όχι, τότε καλείται η «NextQuestion(int[] unit)». Αυτή με τη σειρά της καλεί την «ClearQuestions()» η οποία επαναφέρει τα πανελ των ερωτήσεων στην αρχική τους κατάσταση. Ύστερα, αυξάνει τον μετρητή των ερωτήσεων κατά ένα και καλεί την «GenerateQuestion(int[] unit)» για να παραχθεί η επόμενη ερώτηση.

```
private void NextQuestion(int[] unit)
{
    //Clear question panels
    ClearQuestions();

    questionCount++;
    GenerateQuestion(unit);
}
```

Εάν όμως, στην «button_next_end_Click()» η ερώτηση είναι η τελευταία του τεστ τότε γίνεται η μέτρηση των σωστών απαντήσεων και υπολογίζεται το ποσοστό επιτυχίας του εκάστοτε τεστ. Εάν το ποσοστό επιτυχίας είναι κάτω του 50% εμφανίζεται μήνυμα αποτυχίας, καθαρίζονται τα πάνελ με τη «ClearQuestions()» και εμφανίζεται το κουμπί «Έναρξη». Εάν ένας χρήστης αποτύχει σε τρία συνεχόμενα τεστ αυτοαξιολόγησης εμφανίζεται μήνυμα ενημέρωσης για επανάληψη της θεωρίας και ανοίγει η φόρμα της θεωρίας της ενότητας με υποχρεωτικό τεστ σαν όρισμα. Αν το ποσοστό επιτυχίας είναι άνω του 50% τότε εμφανίζεται αντίστοιχο μήνυμα και το τεστ έχει τελειώσει. Τότε ο χρήστης μπορεί να πάει πίσω στην κεντρική σελίδα και να προχωρήσει στην επόμενη ενότητα.

Ανεξαρτήτως του σκορ (ποσοστού επιτυχίας) του χρήστη στο τεστ εμφανίζεται ο αριθμός των σωστών απαντήσεων έναντι του συνολικού αριθμού ερωτήσεων και με τη βοήθεια της μεθόδου «InsertScore» του αντικειμένου «statisticsService» αποθηκεύεται η προσπάθειά του αυτή στη τοπική βάση δεδομένων.

```
            // Hide all the question panels
            panelQuestionList.ForEach(panel => panel.Visible = false);
            button_next_end.Visible = false;
            label_correctLabel.Visible = true;
            label_correct.Visible = true;
            label_correct.Text = correctAnswers.ToString() + "/" + maxNumberOfQuestions.ToString();

            // Store the sucess percentage for statistics
            statisticsService.InsertScore(quiz_id, successPer);
        }
        else
        {
            NextQuestion(unit);
        }
    }
}
```

5.6. ProfileForm

Αποτελεί την φόρμα παρουσίασης των προσωπικών και στατιστικών στοιχείων της εφαρμογής. Κατά την δημιουργία της αρχικοποιείται ένα αντικείμενο «auth» της κλάσης «AuthenticationService» και ένα αντικείμενο «profressService» της κλάσης «ProgressService».

```
5 references
public partial class ProfileForm : Form
{
    private AuthenticationService auth;
    private ProgressService progressService;
    private List<double> unitsTestsSuccess;
    private List<double> unitsTestsFailure;
    private double revisionTestSuccess;
    private double revisionTestFailure;
    private List<int> units;

    1 reference
    public ProfileForm(AuthenticationService _auth)
    {
        InitializeComponent();
        this.auth = _auth;
        this.progressService = new ProgressService(auth.GetUser());
    }
```

Κατά τη φόρτωση της φόρμας, παραμετροποιούνται στοιχεία της για να αντιπροσωπεύουν τον συνδεδεμένο χρήστη. Πιο συγκεκριμένα το όνομα, το επώνυμο, και το εμαιλ του χρήστη αντιπροσωπεύουν τα προσωπικά στοιχεία του. Ακόμη, παρουσιάζεται ο αριθμός των ολοκληρωμένων ενοτήτων του χρήστη και γίνεται φόρτωση της προόδου του μέσω της «LoadProgress».

```
1 reference
private void ProfileForm_Load(object sender, EventArgs e)
{
    label_name.Text = auth.GetUser().GetName();
    label_surname.Text = auth.GetUser().GetSurname();
    label_email.Text = auth.GetUser().GetEmail();

    LoadProgress();
    label_unit.Text = units.Count.ToString();
    LoadCharts(unitsTestsSuccess, unitsTestsFailure);
}

1 reference
public void LoadProgress()
{
    unitsTestsSuccess = progressService.GetUnitTestScore(true);
    unitsTestsFailure = progressService.GetUnitTestScore(false);
    units = progressService.GetUnlockedUnits();
    revisionTestSuccess = progressService.GetRevisionRates(true);
    revisionTestFailure = progressService.GetRevisionRates(false);
}
```

Μέσω της μεθόδου «GetUnitTestScore(bool success)» του «progressService» αντικειμένου και την τιμή παραμέτρου «true» και «false» ορίζονται για κάθε ενότητα το ποσοστό επιτυχιών και αποτυχιών των τεστ αυτοαξιολόγησης αντίστοιχα σε μορφή «List<double>». Μέσω της «GetUnlockedUnits» του «progressService» ορίζεται ο αριθμός των ολοκληρωμένων ενοτήτων, δηλαδή αυτών που έχει επιτύχει σε τουλάχιστον ένα τεστ αυτοαξιολόγησης. Μέσω της «GetRevisionRates(bool success)» του «progressService» και την τιμή παραμέτρου «true» και «false» υπολογίζεται το ποσοστό επιτυχίας και αποτυχίας των επαναληπτικών τεστ.

