ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Χρήση Πολυμέσων στη Διαδικτυακή Εξέταση

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

του

ΓΑΡΑΦΑΛΙΔΗ ΑΘΑΝΑΣΙΟΥ

(ΑΕΜ: 760)

Επιβλέπων : **Δρ. Σινάτκας Ιωάννης**

Καθηγητής

Καστοριά Μήνας - Έτος (παρουσίασης της εργασίας)

ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Χρήση Πολυμέσων στη Διαδικτυακή Εξέταση

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

του

ΓΑΡΑΦΑΛΙΔΗ ΑΘΑΝΑΣΙΟΥ

(ΑΕΜ: 760)

Επιβλέπων : **Δρ. Σινάτκας Ιωάννης**
Καθηγητής

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την ημερομηνία εξέτασης

.....	.....	.....
Ον/μο Μέλους	Ον/μο Μέλους	Ον/μο Μέλους
Ιδίοτητα Μέλους	Ιδίοτητα Μέλους	Ιδίοτητα Μέλους

Καστοριά Μήνας - Έτος (παρουσίασης της εργασίας

Ευχαριστίες

Περίληψη

Στη σημερινή εποχή, με τις αυξημένες ανάγκες που υφίστανται για απομακρυσμένη πρόσβαση σε πολλές υπηρεσίες και καθημερινές υποχρεώσεις, δε θα μπορούσε να διαφέρει και η δυνατότητα εξέτασης μαθημάτων από απόσταση. Και ενώ έχουν δημιουργηθεί και βελτιστοποιηθεί πολλές εφαρμογές για την βοήθεια στην εξέταση, σε αυτή την εργασία θα διερευνήσουμε τη δυνατότητα χρήσης πολυμέσων στην εξέταση ώστε να εξελίξουμε τον τομέα αυτό.

Τα πολυμέσα χρησιμοποιούνται σε πάρα πολλούς τομείς στην καθημερινότητα μας και η δυνατότητα για διαδικτυακή εξέταση μας δίνει την ευκαιρία να εξερευνήσουμε τις πιθανές χρήσεις των πολυμέσων προς όφελος της διαδικασίας της εξέτασης, και για τους εξεταστές αλλά και για τους εξεταζόμενους.

Θα χρειαστούμε μια βάση δεδομένων και μια εφαρμογή για την διεξαγωγή των εξετάσεων, και έχουμε επιλέξει από τα πιο γνωστά συστήματα βάσεων δεδομένων, τον SQL Server της Microsoft, και για την δημιουργία της εφαρμογής τη χρήση των WinForms μέσω του Microsoft Visual Studio γραμμένο σε γλώσσα προγραμματισμού C#. Θα προσπαθήσουμε να παρουσιάσουμε τα πλεονεκτήματα και τα μειονεκτήματα της διεξαγωγής διαδικτυακών εξετάσεων χρησιμοποιώντας την παρούσα εφαρμογή.

Λέξεις Κλειδιά: Πολυμέσα, Βάσεις Δεδομένων, MSSQL, Microsoft Visual Studio, Microsoft WinForms, C#

Abstract

Nowadays, with the increased needs that exist for remote access to many services and day-to-day obligations, the possibility of taking exams from distance could not be any different. And while many applications have been developed and optimized to help remotely taking exams, in this exercise we will explore the possibility of using multimedia in exams to evolve this area.

Multimedia is used in many areas in our daily lives and the possibility for online examination opens up the opportunity to explore the possible use of multimedia for the benefit of the examination process, both for the examiners as well as for the examinees.

We will need a database and an application to conduct the exams, and we have selected some of the most well-known database systems, Microsoft's SQL Server, and to create the application we used WinForms through Microsoft Visual Studio written in programming language C#. We will try to present the advantages and disadvantages of conducting online examination using this application.

Key Words: Multimedia, Databases, , MSSQL, Microsoft Visual Studio, Microsoft WinForms, C#

Πίνακας Περιεχομένων

Λίστα Εικόνων

Λίστα Πινάκων

1 Σκοπός

Ο σκοπός της εργασίας είναι να δώσουμε τη δυνατότητα στους καθηγητές αλλά και στους σπουδαστές να έχουν μια επεξήγηση για την απάντηση που έχουν επιλέξει στα διαδικτυακά τους διαγωνίσματα ώστε να μπορέσουν και οι μεν και οι δε να έχουν μια καλύτερη εικόνα για την γνώση πάνω στα θέματα που διαγωνίζονται.

Η ανάγκη για το παραπάνω προκύπτει καθώς λόγω της φύσης της διαδικτυακής εξέτασης ο καθηγητής δεν μπορεί να είναι σίγουρος εάν οι σπουδαστές έχουν αφομοιώσει την ύλη πάνω στην οποία εξετάζονται σε βάθος. Με την δυνατότητα να εξηγήσουμε σε ένα βίντεο λίγων δευτερολέπτων την επιλογή της απάντησής μας γίνεται πιο ξεκάθαρο το επίπεδο στο οποίο ο εξεταζόμενος έχει κατανοήσει το υλικό πάνω στο οποίο εξετάζεται.

2 Πολυμέσα (Multimedia)

Τα πολυμέσα θεωρούνται αποτελεσματικά στην εκπαίδευση, καθώς παρέχουν ευκολία και δυνατότητες σε αυτήν. Χάριν στη χρήση των πολυμέσων και των πρακτικών τους, οι μαθητές οποιασδήποτε βαθμίδας μπορούν να μάθουν καινούργιες πληροφορίες. Σε κάποιες περιπτώσεις, οι μαθητές μπορούν να εμπλουτίσουν τις γνώσεις τους και την πληροφόρησή τους με τέτοιους τρόπους, που δε θα ήταν εφικτό με τη χρήση των παραδοσιακών πρακτικών (1993, translated by Çeliköz, 1998).

Βάσει αυτού του χαρακτηριστικού των πολυμέσων, θα μπορούσε να αναφερθεί ότι η χρήση τους μπορεί να διευκολύνει και να βοηθήσει την εκμάθηση μαθητών με διαφορετικές μαθησιακές ικανότητες, διαφορετικά χαρακτηριστικά και διαφορετικούς τρόπους διδασκαλίας. Παράλληλα, τα πολυμέσα και η χρήση τους δίνουν τη δυνατότητα στους μαθητές να ασκήσουν ατομικά τη μαθησιακή τους ενασχόληση (Dwyer, 1993; trans by Çeliköz, 1998).

Επιπρόσθετα, έχει παρατηρηθεί ότι τα πολυμέσα προσδίδουν αυθεντικότητα και ποικιλομορφία στην εκμάθηση και στην εκπαίδευση. Ευρέως γνωστό το γεγονός ότι το μήνυμα που θα μεταδοθεί με τη χρήση πολυμέσων, θα έχει περισσότερους παραλήπτες, οι οποίοι θα μπορέσουν να βιώσουν την εμπειρία σε εξατομικευμένα μαθησιακά περιβάλλοντα. Οι υποκείμενοι στην εκπαίδευση μέσω των πολυμέσων μπορούν να εκπαιδευτούν με τη χρήση τεχνικών που βασίζονται σε διαδικτυακά ηχητικά και οπτικά μέσα, video και animations, με τέτοιο τρόπο που δεν θα μπορούσαν να διδαχθούν με τις παραδοσιακές τεχνικές εκπαίδευσης. Με αυτό τον τρόπο, η επαφή με την πραγματικότητα και η ολοκληρωμένη εκπαιδευτική διαδικασία επιτυγχάνονται στον μέγιστο βαθμό (Semerci, 1999).

Σε συνέχεια αυτού, έχει διαπιστωθεί ότι τα πολυμέσα διευκολύνουν την εκπαίδευση όσον αφορά τα δεδομένα που χρησιμοποιούνται, τον αποθηκευτικό χώρο που καταλαμβάνουν, τη δυνατότητα διαμοιρασμού και μεταφοράς του οπτικού και μη γραπτού εκπαιδευτικού υλικού, γραφημάτων, ηχητικών βοηθημάτων, και άλλων εκπαιδευτικών μέσων (Bitter, 1989; cited in Semerci, 1999). Παράλληλα, η χρήση πολυμέσων δημιουργεί ένα οικείο, ποικιλόμορφο, οικονομικό και πρακτικό περιβάλλον στην εκπαίδευση (Uşun, 2000). Ακόμα μία συνεισφορά των πολυμέσων είναι η αύξηση των ακαδημαϊκών επιτευγμάτων των μαθητών. Η χρήση των πολυμέσων επηρεάζει θετικά την εκπαίδευση, σε σύγκριση με τις παραδοσιακές πρακτικές (Akkoyunlu and Yılmaz, 2005).

Μελέτες υποδεικνύουν ότι η χρήση των πολυμέσων διευκολύνει και αντικειμενοποιεί την παιδευτική διαδικασία, μιας και παρουσιάζει παραπάνω από έναν τεχνολογικούς παράγοντες στον εκπαιδευόμενο, και απευθύνεται σε περισσότερες από μία αισθήσεις του.

Οι συνεχείς τεχνολογικές εξελίξεις έχουν επηρεάσει την εκπαίδευση αλλά και την εξέταση, όπως και έχει γίνει σε πολλούς ακόμα τομείς της καθημερινότητας. Ο αριθμός των εφαρμογών που βασίζονται στην τεχνολογία και χρησιμοποιούνται για την εκπαίδευση ολοένα και αυξάνεται, με βασικό εργαλείο τα πολυμέσα. Τα πολυμέσα προσδίδουν δυναμική ως τεχνολογία που χρησιμοποιείται για την εκμάθηση και εμπλουτίζουν τη μαθησιακή διαδικασία. Σύμφωνα με μελέτες που έχουν διεξαχθεί, οι μαθητές που λαμβάνουν ερεθίσματα τα οποία εμπεριέχουν εικόνες και λέξεις, έχουν καλύτερο αποτέλεσμα από αυτά που συσχετίζονται μόνο με λέξεις (Mayer, 1989; Mayer & Anderson, 1992; Mayer, Bove, Bryman, Mars, & Tapangco, 1996; Mayer & Moreno, 1998). [1]

Δεδομένα κειμένου, γραφικά, ήχος και προσομοίωση κίνησης χρησιμοποιούνται κατά την εφαρμογή των πολυμέσων. Όταν η εφαρμογή βρίσκεται σε εξέλιξη, ο χρήστης δεν διαθέτει τη δυνατότητα να παρέμβει σε αυτή, παρά μόνο να τη σταματήσει. Καθώς η εφαρμογή βρίσκεται σε εξέλιξη, ο χρήστης δεν μπορεί να αλλάξει τη ροή της, ούτε να πραγματοποιήσει επιλογές και αλλαγές διαφορετικές από αυτές που ήδη έχουν προγραμματιστεί. Η χρήση της εφαρμογής πολυμέσων αποβλέπει στη μετάδοση της πληροφορίας, και ανάλογα με τη δομή της και την αρχιτεκτονική της, γίνεται λόγος για χρήση διαλογικών πολυμέσων (Interactive Multimedia Application), είτε για χρήση υπερμέσων (Hypermedia Application).

2.1 Διαλογικά Πολυμέσα (Interactive Multimedia)

Τα διαλογικά πολυμέσα αποτελούνται από οποιοδήποτε ψηφιακό μέσο το οποίο επιτρέπει στο χρήστη να αλληλοεπιδράσει με αυτό και να ελέγξει, συνδυάσει και διαχειριστεί διαφορετικούς τύπου μέσων όπως κείμενο, ήχο, εικόνα, γραφικά και κινούμενες απεικονίσεις. Το εύρος χρήσης των διαλογικών πολυμέσων μεγαλώνει καθημερινά με αποτέλεσμα να τα βλέπουμε ολοένα και περισσότερο να χρησιμοποιούνται στην εκπαίδευση και όχι μόνο. Παρατηρούμε ότι χρησιμοποιούνται και σε διαχείριση επιχειρήσεων, διακυβέρνηση, ψυχαγωγία. Η χρήση των διαλογικών πολυμέσων στην εκπαίδευση έχει επιφέρει πολύ θετικά αποτελέσματα. Ειδικότερα εάν παραμετροποιήθει σύμφωνα με τις μαθησιακές ανάγκες και δυνατότητες της εκάστοτε εκπαιδευτικής βαθμίδας μπορεί να διευκολύνει σε μέγιστο βαθμό την διαδικασία της εκμάθησης. Σήμερα, έχουν θεμελιωθεί σε σχολεία και εκπαιδευτικά κέντρα λόγω της ευχρηστίας και της αποτελεσματικότητας τους. (INTERACTIVE MEDIA AND ITS IMPACT ON EDUCATION, Arshi Gouhar and Dr. Mahapatra 2016) [2]

Στην παρούσα εργασία, τα διαλογικά πολυμέσα θα χρησιμοποιηθούν για την βελτίωση της ποιότητας της διαδικτυακής εξέτασης με τη χρήση βιντεοσκόπησης των απαντήσεων των φοιτητών σε ερωτήματα που θα μπορεί να ορίσει ο καθηγητής ώστε να επιβεβαιώσει ότι ο εξεταζόμενος έχει κατανοήσει την μαθησιακή ύλη.

3 Microsoft SQL Server (MSSQL)

Ο Microsoft SQL Server (MSSQL) είναι ένα σχεσιακό σύστημα διαχείρισης βάσεων δεδομένων που παράχθηκε από τη Microsoft. Πρόκειται για ένα server βάσης δεδομένων (database server), όπου σαν προϊόν λογισμικού έχει την βασική λειτουργία της αποθήκευσης και ανάκτησης δεδομένων όπως αυτά χρειάζονται από άλλες εφαρμογές λογισμικού –οι οποίες μπορεί να τρέχουν στον ίδιο ή σε άλλους υπολογιστές στο δίκτυο (ή ακόμα και στο διαδίκτυο). Η Microsoft προσφέρει τουλάχιστον 12 διαφορετικές εκδόσεις του MSSQL, οι οποίες απευθύνονται σε διαφορετικά κοινά με διαφορετικές εργασιακές ανάγκες που εκτείνονται από μικρές εφαρμογές 1 χρήστη (single-machine applications) μέχρι και μεγάλες διαδικτυακές εφαρμογές με πολλούς ταυτόχρονους χρήστες. (Wikipedia) [3]

3.1 Χρησιμότητα του MSSQL

Για να εξετάσουμε τη χρησιμότητα του MSSQL πρέπει πρώτα να γνωρίζουμε πως θα χρησιμοποιηθούν τα δεδομένα μας. Εάν τα δεδομένα είναι κατά κύριο λόγο δομημένα, τότε η χρήση βάσης δεδομένων SQL είναι η σωστή επιλογή.

Μια βάση SQL είναι κατάλληλη για συστήματα βασισμένα σε συναλλαγές (transaction-oriented systems) όπως επί παραδείγματι εργαλεία διαχείρισης πελατειακών σχέσεων, λογισμικό για λογιστικές υπηρεσίες και πλατφόρμες online εμπορίου. Κάθε γραμμή σε μια βάση SQL έχει μια ξεχωριστή οντότητα (πχ. ένα πελάτη) και κάθε στήλη έχει μια ιδιότητα που περιγραφή την οντότητα (πχ. διεύθυνση, εργασιακός τίτλος, είδη που αγοράστηκαν κ.ο.κ.).

Εξ αίτιας αυτών των διακριτών, δομημένων σχέσεων μεταξύ γραμμών και στηλών σε κάθε πίνακα, οι βάσεις δεδομένων SQL είναι η βέλτιστη επιλογή όταν χρειάζεται να ακολουθούμε το μοντέλο ΑΣΑΑ (ACID compliance).

- Ατομικότητα (Atomicity) – Κάθε συναλλαγή επιτυγχάνει ή ανακαλείται πλήρως.

- Συνοχή (Consistency) – Δεδομένα που γίνονται εγγραφή στη βάση πρέπει να είναι έγκυρα βάσει των προκαθορισμένων κανόνων.

- Απομόνωση (Isolation) – Όταν συναλλαγές τρέχουν ταυτοχρόνως, δεν έρχονται σε αντιπαράθεση αλλά τρέχουν η μια μετά την άλλη.

- Αντοχή (Durability) – Όταν μια συναλλαγή εισέρχεται στην βάση δεδομένων, θεωρείται μόνιμη, ακόμα και σε περίπτωση βλάβης συστήματος.

Εάν τα δεδομένα που χρησιμοποιούνται είναι πολύ δομημένα και ακολουθούν το παραπάνω μοντέλο τότε η χρήση MSSQL είναι απαραίτητη. [4]

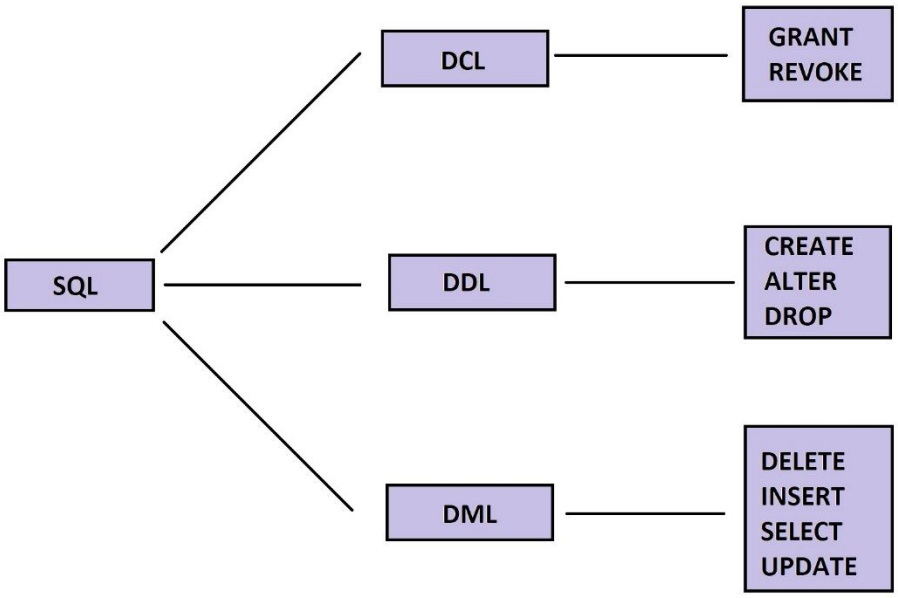
3.2 Λειτουργικότητα του MSSQL

Οι εντολές του SQL χωρίζονται σε 3 κατηγορίες:

- Γλώσσα Ελέγχου Δεδομένων (Data Control Language) – Η Γλώσσα Ελέγχου Δεδομένων περιέχει εντολές που δίνουν δικαιώματα για πρόσβαση και έλεγχο στους χρήστες της βάσης δεδομένων.

- Γλώσσα Ορισμού Δεδομένων (Data Definition Language) – Η Γλώσσα Ορισμού Δεδομένων περιέχει εντολές που δημιουργούν το σχέδιο της βάσης δεδομένων. Είναι οι εντολές που μπορούν να δημιουργήσουν, να αλλάξουν ή να διαγράψουν τη δομή της βάσης (πχ. πίνακες, συναρτήσεις κτλ.)

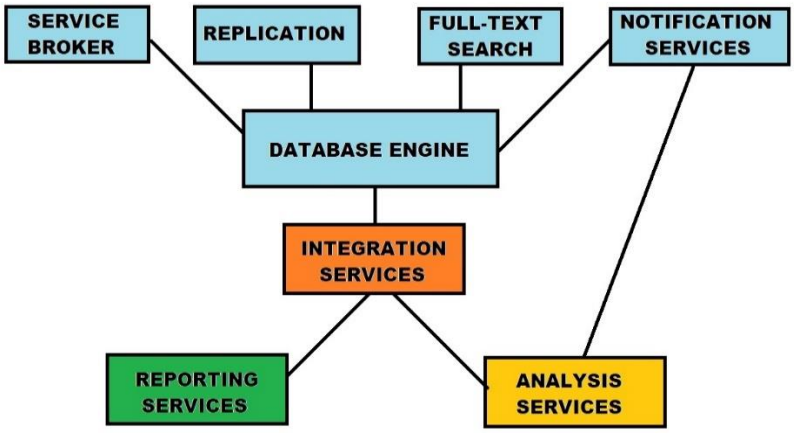
- Γλώσσα Διαχείρισης Δεδομένων (Data Manipulation Language) – Η Γλώσσα Διαχείρισης Δεδομένων περιέχει εντολές που διαχειρίζονται τα δεδομένα της βάσης και έχουν δυνατότητες εισαγωγής, μεταβολής ή διαγράφης αυτών. [5]



Εικόνα 1: Κατηγορίες Εντολών της SQL(Sumanthi, Esakkirajan, 2007)

3.3 Δυνατότητες της SQL

Στην ενότητα που ακολουθεί θα περιγραφούν οι δυνατότητες που έχει η γλώσσα προγραμματισμού SQL. Στο σχήμα που ακολουθεί θα δούμε τις σχέσεις μεταξύ των τμημάτων της.



Εικόνα 2: Τα τμήματα του SQL και οι μεταξύ τους σχέσεις

3.3.1 Database Engine

Η Database Engine είναι η βασική υπηρεσία για αποθήκευση, επεξεργασία και ασφάλεια δεδομένων. Παρέχει ελεγχόμενη πρόσβαση και ταχύτατες επεξεργασία των συναλλαγών, προκειμένου να εξυπηρετεί ακόμα και τις πιο απαιτητικές εφαρμογές της αγοράς. Η Database Engine περιλαμβάνει 2 κυρίες λειτουργίες, την μηχανή αποθήκευσης (storage engine) και τον επεξεργαστή ερωτημάτων (query processor). Υπάρχουν πολλών τύπων SQL Database Engines που έχουν διαφορετική αρχιτεκτονική, αλλά όλες χρησιμοποιούνται για τον ίδιο σκοπό που περιλαμβάνει την δημιουργία, ανάγνωση, ενημέρωση και διαγραφή δεδομένων της βάσης μαζί με πολλές άλλες δυνατότητες.

Η μηχανή αποθήκευσης καταγραφεί και ανακτά τα δεδομένα σε κάποιο φυσικό μέσο (π.χ. σκληρός δίσκος). Ο επεξεργαστής ερωτημάτων είναι υπεύθυνος για την ανάγνωση, έλεγχο σύνταξης και λαθών και εκτέλεση των εντολών και ερωτημάτων του χρήστη. Επίσης ελέγχει αν ο χρήστης που δίνει την εντολή έχει τα απαραίτητα δικαιώματα για την εκτέλεση της. Όταν κάποιος χρήστης υποβάλλει ένα ερώτημα τότε ο επεξεργαστής ερωτημάτων ανατρέχει στο φυσικό αρχείο που είναι αποθηκευμένη η πληροφορία, συνήθως σε .mdf (main directory file) ή .ndf (secondary directory file) τύπου αρχεία, ανακτά τα δεδομένα, τα εμφανίζει ή τα επεξεργάζεται και τα ενημερώνει αντίστοιχα.

3.3.2 Analysis Services

Τα Analysis Services είναι μια μηχανή ανάλυσης δεδομένων που χρησιμοποιείται σε ανάλυση επιχειρηματικών στοιχείων και λήψη αποφάσεων επιχειρηματικότητας. Παρέχει αναλυτικά μοντέλα δεδομένων και εκθέσεις για επιχειρήσεις και εφαρμογές πελατών όπως τα Power BI, Excel, εκθέσεις Reporting Services και άλλες εφαρμογές τρίτων κατασκευαστών. [6]

3.3.3 Reporting Services

Τα Reporting Services είναι ένα φάσμα εργαλείων και υπηρεσιών που δημιουργούν, παραθέτουν και διαχειρίζονται σελιδοποιημένες ή διαδραστικές αναφορές. Παρέχουν τη δυνατότητα για «παραδοσιακές» σελιδοποιημένες αναφορές, διαδραστικές με πίνακες γραφικά ή online σε portal όπου μπορεί οποιοσδήποτε μοντέρνος περιηγητής να έχει πρόσβαση. [7]

3.3.4 Integration Services

Πρόκειται για μια πλατφόρμα οικοδόμησης δεδομένων σε επίπεδο επιχειρήσεων. Η χρήση τους βοηθά στην επίλυση συνθέτων προβλημάτων που αντιμετωπίζουν επιχειρήσεις κάνοντας την πρόσβαση σε δεδομένα και τη χρήση τους ευκολότερη χρησιμοποιώντας τα δεδομένα και αντικείμενα που διαχειρίζεται ο SQL.

3.4 Filestream & αποθήκευση πολυμέσων στη βάση δεδομένων

Η υπηρεσία του Filestream (ροή αρχείων) είναι επιπρόσθετη δυνατότητα του SQL Server η οποία επιτρέπει εφαρμογές που τρέχουν με βάσεις δεδομένων σε SQL Server να αποθηκεύουν μη δομημένα δεδομένα όπως πχ. έντυπα και εικόνες ή στην περίπτωση μας πολυμέσα, τις βιντεοσκοπημένες απαντήσεις των εξεταζόμενων.

Η λειτουργία του Filestream ενσωματώνει την Μηχανή Βάσης Δεδομένων (Database Engine) του SQL Server με συστήματα αρχείων NTFS ή ReFS, αποθηκεύοντας δεδομένα τύπου varbinary(max) δυαδικού μεγάλου αντικειμένου ή BLOB (Binary Large OBject) ως αρχεία στο σύστημα αρχείων. Χρησιμοποιώντας δηλώσεις Transact-SQL (T-SQL) μπορούμε να εισάγουμε, να ενημερώνουμε, να αναζητούμε και να δημιουργούμε αντίγραφα ασφάλειας των δεδομένων Filestream. Στην παρούσα φάση στη δική μας εφαρμογή, χρησιμοποιείται μόνο για αποθήκευση των βιντεοσκοπημένων απαντήσεων των φοιτητών, καθώς και ανάκτηση για να κάνει αναπαραγωγή το επισυναπτόμενο βίντεο ο καθηγητής κατά την διόρθωση των θεμάτων. Ωστόσο, λόγω των πολλών δυνατοτήτων που προσφέρει το Filestream, υπάρχουν πολλά περιθώρια ανάπτυξης και βελτίωσης της εφαρμογής μας. [8]

4 Microsoft Visual Studio, C# & Windows Forms

4.1 Microsoft Visual Studio

Το Microsoft Visual Studio είναι ένα ενσωματωμένο περιβάλλον ανάπτυξης κώδικα (integrated development environment ή IDE) από την Microsoft.

Χρησιμοποιείται για την δημιουργία και ανάπτυξη προγραμμάτων για υπολογιστές, αλλά και για σελίδες, διαδικτυακές εφαρμογές, δικτυακές υπηρεσίες και εφαρμογές κινητών τηλεφώνων. Το Visual Studio χρησιμοποίει πολλές πλατφόρμες ανάπτυξης λογισμικού όπως οι Windows API, Windows Forms, Windows Presentation Foundation, Windows Store και Microsoft Silverlight. Μπορεί να παράγει είτε κώδικα μηχανής (native code) είτε διαχειριζόμενο κώδικα (managed code).

Η αρχιτεκτονική που χρησιμοποιεί το Visual Studio δεν έχει άμεση υποστήριξη για κάποια γλώσσα προγραμματισμού ή κάποιο εργαλείο. Αντιθέτως, επιτρέπει την εισαγωγή λειτουργικού κωδικοποιημένου ως VSPackage. Η πλατφόρμα του Visual Studio ή IDE παρέχει τρείς υπηρεσίες:

- SVsSolution, η οποία απαριθμεί τα έργα και τις λύσεις (projects and solutions) που δημιουργεί ο χρήστης.

- SVsUIShell, που παρέχει το παραθυρικό περιβάλλον του χρήστη (καρτέλες, γραμμές εργαλείων και παράθυρα εργαλείων).

- SVsShell, που ευθύνεται για την εγγραφή των VSPackages.

Στο πρόγραμμα που θα παρουσιάσουμε χρησιμοποιούμε Microsoft Visual Studio 2019. Παρακάτω θα δούμε κάποιες από τις δυνατότητες του εν συντομία.

Δυνατότητες του Microsoft Visual Studio

- Επεξεργαστής κώδικα (Code editor): Το Visual Studio περιλαμβάνει επεξεργαστή κώδικα που υποστηρίζει επισήμανση σύνταξης και συμπλήρωση κώδικα χρησιμοποιώντας το IntelliSense για τις μεταβλητές, συναρτήσεις, μεθόδους, βρόγχους και LINQ (Language Integrated Query) ερωτήματα.

- Εντοπιστής σφαλμάτων (Debugger): Ο εντοπιστής σφαλμάτων του Visual Studio λειτουργεί και σε επίπεδο πηγαίου κώδικα και σε επίπεδο κώδικα μηχανής, δηλαδή γλώσσα χαμηλότερου επιπέδου σε δυαδικό σύστημα.

- Σχεδιαστής (Designer): Ο σχεδιαστής του Visual Studio, συμπεριλαμβάνει πλήθος εικαστικών εργαλείων σχεδίασης όπως Windows Forms Designer, WPF Designer, Web designer/development, Class designer, Mapping designer.

- Επεκτασιμότητα (Extensibility): Το Visual Studio επιτρέπει στους παράγωγους προγραμμάτων να δημιουργούν επεκτάσεις (extensions) για την αναβάθμιση των δυνατοτήτων του. Αυτές οι επεκτάσεις συνδέονται στο Studio και ενισχύουν τη λειτουργικότητα του. Οι επεκτάσεις έρχονται σε μορφές macros, add-ins ή packages.

- Αλλά εργαλεία του Visual Studio: Περιλαμβάνεται ακόμα πληθώρα άλλων εργαλείων στο Visual Studio τα οποία βοηθούν τη χρήση του όπως είναι ο Περιηγητής Ανοιχτών Καρτελών (Open Tabs Browser), ο Επεξεργαστής Ιδιοτήτων (Properties Editor), ο Περιηγητής Αντικειμένων (Object Browser) και αλλά. [9]

Στο πρόγραμμά μας κάνουμε χρήση των Windows Forms του Microsoft Visual Studio καθώς και της C# τα οποία θα αναλύσουμε παρακάτω.

4.2 C# και η χρήσεις της

Η C# είναι μια γλώσσα προγραμματισμού γενικής χρήσης και πολλαπλών παραδειγμάτων (general-purpose, multi-paradigm programming language). Περιλαμβάνει στατική και ισχυρή γραφή κώδικα, λεξικά μεγάλου εύρους, δηλωτική, λειτουργική, γενική, αντικειμενοστραφής (βασισμένη σε κλάσεις) και εξαρτήματα προγραμματισμού.

Η C# δημιουργήθηκε από τον Anders Hejlsberg από τη Microsoft και αργότερα έγινε αποδεκτή σαν διεθνές πρότυπο από την ECMA το 2002 και τον ISO το 2003. Η Microsoft δημοσίευσε την C# μαζί με το .NET Framework και το Visual Studio, τα οποία ήταν προγράμματα κλειστού κώδικα. Μια δεκαετία αργότερα, η Microsoft κυκλοφόρησε το Visual Studio Code (επεξεργαστής κώδικα), Roslyn (μεταγλωττιστής) και την ενωμένη πλατφόρμα του .NET (πλατφόρμα λογισμικού), τα οποία υποστηρίζουν την C# και είναι δωρεάν, ανοικτού κώδικα λογισμικό με συμβατότητα και με άλλες πλατφόρμες προγραμματισμού (cross-platform).

Η σχεδίαση της C# έχει σαν σκοπό από τη δημιουργία της τα παρακάτω:

- Τη δημιουργία μιας γλώσσας απλουστευμένης, μοντέρνας, με γενικό σκοπό σε αντικειμενοστραφή προγραμματισμό.