Τέλος, κατά την φόρτωση της φόρμας, εφόσον έχουν υπολογιστεί τα επιθυμητά ποσοστά και έχουν αποθηκευτεί οι τιμές αυτών σε μεταβλητές και λίστες, γίνεται η παρουσίασή τους σε διαγράμματα με τη μέθοδο «LoadCharts(List<double> successRates, List<double> failureRates)».

```
private void LoadCharts(List<double> successRates, List<double> failureRates)
{
    foreach (int unit in units)
    {
        string quiz_id = unit.ToString();

        //Load rates of Self-assessment tests
        chart_unitRates.Series["Επιτυχία"].Points.AddXY("Ενότητα: "+ quiz_id, successRates.ElementAt(unit-1));
        chart_unitRates.Series["Αποτυχία"].Points.AddXY("Ενότητα: " + quiz_id, failureRates.ElementAt(unit - 1));

        // Load theory revisions
        chart_theoryRevisions
            .Series["Επαναλήψεις θεωρίας"]
            .Points.AddXY(
                "Προπαίδεια του " + quiz_id,
                progressService.GetStatistics(quiz_id).GetTheoryRevisions()
            );

        //Load units that the user faces understanding problems
        if (failureRates.ElementAt(unit - 1) > successRates.ElementAt(unit - 1))
        {
            chart_unitProblems.Series["Ενότητες"].Points.AddXY("Ενότητα: " + quiz_id, 100);
        }
        else
        {
            chart_unitProblems.Series["Ενότητες"].Points.AddXY("Ενότητα: " + quiz_id, 0);
        }
    }

    //Load rates of Revision tests
    chart_revisionRates.Series["Revision"].Points.AddXY("Επιτυχία", revisionTestSuccess.ToString());
    chart_revisionRates.Series["Revision"].Points.AddXY("Αποτυχία", revisionTestFailure.ToString());
    chart_revisionRates.Series["Revision"].Points[1].Color = Color.Red;
}
```

Τα στατιστικά που παρουσιάζονται είναι· τα ποσοστά επιτυχίας-αποτυχίας ανά ενότητα, η επισκεψιμότητα της θεωρίας ανά ενότητα, το ποσοστό αποτυχίας-επιτυχίας των επαναληπτικών τεστ και σε ποιες ενότητες ο χρήστης παρουσιάζει πρόβλημα κατανόησης.

5.7. HelpForm

Αποτελεί τη φόρμα βοήθειας της εφαρμογής. Αναλύει και παρουσιάζει με εικόνες τις οδηγίες χρήσης της εφαρμογής για να διευκολύνει και να βοηθήσει το χρήστη με προβλήματα που μπορεί να συναντήσει κατά τη διάρκεια εκτέλεσης του προγράμματος.

```
11 references
public partial class HelpForm : Form
{
    4 references
    public HelpForm()
    {
        InitializeComponent();
    }

    1 reference
    private void backToolStripMenuItem_Click(object sender, EventArgs e)
    {
        this.Close();
    }
}
```

5.8. Γενικά Χαρακτηριστικά

5.8.1. Κουμπί «Πίσω»

Με το κουμπί πίσω που βρίσκεται στα μενού των σελίδων της εφαρμογής ο χρήστης μπορεί να μετακινείται είτε στην κεντρική σελίδα της εφαρμογής.

```
private void backToolStripMenuItem_Click(object sender, EventArgs e)
{
    this.Hide();
    MainForm mainForm = new MainForm(auth);
    mainForm.ShowDialog();
    this.Close();
}
```

5.8.2. Κουμπί «Βοήθεια»

Με το κουμπί «Βοήθεια» που υπάρχει στα μενού των σελιδών της εφαρμογής ο χρήστης μπορεί να πηγαίνει στη φόρμα βοήθειας και να δει τις οδηγίες χρήσης της εφαρμογής.

```
private void helpToolStripMenuItem_Click(object sender, EventArgs e)
{
    this.Hide();
    HelpForm helpForm = new HelpForm();
    helpForm.ShowDialog();
    this.Show();
}
```

5.8.3. Κουμπί «Πληροφοριών/Βοήθειας»

Το κουμπί αυτό υπάρχει σε κάθε σελίδα της εφαρμογής και εμφανίζει ένα μήνυμα βοήθειας για το τι πρέπει να κάνει ο χρήστης στην εκάστοτε σελίδα. Ενδεικτικά:

```
private void button_help_Click(object sender, EventArgs e)
{
    MessageBox.Show("Για να μπεις στην εφαρμογή βάλε το 'Όνομα χρήστη' σου και τον 'Κωδικό' σου στα αντίστοιχα πεδία " +
        "και πάτησε το κουμπί 'Είσοδος'. \n" +
        "Εάν δεν έχεις λογαριασμό πάτησε το λινκ 'Δεν έχετε λογαριασμό; Εγγραφείτε εδώ'.");
}
```