- Η γλώσσα και οι υλοποιήσεις της πρέπει να παρέχουν υποστήριξη για αρχές μηχανικής λογισμικού (software engineering principles), όπως είναι ο έλεγχος ισχυρής γραφής, έλεγχος ορίων πινάκων, ανίχνευση χρήσης μη αρχικοποιημένων μεταβλητών και αυτόματη συλλογή σκουπιδιών. Η ακεραιότητα, ανθεκτικότητα και παραγωγικότητα του λογισμικού είναι πρωτεύουσας σημασίας για τον προγραμματιστή.

- Η γλώσσα προορίζεται για χρήση σε ανάπτυξη λογισμικών συστημάτων που θα είναι καταλληλά για ενσωμάτωση σε διαμοιραζόμενα περιβάλλοντα.

- Η φορητότητα (portability) είναι πολύ σημαντική για τον πηγαίο κώδικα και τους προγραμματιστές, ειδικά αυτούς που είναι εξοικειωμένοι με την C και C++.

- Η υποστήριξη για διεθνοποίηση είναι πολύ σημαντική.

- Η C# προορίζεται να είναι κατάλληλη για δημιουργία εφαρμογών είτε σε φιλοξενούμενα είτε σε ενσωματωμένα συστήματα (hosted or embedded systems), που κυμαίνονται από τα πολύ μεγάλα που χρησιμοποιούν λειτουργικά συστήματα, μέχρι τα πολύ μικρά που διαθέτουν αποκλειστικές λειτουργίες.

- Τέλος, ενώ οι εφαρμογές γραμμένες σε C# προορίζονταν να εξοικονομούν μνήμη και επεξεργαστικούς πόρους και να έχουν χαμηλότερες προδιαγραφές, δεν ήταν η πρόθεση να ανταγωνιστεί ευθέως σε απόδοση και μέγεθος την C ή την assembly.

Η C# είναι μια γλώσσα εύκολη στη χρήση με πολύ βάθος και δυνατότητες και μπορεί να χρησιμοποιηθεί από πληθώρα εφαρμογών. Στην παρούσα εργασία χρησιμοποιείται λόγω των WinForms και δίνει τη δυνατότητα επικοινωνίας με τη βάση δεδομένων SQL που χρησιμοποιούμε για την αποθήκευση των εξετάσεων μας είτε από καθηγητές είτε από σπουδαστές. [10]

4.3 Windows Forms (WinForms)

Τα Windows Forms γνωστά και ως WinForms είναι μια δωρεάν βιβλιοθήκη γραφικών κλάσεων ανοιχτού κώδικα (open-source graphical class library) που περιλαμβάνεται στο Microsoft .NET, .NET Framework ή Mono Framework, και παρέχει μια πλατφόρμα για γραφή εφαρμογών πελατών για σταθερούς και φορητούς ηλεκτρονικούς υπολογιστές αλλά και για tablet PCs.

Μια εφαρμογή δημιουργημένη με Windows Forms βασίζεται σε συμβάντα υποστηριζόμενα από το .NET Framework της Microsoft. Σε αντίθεση με ένα πρόγραμμα τύπου παρτίδας (batch), καταναλώνει τον περισσότερο χρόνο περιμένοντας μια ενέργεια από το χρήστη όπως το να συμπληρώσει ένα πλαίσιο κειμένου ή να κάνει κλικ σε ένα κουμπί. Ο κώδικας για εφαρμογές σε Windows Forms μπορεί να γραφεί σε γλώσσα προγραμματισμού .NET όπως C# ή Visual Basic.

Τα Windows Forms παρέχουν πρόσβαση σε εγγενή στοιχεία ελέγχου χρήστη (Native Windows User Interface Common Controls) μετασχηματίζοντας τον υπάρχων API των Windows σε διαχειριζόμενο κώδικα.

Όλα τα στοιχεία γραφικών από τη βιβλιοθήκη κλάσεων των Windows Forms προέρχονται από την κλάση Control. Αυτή παρέχει την ελάχιστη λειτουργικότητα ενός στοιχείου διεπαφής χρήστη, όπως τοποθεσία, μέγεθος, χρώμα, γραμματοσειρά, κείμενο, καθώς και κοινότυπα συμβάντα όπως κλικ ή μεταφορά σε άλλο πλαίσιο (drag and drop). Η κλάση Control διαθέτει επίσης υποστήριξη σύνδεσης για να επιτρέψει σε ένα στοιχείο ελέγχου να αναδιατάξει τη θέση του κάτω από το γονέα του. [11] [12]

Στο παρόν project η χρήση των Windows Forms γίνεται για λόγους διευκόλυνσης χρήσης από τους καθηγητές και σπουδαστές, φτιάχνοντας ένα απλουστευμένο περιβάλλον χρήστη (UI) το οποίο μπορεί να εμπλουτιστεί ακόμα περισσότερο, για την εύκολη πρόσβαση και διεξαγωγή των εξετάσεων.

Συμπεριλαμβάνει τη δημιουργία χρηστών για καθηγητές και σπουδαστές, τη δημιουργία, δημοσίευση και ολοκλήρωση διαγωνισμάτων ανά μάθημα για τους καθηγητές καθώς και τη βαθμολόγηση των διαγωνισμάτων.

Αντίστοιχα, οι σπουδαστές μπορούν να δηλώσουν τα μαθήματα που έχουν παρακολουθήσει και να δώσουν την εξέταση διαδικτυακά, με τη δυνατότητα να προσθέσουν και βιντεοσκοπημένες απαντήσεις ολίγων δευτερολέπτων επεξηγώντας την απάντηση που διάλεξαν στο διαγώνισμα. Επίσης έχουν τη δυνατότητα να δουν τη βαθμολογία που έλαβαν στα διαγωνίσματα που πήραν μέρος.

5 Επεξήγηση Κώδικα

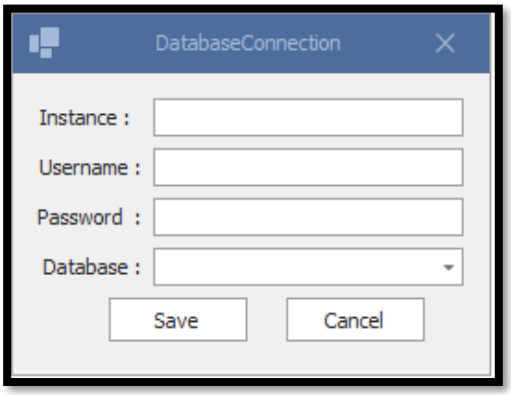
Σε αυτό το κεφάλαιο θα επεξηγήσουμε και θα αναλύσουμε το κομμάτι του κώδικα και τι κάνει αυτό. Επίσης θα δούμε τις διάφορες φόρμες στις οποίες θα έχουν πρόσβαση οι χρήστες του προγράμματος, φοιτητές αλλά και καθηγητές.

Στις παρακάτω ενότητες θα συναντήσουμε ορολογία σε C# που χρησιμοποιείται ξανά και ξανά. Ας διευκρινίσουμε κάποιους από τους όρους για διευκόλυνση της κατανόησης του κώδικα:

- Dictionary <TKey,TValue> Class: Χρησιμοποιείται για την αντικατάσταση ενός κλειδιού με μια τιμή. Στην εφαρμογή μας χρησιμοποιείται για μετάφραση των πεδίων από τη βάση δεδομένων SQL που έρχονται στις φόρμες στα Ελληνικά.
- GetDataIntoTable: Η κλάση αυτή έχει δημιουργηθεί από εμάς ώστε να μπορούμε να ανοίγουμε σύνδεση με την βάση SQL και να κάνουμε εισαγωγή τα δεδομένα που θέλουμε σε ένα DataTable (πίνακάς δεδομένων).
- InitializeComponent(): Είναι η μέθοδος που δημιουργείται αυτόματα και διαχειρίζεται από τα Windows Forms και είναι υπεύθυνη για ότι βλέπουμε στην εκάστοτε φόρμα.
- ColumnsTranslate(): Είναι κλάση που έχει δημιουργηθεί από εμάς ώστε να βοηθά με τη χρήση Dictionary τη μετάφραση ονομάτων στηλών που εμφανίζονται στις φόρμες μας.
- HideColumns(): Είναι κλάση που έχει δημιουργηθεί από εμάς ώστε να κρύβει περιττές στήλες και πληροφορίες που τραβάει ο κώδικας από τη βάση δεδομένων.
- EditOnDoubleClick(): Είναι κλάση που έχει δημιουργηθεί από εμάς ώστε να την καλούμε σε φόρμες οπού θέλουμε να μπορεί να γίνεται και επεξεργασία πατώντας διπλό κλικ.
- GetByID(): Είναι κλάση που έχει δημιουργηθεί από εμάς ώστε να μπορούμε να αντλούμε από τη βάση δεδομένων μας στοιχεία με το ID του αντίστοιχου πεδίου.
- Exists(): Είναι κλάση που έχει δημιουργηθεί από εμάς ώστε να μπορούμε να επιστρέφουμε την τιμή του πεδίου όταν είναι True.
- Fill(): Είναι κλάση που έχει δημιουργηθεί από εμάς ώστε να μπορεί να συμπληρώνει τα πεδία μιας φόρμας ελέγχοντας πρώτα τον τύπο του πεδίου και τον τύπο των δεδομένων που εισάγονται.
- Save(): Είναι κλάση που έχει δημιουργηθεί από εμάς ώστε να αποθηκεύει τα δεδομένα που εισάγουμε στην βάση δεδομένων SQL.

5.1 Σύνδεση στη Βάση Δεδομένων (Database Connection)

Εδώ γίνεται η σύνδεση με τη βάση δεδομένων μας , όπου το πρόγραμμα τραβάει τα στοιχεία σύνδεσης SQL από το αρχείο μορφής .config όπου τα έχουμε καταχωρήσει και συνδέεται με τον SQL Server και τη βάση που έχουμε ορίσει. Σε περίπτωση που το αρχείο .config έχει καταχωρημένα λάθος στοιχεία μας βγάζει τη φόρμα του database connection για να τα καταχωρήσουμε όπως θα δούμε παρακάτω.



Εικόνα 3: Φόρμα Σύνδεσης στη Βάση Δεδομένων (Database Connection Form)

Η φόρμα έχει τα πεδία Instance, Username, Password και Database.

- Instance : Το SQL Server Instance που έχουμε εγκατεστημένο στον υπολογιστή με τον οποίο πρέπει να γίνει η σύνδεση. Επίσης πρέπει να καταχωρήσουμε και το όνομα ή ip του υπολογιστή πριν την ανάποδη κάθετο (\) και το όνομα το Instance του SQL. Η σωστή καταχώρηση είναι ServerName\SQLInstanceName.

- Username: Το όνομα χρήστη που έχουμε δημιουργήσει για τον SQL και για την πρόσβαση σε αυτόν. Στο παράδειγμα γίνεται χρήση του sa (system administrator) χρήστη.

- Password: Ο κωδικός πρόσβασης του χρήστη του SQL με τον οποίο θέλουμε να συνδεθούμε.

- Database: Η βάση δεδομένων με την οποία επιθυμούμε να συνδεθούμε. Έχουμε θέσει από τον κώδικα στο dropdown list να μην αναφέρονται οι πρότυπες βάσεις του SQL (master,tempdb,model,msdb).

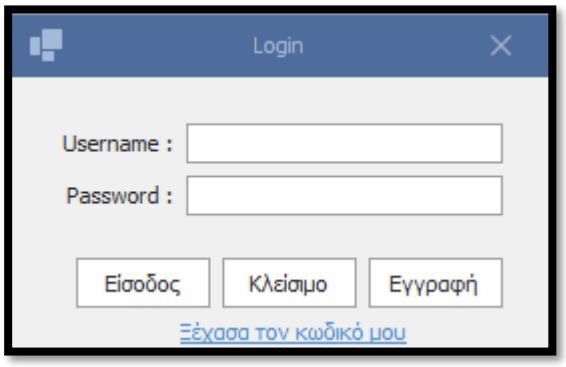
Παρακάτω θα παραθέσουμε τον κώδικα της φόρμας:

```
namespace Thanos.DatabaseConnection {
public partial class DatabaseConnectionForm : DevExpress.XtraBars.Ribbon.RibbonForm {
public DatabaseConnectionForm() {
   InitializeComponent();
   teInstance.Text = db.Instance;
   teUsername.Text = db.Username;
   tePassword.Text = db.Password;
   cbDataBase.Text = db.Database;
}
DataBaseActions db = MainForm.db;
Messages msgs = new Messages();
private void cbDataBase_Enter(object sender, EventArgs e) {
try {
   SqlConnection con = new SqlConnection("Data Source = " + teInstance.Text + "; Initial Catalog = " + cbDataBase.Text + ";
   User ID = " + teUsername.Text + "; Password=" + tePassword.Text + ";");
   SqlCommand cmd = new SqlCommand("select name from sys.databases where name NOT IN
   ('master','tempdb','model','msdb')",con);
   con.Open();
   SqlDataAdapter da = new SqlDataAdapter(cmd);
   DataTable dt = new DataTable();
   da.Fill(dt);
   con.Close();
   foreach (DataRow dr in dt.Rows) {
      cbDataBase.Properties.Items.Add(dr[0].ToString());
   }
   cbDataBase.SelectedIndex = 0;
} catch (Exception ex){
   cbDataBase.Properties.Items.Clear();
   cbDataBase.Text = "";
}
}
private void btnSave_Click(object sender, EventArgs e) {
try {
   SqlConnection con = new SqlConnection("Data Source = " + teInstance.Text + "; Initial Catalog = " + cbDataBase.Text + ";
   User ID = " + teUsername.Text + "; Password=" + tePassword.Text + ";");
   con.Open();
   con.Close();
   db.SaveConfigFile(teInstance.Text, teUsername.Text, tePassword.Text, cbDataBase.Text);
   this.Close();
} catch (Exception ex) { msgs.Error(ex.Message); }
}
private void btnCancel_Click(object sender, EventArgs e) {
   this.Close();
}
private void DatabaseConnectionForm_Load(object sender, EventArgs e)
{
}
}
}
```

Αφού καταχωρήσουμε τα σωστά στοιχεία τα αποθηκεύει στο αρχείο config και μας πηγαίνει στο επόμενο βήμα:

5.2 Είσοδος/Εγγραφή Χρηστών (Log in/Sign Up)

Σε αυτό το βήμα ο κώδικας έχει συνδεθεί με τη βάση δεδομένων και μας βγάζει τη φόρμα για εισαγωγής στοιχείων σύνδεσης των χρηστών (καθηγητών/φοιτητών) καθώς και την επιλογή δημιουργίας νέου λογαριασμού. Υπάρχει επίσης η δυνατότητα ανάκτησης του κωδικού μέσω mail.



Εικόνα 4: Φόρμα Σύνδεσης Χρήστη (Login Form)

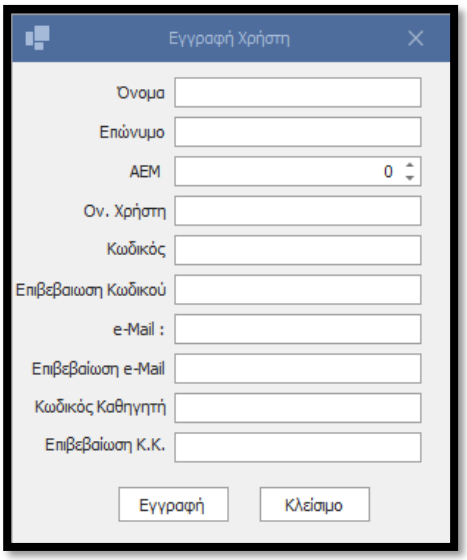
Ελέγχει το username και το password που εισάγει ο χρήστης και προχωράει στην αντίστοιχη φόρμα αναλόγως τα στοιχεία που εισάγει ο χρήστης (καθηγητής ή σπουδαστής). Εάν το username είναι λανθασμένο βγάζει προειδοποιητικό μήνυμα “User does not exist”, εάν μόνο ο κωδικός είναι λανθασμένος βγάζει προειδοποιητικό μήνυμα “Password is incorrect for this user”.

Ακολουθεί το κομμάτι του κώδικα για τη φόρμα εισόδου Login:

```
namespace Thanos.LogIn {
public partial class LogInForm : DevExpress.XtraBars.Ribbon.RibbonForm {
public LogInForm(Users obj) {
   InitializeComponent();
   if (!db.TestConnection()) {
      DatabaseConnectionForm nformDbCon = new DatabaseConnectionForm();
      nformDbCon.ShowDialog(this);
   }
   if (obj != null) {
      user = obj;
   } else {
      user = new Users();
   }
}
Users user;
DataBaseActions db = new DataBaseActions();
Messages msgs = new Messages();
private void LogInForm_Load(object sender, EventArgs e) {
}
private void btnLogIn_Click(object sender, EventArgs e) {
db = new DataBaseActions();
   try {
      try {
         user.ID = Guid.Parse(db.GetDataIntoTable("select ID from users where username ='" + teUserName.Text +
          "'").Rows[0][0].ToString());
      }catch { msgs.Error("User does not exists");
         return;
      }
      user.GetByID(user.ID);
      if (user.UserName.ToUpper() == teUserName.Text.ToUpper() && user.uPassword.Decrypt() == tePassword.Text) {
         this.Hide();
```

```
            MainForm nform = new MainForm(user);
            nform.ShowDialog(this);
        } else {
            msgs.Error("Password is incorrect for this user");
        }
    } catch (Exception ex){ msgs.Error(ex.Message); Application.Exit(); }
}
private void hlPasswordForget_Click(object sender, EventArgs e) {
    PasswordForgetForm nform = new PasswordForgetForm();
    nform.ShowDialog(this);
}
private void btnSingIN_Click(object sender, EventArgs e) {
    SingInForm nform = new SingInForm();
    nform.ShowDialog(this);
}
private void btnCancel_Click(object sender, EventArgs e) {
    this.Close();
}
}
}
```

Αν θέλουμε τη δημιουργία νέου χρήστη τότε επιλέγοντας το κουμπί Sign Up μπορούμε να κάνουμε εγγραφή. Πατώντας το εμφανίζεται η παρακάτω φόρμα:



Εικόνα 5: Φόρμα Εγγραφής Χρήστη (Sign Up Form)

Εδώ καταχωρούμε Όνομα, Επώνυμο, Αριθμό Ειδικού Μητρώου, Όνομα και Κωδικό χρήστη, καθώς και mail. Υπάρχει επιβεβαίωση κωδικού χρήστη και mail στη φόρμα για επιπλέον ασφάλεια. Επίσης υπάρχει το πεδίο teacher's code, όπου πρόκειται για ειδικό κωδικό που δίνεται στους καθηγητές ώστε η δημιουργία χρήστη να είναι ορισμένη σαν καθηγητής.

```
namespace Thanos.LogIn {
public partial class SingInForm : DevExpress.XtraBars.Ribbon.RibbonForm {
```

```
public SingInForm() {
   InitializeComponent();
}
Messages msgs = new Messages();
Users user = new Users();
private void btnCancel_Click(object sender, EventArgs e) {
   this.Close();
}
private void btnCreate_Click(object sender, EventArgs e) {
   try {
      string Errors = string.Empty;
      this.Fill(user);
      if (string.IsNullOrEmpty(teName.Text)) {
         Errors += "Field Name is mandatory \n";
      }
      if (string.IsNullOrEmpty(teSurname.Text)) {
         Errors += "Field Surname is mandatory \n";
      }
      if (tePassword.Text != teConfPassword.Text) {
         Errors += "Password does not match \n";
      }
      if (teeMail.Text != teConfeMail.Text) {
         Errors += "e-Mail does not match \n";
      }
      if (teTeachersCode.Text != teConfTeachersCode.Text) {
         Errors += "Teachers Code does not match\n";
      }
      user.isTeacher = teTeachersCode.Text == "e63336d2" ? true : false;
      if (!user.isTeacher && spAEM.Value == 0) {
         Errors += "AEM is mandatory\n";
      }else if (user.isTeacher) {
         user.AEM = null;
      }
      if (!string.IsNullOrEmpty(Errors)) {
         msgs.Error(Errors);
      } else {
         user.uPassword = user.uPassword.Encrypt();
         user.Save();
         this.Close();
      }
   }catch(Exception ex) { msgs.Error(ex.Message); }
}
private void SingInForm_Load(object sender, EventArgs e)
{
}
private void label8_Click(object sender, EventArgs e)
{
}
private void label2_Click(object sender, EventArgs e)
{
}
}
}
}
```

Στον κώδικα της φόρμας βλέπουμε τα μηνύματα σφαλμάτων που βγάζει σε περίπτωση λάθος καταχώρησης του αντίστοιχου πεδίου. Επίσης εδώ βλέπουμε και τον ειδικό κωδικό δημιουργίας λογαριασμού για καθηγητές. Στο παράδειγμα έχει τεθεί ο κωδικός **"e63336d2"**. Καταχωρώντας τον παραπάνω κωδικό στο πεδίο Κωδικός Καθηγητή και στο πεδίο επιβεβαίωσης από κάτω δίνει δικαίωμα στο χρήστη που δημιουργείται να μπαίνει σαν καθηγητής. Αν το πεδίο παραμείνει κενό τότε ο χρήστης που δημιουργείται κάνει είσοδο σαν φοιτητής.

5.3 Κυρίως φόρμα Καθηγητών/Φοιτητών (Main Form Teachers/Students)

Παρακάτω, θα δούμε τις φόρμες των καθηγητών και τις επιλογές που τους παρέχονται για να κάνουν από το πρόγραμμα.

Εικόνα 6: Κυρίως Φόρμα (Main Form)

Όπως βλέπουμε στη φόρμα ο καθηγητής έχει τη δυνατότητα να αποθηκεύσει μαθήματα στα οποία οι φοιτητές θα εξεταστούν καθώς και το ποτέ θα γίνει η εξέταση και η περίοδος δήλωσης για την εξέταση, να δημιουργήσει διαγωνίσματα και να πάρει Reports για τους μαθητές όπως πχ. Βαθμολόγηση.

Τώρα θα παραθέσουμε τον κώδικα της κυρίως φόρμας για τους καθηγητές και φοιτητές. Η παρούσα φόρμα χρησιμοποιεί το ίδιο template και για τους 2 με άλλες επιλογές και δυνατότητες αναλόγως τον τύπο του χρήστη που κάνει είσοδο στην εφαρμογή. Όπως βλέπουμε στην αρχή του κώδικα ελέγχει τον χρήστη (Users) και κάνει ορατά τα αντίστοιχα controls της φόρμας.

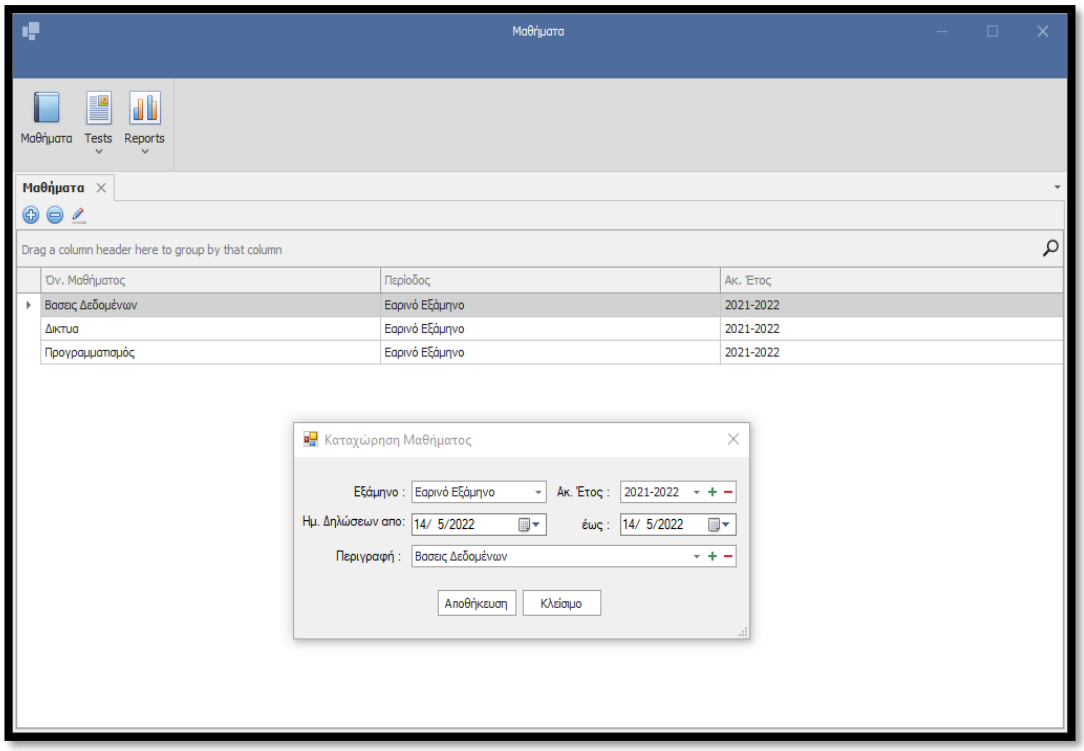
Έχουμε βάλει έλεγχο ώστε να μην ανοίγει 2^{η} φορά κάποια από τις φόρμες από τις επιλογές που έχουν διαθέσιμες οι χρήστες αλλά να παραμένει στην υπάρχουσα που ο χρήστης έχει ήδη ανοιχτή.

```
namespace Thanos {
public partial class MainForm : DevExpress.XtraBars.Ribbon.RibbonForm {
public MainForm(Users obj) {
  InitializeComponent();
  if (obj.isTeacher) {
    rbpgStudents.Visible = false;

  } else {
    rbpgTeachers.Visible = false;
  }
  user = obj;
}
public static Users user;
public static DataBaseActions db = new DataBaseActions();
Messages msg = new Messages();
private bool ActivateFormIfExists(Type FormType) {
  bool Exists = false;
  try {
    foreach (Form form in Application.OpenForms) {
      if (form.GetType() == FormType) {
        form.Activate();
        Exists = true;
        return Exists;
      }
    }
    return Exists;
  } catch (Exception ex) { throw; return Exists; }
}
private void btnCourse_ItemClick(object sender, DevExpress.XtraBars.ItemClickEventArgs e) {
  try {
    if(!ActivateFormIfExists(typeof(CoursesReportForm))) {
      CoursesReportForm nform = new CoursesReportForm(user);
      nform.MdiParent = this;
      nform.Show();
    }
  } catch (Exception ex) { msg.Error(ex.Message); }
}
private void barButtonItem4_ItemClick(object sender, DevExpress.XtraBars.ItemClickEventArgs e) {
  try {
    if (!ActivateFormIfExists(typeof(QuestionsReportForm))) {
      QuestionsReportForm nform = new QuestionsReportForm(user);
      nform.MdiParent = this;
      nform.Show();
    }
  } catch (Exception ex) { msg.Error(ex.Message); }
}
private void barButtonItem5_ItemClick(object sender, DevExpress.XtraBars.ItemClickEventArgs e) {
  try {
    if (!ActivateFormIfExists(typeof(TestsReportForm))) {
      TestsReportForm nform = new TestsReportForm(user);
      nform.MdiParent = this;
      nform.Show();
    }
  } catch (Exception ex) { msg.Error(ex.Message); }
}
private void MainForm_FormClosed(object sender, FormClosedEventArgs e) {
  Application.Exit();
}
private void barButtonItem6_ItemClick(object sender, DevExpress.XtraBars.ItemClickEventArgs e) {
  try {
    if (!ActivateFormIfExists(typeof(StudentsCoursesReportForm))) {
      StudentsCoursesReportForm nform = new StudentsCoursesReportForm();
```

```
            nform.MdiParent = this;
            nform.Show();
        }
    } catch (Exception ex) { msg.Error(ex.Message); }
}
private void barButtonItem7_ItemClick(object sender, DevExpress.XtraBars.ItemClickEventArgs e) {
    try {
        if (!ActivateFormIfExists(typeof(CompletedTestReportForm))) {
            CompletedTestReportForm nform = new CompletedTestReportForm();
            nform.MdiParent = this;
            nform.Show();
        }
    } catch (Exception ex) { msg.Error(ex.Message); }
}
private void barButtonItem8_ItemClick(object sender, DevExpress.XtraBars.ItemClickEventArgs e) {
    try {
        if (!ActivateFormIfExists(typeof(TestParticipationReportForm))) {
            TestParticipationReportForm nform = new TestParticipationReportForm();
            nform.MdiParent = this;
            nform.Show();
        }
    } catch (Exception ex) { msg.Error(ex.Message); }
}
private void barButtonItem3_ItemClick(object sender, DevExpress.XtraBars.ItemClickEventArgs e) {
    try {
        if (!ActivateFormIfExists(typeof(EevaluationReportForm))) {
            EevaluationReportForm nform = new EevaluationReportForm();
            nform.MdiParent = this;
            nform.Show();
        }
    } catch (Exception ex) { msg.Error(ex.Message); }
}
private void MainForm_Load(object sender, EventArgs e)
{
}
}
}
```

5.3.1 Φόρμες Καθηγητών

Σε αυτή την ενότητα θα δούμε τη φόρμα δήλωσης μαθημάτων προς εξέταση από τον καθηγητή:



Εικόνα 7: Φόρμα Δημιουργίας Μαθήματος από Καθηγητή

Αυτή είναι η φόρμα για εισαγωγή μαθήματος που διδάσκει ο καθηγητής και θα εξεταστούν οι σπουδαστές που έχουν δηλώσει το μάθημα. Ο καθηγητής δηλώνει το Εξάμηνο (Χειμερινό/Εαρινό) και το Ακαδημαϊκό Έτος (πχ. 2021-2022), την περίοδο δηλώσεων , από ποτέ δηλαδή ανοίγουν οι δηλώσεις για το συγκεκριμένο μάθημα μέχρι ποτέ είναι η τελευταία μέρα που μπορούν οι σπουδαστές να το δηλώσουν. Στην περιγραφή μπορεί είτε να επιλέξει από τα ήδη υπάρχοντα μαθήματα που έχει δημιουργήσει ο ίδιος, ή να εισάγει ένα νέο μάθημα πατώντας το κουμπί εισαγωγής (+) διπλά από τη λίστα.

Παρακάτω παρατίθεται ο κώδικας της φόρμας όπου μπορούμε να δούμε τη δημιουργία της φόρμας καθώς και τη μετάφραση των ονομάτων των στηλών που τραβάει από τη βάση δεδομένων ώστε να έρχονται στα ελληνικά, καθώς και το ότι είναι μόνο για ανάγνωση (read only) και δεν μπορούμε να επεξεργαστούμε τα ονόματα (non editable).

Ο κώδικας αντλεί και φέρνει στη λίστα που βλέπουμε από πίσω όλες τις ήδη υπάρχουσες εγγραφές από τους πίνακες της βάσης δεδομένων SQL μας με ένα απλό join των 4 πινάκων που χρησιμοποιούμε Courses, Coursename, eAcademicYears, ePeriods.

```
namespace Thanos.Reports {
public partial class CoursesReportForm : Form {
public CoursesReportForm(Users obj) {
   InitializeComponent();
   user = obj;
   GridRefresh();
   FixGridView();
}
Users user;
DataBaseActions db = MainForm.db;
Messages msg = new Messages();
private void FixGridView() {
   Dictionary<string, string> Translate = new Dictionary<string, string>();
   Translate.Add("Name", "Όν. Μαθήματος");
   Translate.Add("AcademicYear", "Ακ. Έτος");
   Translate.Add("PeriodName", "Περίοδος");
   gvCourses.ColumnsTranslate(Translate);
   gvCourses.HideColumns();
   gvCourses.OptionsBehavior.ReadOnly = true;
   gvCourses.OptionsBehavior.Editable = false;
}
private void GridRefresh(){
   gcCoures.DataSource = db.GetDataIntoTable(" SELECT " +
                                 "    c.ID" +
                                 "    ,cn.Name" +
                                 "    ,PeriodName" +
                                 "    ,ay.AcademicYear" +
                                 " FROM courses c" +
                                 " JOIN CourseName cn" +
                                 "    ON cn.ID = c.eCourseNameID" +
                                 " JOIN eAcademicYears ay" +
                                 "    on ay.ID = c.eAcademicYearID"+
                                 " JOIN eperiods p" +
                                 "    ON c.eperiodsid = p.id " +
                                 "    where c.eUserID = '" + user.ID + "'");
}
private void btnAdd_Click(object sender, EventArgs e) {
   try {
      Courses course = new Courses();
      CoursesEntryForm frm = new CoursesEntryForm(course);
      frm.ShowDialog(this);
      GridRefresh();
   } catch (Exception ex) { msg.Error(ex.Message); }
}
private void btnDelete_Click(object sender, EventArgs e) {
   try {
      gvCourses.DeleteSelectedRow(typeof(Courses), GridRefresh);
   } catch (Exception ex) { msg.Error(ex.Message); }
}
private void btnEdit_Click(object sender, EventArgs e) {
   try {
      gvCourses.EditOnDoubleClick(this, typeof(Courses), typeof(CoursesEntryForm), GridRefresh);
   } catch (Exception ex) { msg.Error(ex.Message); }
}
private void gvCourses_DoubleClick(object sender, EventArgs e) {
   try {
      gvCourses.EditOnDoubleClick(this, typeof(Courses), typeof(CoursesEntryForm), GridRefresh);
   } catch (Exception ex) { msg.Error(ex.Message); }
}
```

```
private void CoursesReportForm_Load(object sender, EventArgs e)
{
}
}
}
```

Στο επόμενο κομμάτι του κώδικα βλέπουμε τα buttons που υπάρχουν στη φόρμα (add, delete και edit) και τι κάνουν.

- Add (+): Προσθέτει νέο μάθημα. Καλεί τη φόρμα εισαγωγής μαθήματος που βλέπουμε και παραπάνω ώστε να συμπληρώσει τα στοιχεία μαθήματος ο καθηγητής.

- Delete (-): Διαγραφή ενός υπάρχοντος μαθήματος.

- Edit: Επεξεργασίας υπάρχουσας εγγραφής μαθήματος για αλλαγή ή διόρθωση στοιχείων.

Παρακάτω θα παραθέσουμε τον κώδικα της φόρμας εισαγωγής μαθημάτων.

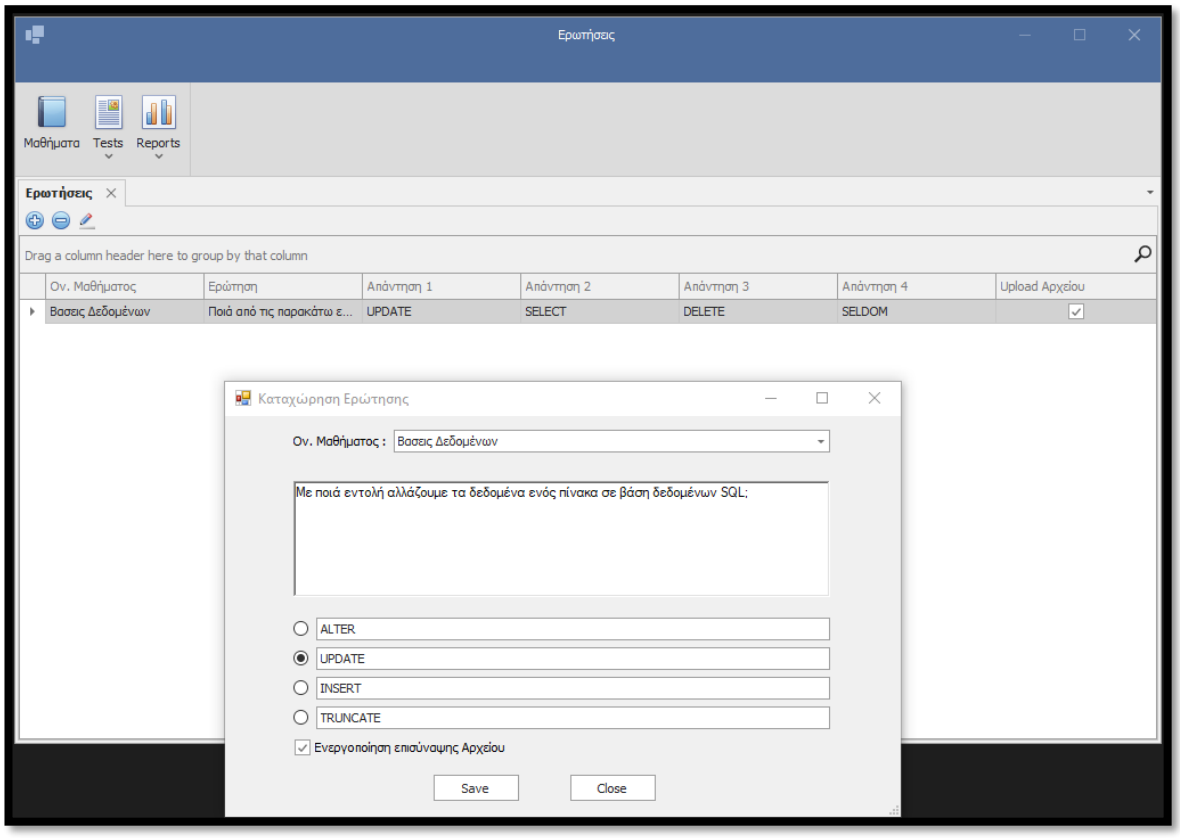
```
namespace Thanos.EntriesForms {
public partial class CoursesEntryForm : Form {
public CoursesEntryForm(Courses obj) {
   InitializeComponent();
   course = obj;
   user = MainForm.user;
   FixLookUpEdit1();
   FixLookUpEdit2();
   FixLookUpAcademicYear();
   if (obj.ePeriodsID != Guid.Empty) {
      course.Fill(this);
   }

}
Users user;
DataBaseActions database = MainForm.db;
DataTable dt = new DataTable();
DataTable dt2 = new DataTable();
DataTable dt3 = new DataTable();
Courses course;
Messages msg = new Messages();
private void FixLookUpEdit1() {
   dt = database.GetDataIntoTable("select * from ePeriods order by PeriodName");
   lookUpEdit1.SetLookUpEdit(dt, "PeriodName", "ID");
   lookUpEdit1.Refresh();
}
private void FixLookUpEdit2() {
   dt2 = database.GetDataIntoTable("select * from CourseName order by Name");
   lookUpEdit2.SetLookUpEdit(dt2, "Name", "ID", null, true, true, false, true, LookUpEditButton_Click);
   lookUpEdit2.Refresh();
}
private void FixLookUpAcademicYear() {
   dt3 = database.GetDataIntoTable("select * from eAcademicYears order by right(AcademicYear,4) desc");
```

```
    luAcademicYear.SetLookUpEdit(dt3, "AcademicYear", "ID", null, true, true, false, true, LookUpEditAcademicYears_Click);
    luAcademicYear.Refresh();
}
private void LookUpEditAcademicYears_Click(object sender, EventArgs e) {
    string Tag = (sender as DevExpress.XtraEditors.Controls.EditorButton).Tag.ToString();
    try {
        eAcademicYears cName = new eAcademicYears();
        switch (Tag) {
            case "Delete":
                DialogResult dr = msg.QuestionYesNo("Θέλετε να διαγράψετε την εγγραφή '" + luAcademicYear.Text + "';");
                if (dr == DialogResult.Yes) {
                    cName.GetByID(Guid.Parse(luAcademicYear.EditValue.ToString()));
                    cName.DeleteByID();
                    FixLookUpAcademicYear();
                    luAcademicYear.EditValue = dt3.Rows[0]["ID"];
                }
                break;
            case "Add":
                AcademicYearEntryForm nform = new AcademicYearEntryForm(cName);
                nform.ShowDialog(this);
                FixLookUpAcademicYear();
                luAcademicYear.EditValue = cName.ID;
                break;
        }
    } catch (Exception ex) { msg.Error(ex.Message); }
}
private void LookUpEditButton_Click(object sender, EventArgs e) {
    string Tag = (sender as DevExpress.XtraEditors.Controls.EditorButton).Tag.ToString();
    try {
        CourseName cName = new CourseName();
        switch (Tag) {
            case "Delete":
                DialogResult dr = msg.QuestionYesNo("Θέλετε να διαγράψετε την εγγραφή '" + lookUpEdit2.Text + "';");
                if (dr == DialogResult.Yes) {
                    cName.GetByID(Guid.Parse(lookUpEdit2.EditValue.ToString()));
                    cName.DeleteByID();
                    FixLookUpEdit2();
                    lookUpEdit2.EditValue = dt2.Rows[0]["ID"];
                }
                break;
            case "Add":
                NewCourseNameEntryForm nform = new NewCourseNameEntryForm(cName);
                nform.ShowDialog(this);
                FixLookUpEdit2();
                lookUpEdit2.EditValue = cName.ID;
                break;
        }
    } catch (Exception ex) { msg.Error(ex.Message); }
}
private void btnSave_Click(object sender, EventArgs e) {
    try {
        this.Fill(course);
        course.eUserID = user.ID;
        course.Save();
        this.Close();
    } catch (Exception ex) { msg.Error(ex.Message); }
}

private void btnClose_Click(object sender, EventArgs e) {
    this.Close();
}
private void CoursesEntryForm_Load(object sender, EventArgs e)
```

```
{

}
}
}
```

Στον παραπάνω κώδικα βλέπουμε τη δημιουργία των 3 λιστών για τις περιόδους (DataTable dt), για το όνομα μαθήματος (DataTable dt2) και για το ακαδημαϊκό έτος (DataTable dt3). Στα επόμενα 3 βήματα βλέπουμε ότι ο κώδικας τραβάει και από τη βάση δεδομένων τα αντίστοιχα πεδία που υπάρχουν καταχωρημένα και τα φέρνει στη λίστα. Στην περίπτωση των πεδίων Ακαδημαϊκού Έτους και Μαθήματος μας επιτρέπει να προσθέσουμε νέο έτος/μάθημα ή να διαγράψουμε κάποιο από τα υπάρχοντα. Έπειτα βλέπουμε πως τρέχει την επεξεργασία (edit) υπάρχοντος ακαδημαϊκού έτους ή μαθήματος στα επόμενα βήματα αντίστοιχα. Βγάζει ξανά τη φόρμα καταχώρησης μαθήματος από την οποία μας επιτρέπει να αλλάξουμε τα στοιχεία που έχει το μάθημα που επιλέξαμε. Η ημερομηνία έναρξης και λήξης δηλώσεων μαθήματος αποθηκεύεται στη βάση δεδομένων SQL στον πίνακα Courses.

Στην επόμενη ενότητα θα δούμε τη φόρμα και τον κώδικα της επιλογής δημιουργίας νέου ερωτήματος από τον καθηγητή. Όταν ο καθηγητής επιλέγει από το dropdown list των Tests την επιλογή Ερωτήσεις η παρακάτω φόρμα ανοίγει.



Εικόνα 8: Φόρμα Δημιουργίας Ερωτημάτων για Διαγωνίσματα από Καθηγητή

Στην προκειμένη περίπτωση έχουμε επιλέξει και την επιλογή προσθήκης ερωτήματος ώστε να κάνουμε καταχώρηση μιας νέας ερωτήσεις που θα μπορεί να χρησιμοποιηθεί σε μελλοντικό διαγώνισμα. Επίσης, βλέπουμε ήδη υπάρχουσα ερώτηση που έχει δημιουργηθεί στην φόρμα από πίσω. Ας δούμε τώρα πως λειτουργεί και ο κώδικας της παρούσας φόρμας:

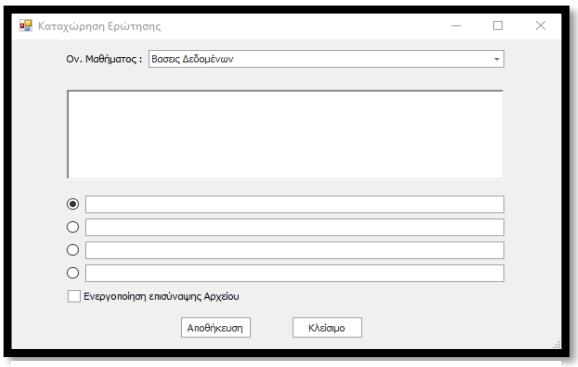
```
namespace Thanos.Reports {
public partial class QuestionsReportForm : Form {
public QuestionsReportForm(Users usr) {
   InitializeComponent();
   user = usr;
   GridRefresh();
   FixGridView();
}
Users user;
DataBaseActions db = MainForm.db;
Messages msgs = new Messages();
private void GridRefresh() {
   gcQuestions.DataSource = db.GetDataIntoTable("SELECT " +
                                    "   q.ID " +
                                    "   ,Name " +
                                    "   ,mainQuestion " +
                                     "   ,Answer1 " +
                                    "   ,Answer2 " +
                                    "   ,Answer3 " +
                                    "   ,Answer4 " +
                                    "   ,AllowUploadFile " +
                                    " FROM dbo.Questions q "+
                                    " JOIN dbo.CourseName cn " +
                                    "      ON cn.ID = q.eCourseNameID "+
                                    " where eUserID = '" + user.ID + "'");
}
private void btnAdd_Click(object sender, EventArgs e) {
   Questions question = new Questions();
   QuestionsEntryForm nform = new QuestionsEntryForm(question);
   nform.ShowDialog(this);
   GridRefresh();
}
private void FixGridView(){
   Dictionary<string, string> Translate = new Dictionary<string, string>();
   Translate.Add("Name", "Ον. Μαθήματος");
   Translate.Add("mainQuestion", "Ερώτηση");
   Translate.Add("Answer1", "Απάντηση 1");
   Translate.Add("Answer2", "Απάντηση 2");
   Translate.Add("Answer3", "Απάντηση 3");
   Translate.Add("Answer4", "Απάντηση 4");
   Translate.Add("AllowUploadFile", "Upload Αρχείου");
   gvQuestions.ColumnsTranslate(Translate);
   gvQuestions.HideColumns();
   gvQuestions.OptionsBehavior.ReadOnly = true;
   gvQuestions.OptionsBehavior.Editable = false;
}
private void btnDelete_Click(object sender, EventArgs e) {
   try {
      gvQuestions.DeleteSelectedRow(typeof(Questions), GridRefresh);
   } catch (Exception ex) { msgs.Error(ex.Message); }
}
private void btnEdit_Click(object sender, EventArgs e) {
   try {
```

```
        gvQuestions.EditOnDoubleClick(this, typeof(Questions), typeof(QuestionsEntryForm), GridRefresh);
    } catch (Exception ex) { msgs.Error(ex.Message); }
}
private void gvQuestions_DoubleClick(object sender, EventArgs e) {
    try {
        gvQuestions.EditOnDoubleClick(this, typeof(Questions), typeof(QuestionsEntryForm), GridRefresh);
    }catch(Exception ex) { msgs.Error(ex.Message); }
}
private void panel2_Paint(object sender, PaintEventArgs e)
{
}
}
}
```

Βλέπουμε ότι δημιουργεί μια σύνδεση με τη βάση δεδομένων μας και περνάει στο grid της φόρμας (gcQuestions) με script στον SQL κάνοντας select ο τον πίνακα Questions και join τον πίνακα CourseName τα αντίστοιχα πεδία που χρειάζονται (ID, name, mainQuestion, Answer1/2/3/4 και την επιλογή AllowUploadFile). Παρακάτω, έχει και τον κώδικα του κουμπιού Add (+) όπου φέρνει την φόρμα εισαγωγής νέου ερωτήματος που βλέπουμε στην παραπάνω εικόνα της φόρμας.

Έχουμε μεταφράσει τους πίνακες από τη βάση δεδομένων στα ελληνικά με τη χρήση λεξικού Dictionary. Προσθέσαμε τα λεκτικά της φόρμας που από τη βάση δεδομένων έρχονται σε Αγγλικά και τα μεταφράσαμε σε Ελληνικά. Τα πεδία ονομασίας των στηλών είναι read only και non editable. Κατόπιν, έχουμε τα κουμπιά delete, edit καθώς και τη δυνατότητα να κάνουμε edit κάνοντας double click σε κάποια ήδη υπάρχουσα ερώτηση. Όταν επιλέγουμε να κάνουμε edit η φόρμα εισαγωγής ξαναβγαίνει με τα στοιχεία του ερωτήματος που θέλουμε να μεταβάλλουμε.

Παρακάτω θα παραθέσουμε και τον κώδικα που τρέχει κατά την εκτέλεση της φόρμας εισαγωγής ερωτήματος.



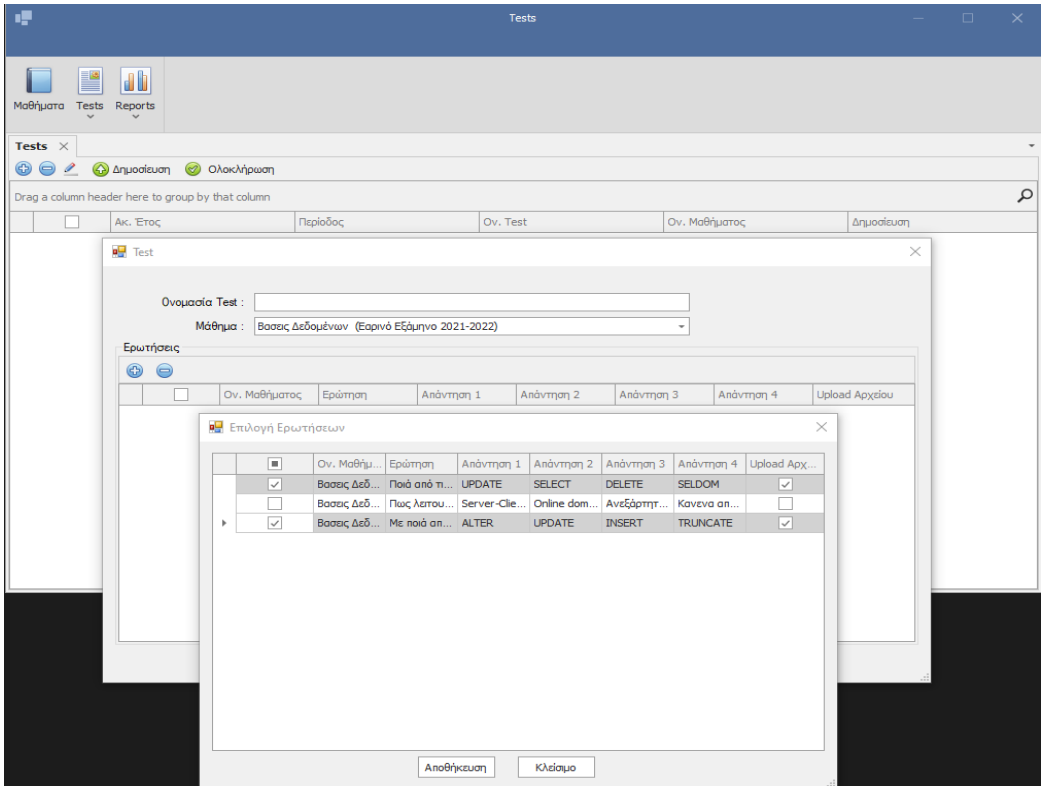
Εικόνα 9: Φόρμα Δημιουργίας Ερωτήματος

```
namespace Thanos.EntriesForms {
public partial class QuestionsEntryForm : Form {
public QuestionsEntryForm(Questions obj) {
   InitializeComponent();
   user = MainForm.user;
   question = obj;
   if (question.eUserID != Guid.Empty){
      question.Fill(this);
   }
   FixLookUpEditCourseName();
}
DataBaseActions db = MainForm.db;
DataTable dt = new DataTable();
Questions question = new Questions();
Messages msg = new Messages();
Users user;

private void FixLookUpEditCourseName() {
   dt = db.GetDataIntoTable("SELECT " +
                     " cn.* FROM dbo.CourseName cn " +
                     " JOIN Courses c ON cn.ID = c.eCourseNameID " +
                     " WHERE eUserID = '"+user.ID+"'");
   luCourseName.SetLookUpEdit(dt, "Name", "ID");
   luCourseName.Refresh();
}
private void btnSave_Click(object sender, EventArgs e) {
   try {
      this.Fill(question);
      question.eUserID = user.ID;
      question.Save();
      msg.Information("Οι αλλαγές αποθηκεύτηκαν επιτυχώς!");
      this.Close();
   } catch (Exception ex) { msg.Error(ex.Message); }
}
private void btnClose_Click(object sender, EventArgs e) {
   this.Close();
}
private void QuestionsEntryForm_Load(object sender, EventArgs e)
{
}
}
}
```

Ο κώδικας που τρέχει για τη φόρμα εισαγωγής ερωτημάτων περνάει στον πίνακα δεδομένων του SQL Courses τα στοιχεία που ο καθηγητής εισάγει χρησιμοποιώντας το class Questions που έχουμε δημιουργήσει. Όταν κάνουμε επεξεργασία (edit) κάποιου ερωτήματος που ήδη υπάρχει τρέχει το script ώστε να φέρει από τον πίνακα Courses που γίνεται join με τον CourseName όλα τα στοιχεία του ερωτήματος και τα εισάγει στον πίνακα dt (DataTable dt) ελέγχοντας ότι το id του χρήστη είναι σωστό σύμφωνα με το καταχωρημένο id που έκανε την εγγραφή στους πίνακες της βάσης.

Παρακάτω, έχουμε το κουμπί Save όπου πατώντας το μετά την επεξεργασία βγάζει το ενημερωτικό μήνυμα “Οι αλλαγές αποθηκευτήκαν επιτυχώς!” και κλείνει τη φόρμα. Επίσης υπάρχει και το Close όπου κλείνει τη φόρμα.

Προχωράμε στη φόρμα δημιουργίας διαγωνισμάτων (Tests), όπου ο καθηγητής την επιλεγεί από τη αναπτυσσόμενη λίστα των Tests.



Εικόνα 10: Φόρμα Επιλογής Ερωτημάτων για Δημιουργία Test από Καθηγητή

Στην παραπάνω φόρμα ο καθηγητής μπορεί να δημιουργήσει, να διαγράψει ή να επεξεργαστεί νέα ή υπάρχοντα διαγωνίσματα. Αφού έχει δημιουργηθεί το διαγώνισμα μπορεί να το δημοσιεύσει ώστε να μπορούν οι σπουδαστές να το δηλώσουν στο διάστημα δήλωσης που έχει καταχωρηθεί από τον καθηγητή για αυτό το μάθημα. Αφού έχει έρθει το πέρας της εξέτασης ο καθηγητής μπορεί να επιλέξει την Ολοκλήρωση ώστε να μη μπορεί κάνεις άλλος να δηλώσει ή να λάβει μέρος στο διαγώνισμα. Ας δούμε τον κώδικα της φόρμας.

```
namespace Thanos.Reports {
public partial class TestsReportForm : Form {
public TestsReportForm(Users usr) {
   InitializeComponent();
   user = usr;
   GridRefresh();
   FixGrid();
}
DataBaseActions db = MainForm.db;
DataTable dt = new DataTable();
Messages msgs = new Messages();
Users user;
private void btnAdd_Click(object sender, EventArgs e) {
   try {
      Tests test = new Tests();
      TestsEntryForm nForm = new TestsEntryForm(test);
```

```
      nForm.ShowDialog(this);
      GridRefresh();
    } catch (Exception ex) { msgs.Error(ex.Message); }
  }
  private void GridRefresh(){
    dt = db.GetDataIntoTable("SELECT " +
                        "    t.ID, " +
                        "    ay.AcademicYear, " +
                        "    p.PeriodName, " +
                        "    t.TestName, " +
                        "    cn.Name, " +
                        "    cast(t.Published as varchar(15)) Published " +
                        " FROM tests t " +
                        " JOIN dbo.Courses c " +
                        "    ON c.ID = t.eCoursesID " +
                        " JOIN dbo.CourseName cn " +
                        "    ON cn.ID = c.eCourseNameID " +
                        " JOIN dbo.eAcademicyears ay " +
                        "    ON ay.ID = c.eAcademicYearID " +
                        " JOIN dbo.ePeriods p " +
                        "    ON p.ID = c.ePeriodsID " +
                        " WHERE t.eUserID = '"+user.ID+"' " +
                        " AND t.Published != 2 " +
                        " ORDER BY right(ay.AcademicYear,4) DESC, cn.Name ASC, p.PeriodName desc, t.TestName");
    foreach( DataRow dr in dt.Rows){
      var q = (Tests.Publish)(Convert.ToInt32(dr["Published"]));
      dr["Published"] = q.GetEnumDescription();
    }
    gcTests.DataSource = dt;
    gvTests.RefreshData();
  }
  private void FixGrid(){
    Dictionary<string, string> Translate = new Dictionary<string, string>();
    Translate.Add("AcademicYear", "Ακ. Έτος");
    Translate.Add("PeriodName", "Περίοδος");
    Translate.Add("TestName", "Ον. Test");
    Translate.Add("Name", "Ον. Μαθήματος");
    Translate.Add("Published", "Δημοσίευση");
    gvTests.ColumnsTranslate(Translate);
    gvTests.HideColumns();
    gvTests.OptionsSelection.MultiSelect = true;
    gvTests.OptionsSelection.MultiSelectMode = DevExpress.XtraGrid.Views.Grid.GridMultiSelectMode.CheckBoxRowSelect;
    gvTests.OptionsBehavior.ReadOnly = true;
    gvTests.OptionsBehavior.Editable = false;
  }
  private void btnDelete_Click(object sender, EventArgs e) {
    try {
      Int32[] SelectedRows = gvTests.GetSelectedRows();
      foreach (int i in SelectedRows) {
        Tests t = new Tests();
        Guid g = Guid.Parse(gvTests.GetRowCellValue(i, "ID").ToString());
        t.GetByID(g);
        if(t.Published == Tests.Publish.Published || t.Published == Tests.Publish.Completed){
          msgs.Error("Δεν μπορείτε να διαγράψετε \"Δημοσιευμένο\" ή \"Ολοκληρωμένο\" Test");
          return;
        }
      }
      gvTests.DeleteSelectedRow(typeof(Tests), GridRefresh);
    } catch (Exception ex) { msgs.Error(ex.Message); }
  }
  private void btnEdit_Click(object sender, EventArgs e) {
    try {
```

```
      gvTests.EditOnDoubleClick(this, typeof(Tests), typeof(TestsEntryForm), GridRefresh);
    } catch (Exception ex) { msgs.Error(ex.Message); }
  }
  private void gvTests_DoubleClick(object sender, EventArgs e) {
    try {
      gvTests.EditOnDoubleClick(this, typeof(Tests), typeof(TestsEntryForm), GridRefresh);
    } catch (Exception ex) { msgs.Error(ex.Message); }
  }
  private void btnPublish_Click(object sender, EventArgs e) {
    try {
      Int32[] SelectedRows = gvTests.GetSelectedRows();
      foreach (int i in SelectedRows) {
        Tests t = new Tests();
        Guid g = Guid.Parse(gvTests.GetRowCellValue(i, "ID").ToString());
        t.GetByID(g);
        if (t.Published == Tests.Publish.Completed) {
          msgs.Error("Δεν μπορείτε να Δημοσιεύσετε \"Ολοκληρωμένο\" Test");
          return;
        }
      }
      var dr = msgs.QuestionYesNo("Θέλετε να Δημοσιέυσετε τα επιλεγμένα Tests;");
      if(dr == DialogResult.Yes){
        foreach (int i in SelectedRows){
          Tests t = new Tests();
          Guid g = Guid.Parse(gvTests.GetRowCellValue(i, "ID").ToString());
          t.GetByID(g);
          t.Published = Tests.Publish.Published;
          t.UpdateStatus();
        }
      }
      GridRefresh();
    } catch (Exception ex) { msgs.Error(ex.Message); }
  }
  private void simpleButton1_Click(object sender, EventArgs e) {
    try {
      Int32[] SelectedRows = gvTests.GetSelectedRows();
      foreach (int i in SelectedRows) {
        Tests t = new Tests();
        Guid g = Guid.Parse(gvTests.GetRowCellValue(i, "ID").ToString());
        t.GetByID(g);
        if (t.Published == Tests.Publish.UnPublished) {
          msgs.Error("Δεν μπορείτε να Ολοκληρώσετε \"Μη Δημοσιευμένο\" Test");
          return;
        }
      }
      var dr = msgs.QuestionYesNo("Θέλετε να Ολοκληρωσετε τα επιλεγμένα Tests;");
      if (dr == DialogResult.Yes) {
        foreach (int i in SelectedRows) {
          Tests t = new Tests();
          Guid g = Guid.Parse(gvTests.GetRowCellValue(i, "ID").ToString());
          t.GetByID(g);
          t.Published = Tests.Publish.Completed;
          t.UpdateStatus();
        }
      }
      GridRefresh();
    } catch (Exception ex) { msgs.Error(ex.Message); }
  }
  private void TestsReportForm_Load(object sender, EventArgs e)
  {
  }
}
```

}

Ξεκινώντας, βλέπουμε ότι δημιουργείται η φόρμα με τα διαγωνίσματα του χρήστη που έχει κάνει login και το button της δημιουργίας (add) όπου καλεί την φόρμα εισαγωγής νέου διαγωνίσματος στο οποίο δηλώνουμε μάθημα και ονομασία που επιθυμούμε. Σαν μαθήματα μας δίνεται η δυνατότητα από dropdown list των μαθημάτων που έχουμε δηλώσει προηγουμένως στη δήλωση μαθημάτων που διδάσκουμε.

Παρακάτω βλέπουμε την class GridRefresh όπου καλείται και πριν κατά το άνοιγμα της φόρμας αλλά και κατά την εισαγωγή νέου διαγωνίσματος η οποία εισάγει τα δεδομένα που θέλουμε από την βάση δεδομένων μας στο DataTable dt που έχει δηλωθεί παραπάνω.

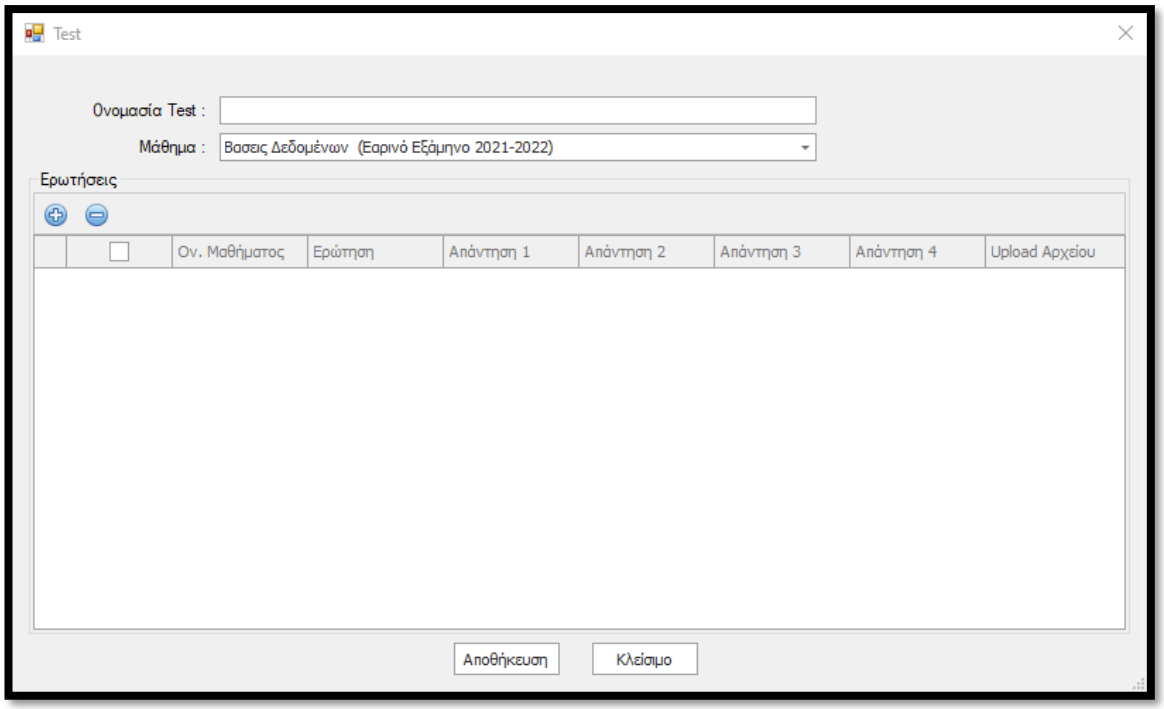
Έπειτα υπάρχει η class FixGrid όπου διορθώνει την εμφάνιση των στηλών της φόρμας, μεταφράζοντας με χρήση Dictionary της στήλες από τη βάση δεδομένων στα Ελληνικά.

Ακολουθεί το κουμπί delete της φόρμας όπου κάνει έλεγχο πριν τη διαγραφή του test αν ο καθηγητής το έχει κάνει δημοσίευση ή το έχει επιλέξει να είναι ολοκληρωμένο. Στις περιπτώσεις αυτές βγαίνει μήνυμα λάθους στον χρήστη που ενημερώνει ότι δε μπορεί να διαγραφεί test το οποίο είναι στις παραπάνω καταστάσεις. Εάν δεν είναι δημοσιευμένο ή ολοκληρωμένο, μπορεί να διαγραφεί.

Το κουμπί edit επίσης καλεί την class gvTests_DoubleClick, όπου το μόνο που κάνουν είναι να επιτρέπουν την επεξεργασία του τεστ.

Υπάρχουν και τα κουμπιά της δημοσίευσης (btnPublish_Click class) και ολοκλήρωσης (simpleButton1_Click) όπου αντίστοιχα αλλάζουν την κατάσταση του τεστ σε δημοσιευμένο ώστε οι σπουδαστές να μπορούν να το δηλώσουν και σε ολοκληρωμένο ώστε να έχει έλθει το πέρας της εξέτασης και να μη μπορεί κάποιος να δηλώσει το μάθημα. Όπως θα δούμε, υπάρχουν ενημερωτικά μηνύματα όταν προσπαθούμε να Δημοσιεύσουμε test το οποίο έχει επιλεχθεί να είναι Ολοκληρωμένο και αντίστοιχα αν προσπαθήσουμε να ολοκληρώσουμε test το οποίο ακόμα δεν έχει δημοσιευθεί.

Παρακάτω, θα δούμε τη φόρμα δημιουργίας νέου διαγωνίσματος όταν ο καθηγητής επιλεγεί το κουμπί (+) add από την προηγουμένη φόρμα. Η εικόνα της φόρμας:



Εικόνα 11: Φόρμα Δημιουργίας Test από Καθηγητή

Ακολουθεί ο κώδικας της φόρμας εισαγωγής των διαγωνισμάτων:

```
namespace Thanos.EntriesForms {
public partial class TestsEntryForm : Form {
public TestsEntryForm(Tests obj) {
   InitializeComponent();
   user = MainForm.user;
   test = obj;
   if(test.eUserID != Guid.Empty){
      test.Fill(this);
      GetListOfQuestions();
   }
   if (test.Published != Tests.Publish.UnPublished){
      label3.Visible = true;
      luCourses.Enabled = false;
      txtTestName.Enabled = false;
      btnAddQuestion.Enabled = false;
      btnDeleteQuestion.Enabled = false;
      btnSave.Enabled = false;
   }
   FixluCourses();
   RefreshGrid();
   FixGrid();
}
#region Vars
Tests test;
Users user;
DataTable dt = new DataTable();
Messages msgs = new Messages();
DataBaseActions db = MainForm.db;
DataTable dtQuestions = new DataTable();
```

```
        List<Guid> ListOfQuestions = new List<Guid>();
        List<TestQuestions> ListOfTestQuestions = new List<TestQuestions>();
        Guid CourseGuid;
        #endregion Vars
        private void GetListOfQuestions(){
          ListOfTestQuestions.Clear();
          foreach (DataRow dr in db.GetDataIntoTable("SELECT eQuestionID FROM TestQuestions WHERE eTestID =
'"+test.ID+"'").Rows){
            ListOfQuestions.Add(Guid.Parse(dr[0].ToString()));
          }
        }
        private void FixluCourses() {
          dt = db.GetDataIntoTable("SELECT  " +
                        " c.ID " +
                        " , cn.Name " +
                        " + '  (' + p.PeriodName + ' ' " +
                        " + ay.AcademicYear + ')' Display " +
                        "FROM dbo.Courses c " +
                        "JOIN dbo.CourseName cn " +
                        "    ON cn.ID = c.eCourseNameID " +
                        "JOIN dbo.ePeriods p " +
                        "    ON p.ID = c.ePeriodsID " +
                        "JOIN dbo.eAcademicYears ay" +
                        "    ON ay.ID = c.eAcademicYearID " +
                        "WHERE c.eUserID = '" +user.ID+"'");
          luCourses.SetLookUpEdit(dt, "Display", "ID");
          luCourses.Refresh();
        }
        private void FixGrid() {
          Dictionary<string, string> Translates = new Dictionary<string, string>();
          Translates.Add("Name", "Ον. Μαθήματος");
          Translates.Add("mainQuestion", "Ερώτηση");
          Translates.Add("Answer1", "Απάντηση 1");
          Translates.Add("Answer2", "Απάντηση 2");
          Translates.Add("Answer3", "Απάντηση 3");
          Translates.Add("Answer4", "Απάντηση 4");
          Translates.Add("AllowUploadFile", "Upload Αρχείου");
          gvQuestions.ColumnsTranslate(Translates);
          gvQuestions.HideColumns();
          gvQuestions.OptionsSelection.MultiSelect = true;
          gvQuestions.OptionsSelection.MultiSelectMode =
DevExpress.XtraGrid.Views.Grid.GridMultiSelectMode.CheckBoxRowSelect;
          gvQuestions.OptionsBehavior.Editable = false;
        }
        private void FillListOfTestQuestions() {
          foreach (Guid g in ListOfQuestions) {
            TestQuestions obj = new TestQuestions();
            obj.eQuestionID = g;
            obj.eTestID = test.ID;
            ListOfTestQuestions.Add(obj);
          }
        }
        private void RefreshGrid() {
          string GuidsExist = string.IsNullOrEmpty(string.Join("'\',\'", ListOfQuestions)) ? Guid.Empty.ToString() : string.Join("','",
ListOfQuestions);
            dtQuestions = db.GetDataIntoTable(" SELECT " +
                        "    q.ID " +
                        "    , cn.Name " +
                        "    , q.mainQuestion " +
                        "    , q.Answer1 " +
                        "    , q.Answer2 " +
                        "    , q.Answer3 " +
```

```
                    "    , q.Answer4 " +
                    "    , q.AllowUploadFile " +
                    " FROM questions q " +
                    " JOIN dbo.CourseName cn " +
                    "     ON cn.ID = q.eCourseNameID " +
                    " where eUserID = '" + user.ID + "' " +
                    " and eCourseNameID ='" + CourseGuid + "'"+
                    " and q.id in ('" + GuidsExist + "')");
      gcQuestions.DataSource = dtQuestions;
      gvQuestions.RefreshData();
    }
    private void lookUpEdit1_EditValueChanged(object sender, EventArgs e) {
      CourseGuid = Guid.Parse(db.GetDataIntoTable("SELECT eCourseNameID FROM dbo.Courses WHERE id ='" +
luCourses.EditValue + "'").Rows[0][0].ToString());
      ListOfQuestions.Clear();
      RefreshGrid();
    }
    private void btnClose_Click(object sender, EventArgs e) {
      this.Close();
    }
    private void btnsQuestion_Click(object sender, EventArgs e) {
      try {
        string ControlName = (sender as SimpleButton).Name;
        switch (ControlName){
          case "btnAddQuestion":
            MassQuestionsAdd nForm = new MassQuestionsAdd(ListOfQuestions, CourseGuid);
            nForm.ShowDialog(this);
            RefreshGrid();
          break;
          case "btnDeleteQuestion":
            Int32[] SelectedRows = gvQuestions.GetSelectedRows();
            foreach (int i in SelectedRows) {
              Guid guid = Guid.Parse(gvQuestions.GetRowCellValue(i, "ID").ToString());
              if (ListOfQuestions.Contains(guid)) {
                ListOfQuestions.Remove(guid);
              }
            }
            RefreshGrid();
            break;
        }
      } catch (Exception ex) { msgs.Error(ex.Message); }
    }
    private void btnSave_Click(object sender, EventArgs e) {
      try {
        var val = msgs.QuestionYesNo("Θέλετε να αποθηκεύσετε το Τεστ?");
        if (val == DialogResult.Yes){
          FillListOfTestQuestions();
          this.Fill(test);
          test.eUserID = user.ID;
          test.Save(ListOfTestQuestions.Cast<ISqlTable>().ToList());
          msgs.Information("Η αποθήκευση ολοκληρώθηκε!");
          this.Close();
        }
      } catch (Exception ex) { msgs.Error(ex.Message); }
    }
    private void TestsEntyForm_Load(object sender, EventArgs e)
    {
    }
    private void groupBox1_Enter(object sender, EventArgs e)
    {
    }
  }
```

}

Παρατηρούμε ότι τρέχει την κλάση FixluCourses με την οποία εισάγει στο dropdown list τα μαθήματα που έχει αναλάβει ο καθηγητής και τα έχει δηλώσει στο μενού των Μαθήματων. Και πάλι τρέχουμε το class FixGrid ώστε να μεταφράσουμε τις επικεφαλίδες των στηλών.

Εδώ τρέχει η κλάση GetListOfQuestions η οποία επιλέγει το id κάθε ερώτησης που υπάρχει καταχωρημένη στη βάση και τα περνάει σε ένα Data Table το οποίο καλείται σε επόμενες κλάσεις.

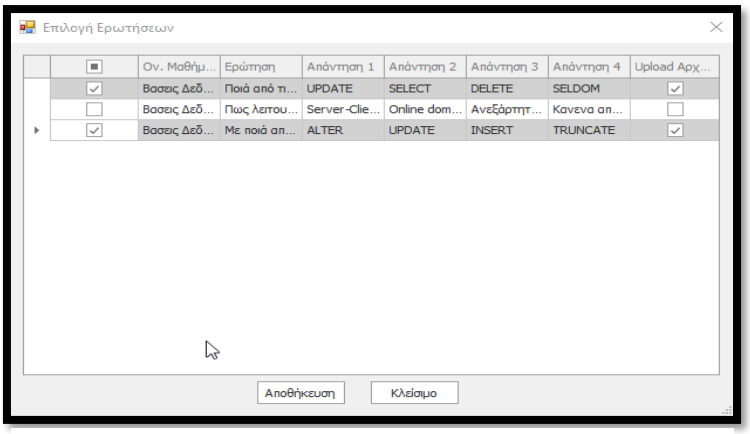
Έπειτα, τρέχει η κλάση FillListOfTestQuestions η οποία τραβάει τα ID των ερωτημάτων που έχουν καταχωρηθεί ήδη από τον χρήστη στη βάση δεδομένων SQL. Τρέχει επίσης και η κλάση RefreshGrid η οποία εμφανίζει τα ερωτήματα που έχουμε επιλέξει στη φόρμα του test που έχουν καταχωρηθεί στη βάση ήδη και μας τα εμφανίζει.

Παρακάτω, έχουμε τις κλάσεις για τα κουμπιά edit και close, όπου στο πρώτο διαβάζει τις ερωτήσεις που υπάρχουν ήδη για το επιλεγμένο προς επεξεργασία τεστ και τις αφαιρεί από τη λίστα των διαθέσιμων ερωτήσεων προς επιλογή και το δεύτερο όπου τερματίζει τη φόρμα.

Μετά τρέχει η κλάση btnsQuestion_Click η οποία προσθέτει μια νέα ερώτηση ή αφαιρεί μια ερώτηση που έχει επιλεχθεί για το τεστ. Όταν επιλέγουμε την προσθήκη ερώτησης καλείται η φόρμα MassQuestionsAdd όπου βγάζει όλες τις διαθέσιμες ερωτήσεις για το επιλεγμένο μάθημα. Αντίστοιχα η αφαίρεση ερώτησης τη διαλέγει από το grid των υπαρχόντων ερωτήσεων και την αφαιρεί. Έχουμε βάλει και τη δυνατότητα αφαίρεσης πολλαπλών επιλεγμένων ερωτήσεων.

Τέλος υπάρχει το κουμπί save όπου πατώντας το μας βγάζει ενημερωτικό μήνυμα “Θέλετε να αποθηκεύσετε το Τεστ?” στο οποίο αν επιλέξουμε Ναι αποθηκεύει στον πίνακα Tests της βάσης SQL το τεστ και κλείνει το παράθυρο διαλόγου, αν επιλέξουμε όχι μας επιστρέφει στο τεστ για να συνεχίσουμε την επεξεργασία.

Στη συνέχεια, θα δούμε και θα επεξηγήσουμε την προαναφερθείσα φόρμα MassQuestionsAdd και τον κώδικα της. Η φόρμα αυτή ευθύνεται για την εμφάνιση των ερωτημάτων όταν τα προσθέτουμε σε νέο τεστ που δημιουργούμε.



Εικόνα 12: Φόρμα Εισαγωγής Ερωτημάτων για Test

Η εικονιζόμενη φόρμα είναι η φόρμα εισαγωγής ερωτημάτων MassQuestionsAdd. Είναι η φόρμα στην οποία εμφανίζεται η λίστα όλων των ερωτημάτων που έχει δημιουργήσει ο καθηγητής για το επιλεγμένο μάθημα και από την οποία μπορεί να επιλέξει ποιες θέλει να εισαχθούν στο διαγώνισμα το οποίο δημιουργεί τη δεδομένη στιγμή. Ας δούμε και τον κώδικα της παραπάνω φόρμας:

```
namespace Thanos.MassEntries {
public partial class MassQuestionsAdd : Form {
public MassQuestionsAdd(List<Guid> obj,Guid coGuid) {
   InitializeComponent();
   ListOfQuestions = obj;
   user = MainForm.user;
   CourseGuid = coGuid;
   RefreshGrid();
   FixGrid();
}
List<Guid> ListOfQuestions;
DataBaseActions db = MainForm.db;
DataTable dt = new DataTable();
Messages msgs = new Messages();
Users user;
Guid CourseGuid;
private void FixGrid(){
   Dictionary<string, string> Translates = new Dictionary<string, string>();
   Translates.Add("Name", "Ον. Μαθήματος");
   Translates.Add("mainQuestion", "Ερώτηση");
   Translates.Add("Answer1", "Απάντηση 1");
   Translates.Add("Answer2", "Απάντηση 2");
   Translates.Add("Answer3", "Απάντηση 3");
   Translates.Add("Answer4", "Απάντηση 4");
   Translates.Add("AllowUploadFile", "Upload Αρχείου");
   gvQuestions.ColumnsTranslate(Translates);
   gvQuestions.HideColumns();
   gvQuestions.OptionsSelection.MultiSelect = true;
   gvQuestions.OptionsSelection.MultiSelectMode =
DevExpress.XtraGrid.Views.Grid.GridMultiSelectMode.CheckBoxRowSelect;
   gvQuestions.OptionsBehavior.Editable = false;
}
private void RefreshGrid(){
   string GuidsExist = string.IsNullOrEmpty(string.Join("','", ListOfQuestions)) ? Guid.Empty.ToString() : string.Join("','",
ListOfQuestions);
      dt = db.GetDataIntoTable(" SELECT " +
```

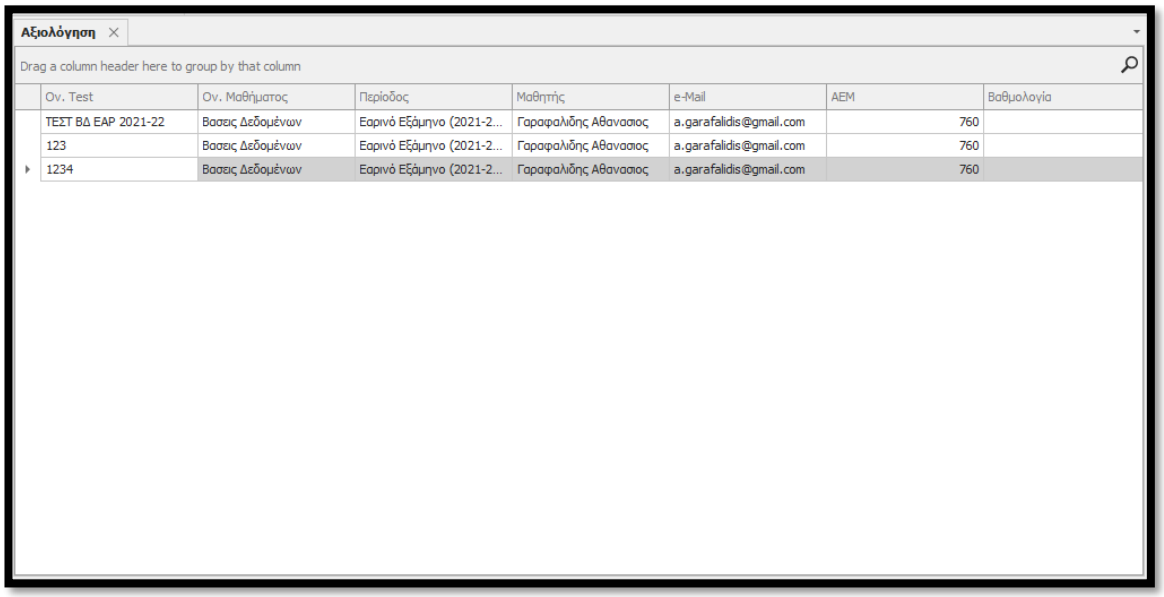
```
                    "    q.ID " +
                    "    , cn.Name " +
                    "    , q.mainQuestion " +
                    "    , q.Answer1 " +
                    "    , q.Answer2 " +
                    "    , q.Answer3 " +
                    "    , q.Answer4 " +
                    "    , q.AllowUploadFile " +
                    " FROM questions q " +
                    " JOIN dbo.CourseName cn " +
                    "     ON cn.ID = q.eCourseNameID " +
                    " where eUserID = '" + user.ID + "' " +
                    " and eCourseNameID ='" + CourseGuid + "'" +
                    " and q.id not in ('" + GuidsExist + "')");
    gcQuestions.DataSource = dt;
    gvQuestions.RefreshData();
  }
  private void btnSave_Click(object sender, EventArgs e) {
    Int32[] SelectedRows = gvQuestions.GetSelectedRows();
    foreach (int i in SelectedRows){
      Guid guid = Guid.Parse(gvQuestions.GetRowCellValue(i, "ID").ToString());
      if(!ListOfQuestions.Contains(guid)){
        ListOfQuestions.Add(guid);
      }
    }
    this.Close();
  }
  private void btnClose_Click(object sender, EventArgs e) {
    this.Close();
  }
  private void MassQuestionsAdd_Load(object sender, EventArgs e)
  {
  }
  }
  }
```

Ξεκινώντας, στη φόρμα αυτή έχουμε την κλάση FixGrid. Ηκλάση αυτή μεταφράζει τις στήλες που εμφανίζονται στη φόρμα από τη βάση SQL στα Ελληνικά χρησιμοποιώντας Dictionary. Επίσης κρύβει τις υπόλοιπες στήλες του πίνακα που δε θέλουμε να εμφανίζονται.

Έπειτα, με την κλάση RefreshGrid αντλεί από τη βάση δεδομένων τα πεδία που χρειαζόμαστε για τη φόρμα από τους πίνακες Questions και CourseName κάνοντάς τους join και περνάει τα δεδομένα χρησιμοποιώντας DataSource ώστε να εισαχθούν στα αντίστοιχα πεδία της φόρμας. Το RefreshData φορτώνει να τα δεδομένα που αντλούνται στο DataSource ώστε να εμφανιστούν στην φόρμα κάθε φορά που γίνεται κάποια επιλογή.

Παρακάτω έχουμε τα κουμπιά Save και Close, από τα οποία το Save αποθηκεύει τα επιλεγμένα ερωτήματα στη φόρμα του διαγωνίσματος και μετά κλείνει τη φόρμα επιλογής ερωτημάτων και το Close απευθείας κλείνει τη φόρμα χωρίς να αποθηκευτούν οποίες ερωτήσεις έχουν επιλεχθεί στο διαγώνισμα.

Στη συνέχεια, θα δούμε τη φόρμα Αξιολόγησης EevaluationReportForm. Η φόρμα εμφανίζεται στον χρήστη σύμφωνα με την ακόλουθη εικόνα:



Εικόνα 13: Φόρμα Αξιολόγησης Εξεταζόμενων

Ας δούμε και τον κώδικα που συσχετίζεται με την παραπάνω φόρμα.

```
namespace Thanos.Reports {
public partial class EevaluationReportForm : Form {
public EevaluationReportForm() {
   InitializeComponent();
   user = MainForm.user;
   GridRefresh();
   FixGridView();
}
Users user;
DataBaseActions db = MainForm.db;
Messages msgs = new Messages();
private void GridRefresh() {
gcEvaluation.DataSource = db.GetDataIntoTable("SELECT t.ID, " +
                                  "      t.TestName, " +
                                  "      cn.Name, " +
                                  "      p.PeriodName + ' (' + ay.AcademicYear + ')' Period, " +
                                  "      u.Surname + ' ' + u.Name [Student], " +
                                  "      u.email [e-Mail], " +
                                  "      u.AEM, " +
                                  "      u.ID studentID, " +
                                  "      sg.Grade " +
                                  "FROM Tests t " +
                                  "   JOIN dbo.Courses c " +
                                  "      ON c.ID = t.eCoursesID " +
                                  "   JOIN dbo.CourseName cn " +
                                  "      ON cn.ID = c.eCourseNameID " +
                                  "   JOIN dbo.ePeriods p " +
                                  "      ON p.ID = c.ePeriodsID " +
                                  "   JOIN dbo.eAcademicYears ay " +
```

```
                                "       ON ay.ID = c.eAcademicYearID " +
                                "    JOIN " +
                                "    (SELECT DISTINCT eTestID, eUserID FROM dbo.StudentsAnswers) sta " +
                                "       ON t.ID = sta.eTestID " +
                                "    LEFT JOIN StudentsGrading sg " +
                                "       ON sta.eUserID = sg.StudentID " +
                                "          AND sg.eTestID = t.ID " +
                                "    JOIN Users u " +
                                "       ON u.ID = sta.eUserID " +
                               "WHERE sg.ID IS null " +
                               "AND t.Published = 2" +
                               "AND t.eUserID = '" +user.ID+"'");
}
private void FixGridView() {
   Dictionary<string, string> Translate = new Dictionary<string, string>();
   Translate.Add("Name", "Ον. Μαθήματος");
   Translate.Add("TestName", "Ον. Test");
   Translate.Add("Period", "Περίοδος");
   Translate.Add("Student", "Μαθητής");
   Translate.Add("Grade", "Βαθμολογία");
   Translate.Add("AEM", "ΑΕΜ");
   gvEvaluation.ColumnsTranslate(Translate);
   gvEvaluation.HideColumns(new string[] {"ID", "studentID" });
   gvEvaluation.OptionsBehavior.ReadOnly = true;
   gvEvaluation.OptionsBehavior.Editable = false;
}
private void gvEvaluation_DoubleClick(object sender, EventArgs e) {
   try {
      gvEvaluation.ClearSelection();
      int SelectedRow = gvEvaluation.FocusedRowHandle;
      if (SelectedRow > 0) {
         Guid TestID = Guid.Parse(gvEvaluation.GetRowCellValue(SelectedRow, "ID").ToString());
         Guid StudentID = Guid.Parse(gvEvaluation.GetRowCellValue(SelectedRow, "studentID").ToString());
         EvaluationEntryForm nForm = new EvaluationEntryForm(TestID, StudentID);
         nForm.ShowDialog(this);

         GridRefresh();
         gvEvaluation.FocusedRowHandle = SelectedRow;
      }
   } catch (Exception ex) { msgs.Error(ex.Message); }
}
private void panel2_Paint(object sender, PaintEventArgs e)
{
}
}
}
```

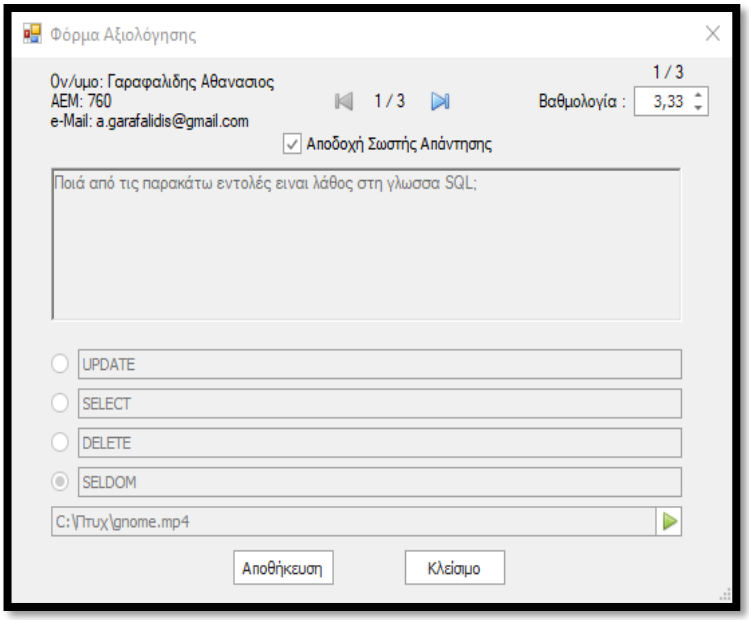
Στη παρούσα φόρμα έχουμε αρχικά την κλάση GridRefresh η οποία επιλεγεί δεδομένα από 7 πίνακες από τη βάση δεδομένων μας κάνοντας select και συνδέοντας με join τους πίνακες στα αντίστοιχα ID και εισάγοντάς τα στο DataSource της φόρμας. Οι πίνακες είναι οι Tests, Courses, CourseName, ePeriods, eAcademicYears, StudentsGrading και Users όπου φέρνει τα στοιχεία του id και ονόματος του διαγωνίσματος, όνομα μαθήματος, την ακαδημαϊκή περίοδο και έτος, ονοματεπώνυμο, email, ΑΕΜ και id σπουδαστή και τέλος τον βαθμό που πήρε. Φυσικά τα τεστ που αντλούνται με τη κλάση αυτή πρέπει να είναι χαρακτηρισμένα

από τον καθηγητή ως Ολοκληρωμένα, η τιμή του οποίου είναι στον πίνακα Tests στη στήλη Published όπου η τιμή είναι 2 σημαίνει ότι είναι Ολοκληρωμένο.

Ακολουθεί η κλάση FixGridView η οποία όπως έχουμε δει και παραπάνω κάνει μετάφραση (Translate) με χρήση λεξικού (Dictionary) τις στήλες που εμφανίζονται πάνω στη φόρμα στα ελληνικά. Έπειτα κρύβει τις υπόλοιπες στήλες που δεν έχουμε επιλέξει για μετάφραση και κάνει τα πεδία στηλών read-only και non editable, δηλαδή να μη μπορεί ο χρήστης να τα επεξεργαστεί.

Έπεται η κλάση DoubleClick όπου είναι για να μπορεί ο καθηγητής να μπει και να επεξεργαστεί την αυτόματη αξιολόγηση που γίνεται σύμφωνα με τις απαντήσεις που έχουν δώσει οι σπουδαστές καλώντας τη φόρμα EvaluationEntryForm.

Η φόρμα αξιολόγησης των απαντημένων ερωτήσεων των διαγωνισμάτων EvaluationEntryForm απεικονίζεται στο χρήστη ως εξής:



Εικόνα 14: Φόρμα Βαθμολόγησης Απάντησης Test

Έχει την ονομασία του διαγωνίσματος, κουμπιά για να μπορεί ο καθηγητής να πάει σε προηγούμενη ή επόμενη ερώτηση, το πλαίσιο που περιέχει την ερώτηση και τις απαντήσεις με την απάντηση που επέλεξε ο σπουδαστής μαρκαρισμένη, καθώς και το επισυναπτόμενο βίντεο αν αυτό είναι ενεργό. Επίσης έχει το τσεκ της Αποδοχής Σωστής Απάντησης όπου μπορεί ο καθηγητής να το επιλέξει ώστε να μετρήσει σαν σωστή την απάντηση που δίνει ο σπουδαστής αν κρίνει ότι πρέπει ακόμα και αν ο σπουδαστής έχει διαλέξει λάθος στο διαγώνισμα. Επίσης πάνω δεξιά επιτρέπει στον καθηγητή την επεξεργασία για αλλαγή του αυτομάτου βαθμού καθώς μπορεί να

επιθυμεί να αφαιρέσει/προσθέσει βαθμούς αναλόγως με την βιντεοσκοπημένη απάντηση του σπουδαστή.

Ας δούμε και τον κώδικα της φόρμας αυτής:

```
namespace Thanos.EntriesForms {
public partial class EvaluationEntryForm : Form {
public EvaluationEntryForm(Guid TestID,Guid StudentID) {
   this.SuspendLayout();
   InitializeComponent();
   user = MainForm.user;
   StudentGrade.StudentID = StudentID;
   StudentGrade.TeacherID = user.ID;
   StudentGrade.eTestID = TestID;
   GetStudent(StudentID.ToString());
   string qListOfQuestions = " SELECT q.* FROM Questions q " +
                        " JOIN dbo.TestQuestions t ON t.eQuestionID = q.ID " +
                        " WHERE t.eTestID = '"+ TestID.ToString()+ "'";
   ListOfQuestions =
db.GetDataIntoTable(qListOfQuestions).TurnToListOfObjects(typeof(Questions)).Cast<Questions>().ToList();
   string qListOfAnswers = "SELECT * from dbo.StudentsAnswers WHERE eTestID = '" + StudentGrade.eTestID + "'";
   ListOfAnswers =
db.GetDataIntoTable(qListOfAnswers).TurnToListOfObjects(typeof(StudentsAnswers)).Cast<StudentsAnswers>().ToList();
   GetCorrectAnswers();
   Question = ListOfQuestions[1];
   UpdateControls();
   UpdateGradeLabel();
   this.ResumeLayout();
}
StudentsGrading StudentGrade = new StudentsGrading();
Users user;
Questions Question;
StudentsAnswers Answer;
DataBaseActions db = MainForm.db;
Messages msgs = new Messages();
List<Questions> ListOfQuestions = new List<Questions>();
List<StudentsAnswers> ListOfAnswers = new List<StudentsAnswers>();
Dictionary<Questions, bool> CorrectAnswers = new Dictionary<Questions, bool>();
Messages msg = new Messages();
string exportPath = System.IO.Path.GetDirectoryName(Application.ExecutablePath) + @"\BinaryFilesFromDB";
int SelectedQuestion = 1;

private void  GetStudent(string StudentID) {
   string txt = "Ov/υμο: {0}\nAEM: {1}\ne-Mail: {2}";
   string[] vals =  db.GetDataIntoTable("SELECT CAST(ISNULL(AEM,'') AS NVARCHAR(20))+ ','+ Surname+' '+Name+ ',' " +
      "+email FROM Users WHERE id = '" + StudentID + "'").Rows[0][0].ToString().Split(',');
   lbStudent.Text = string.Format(txt,vals[1], vals[0], vals[2]);
}
private void UpdateControls() {
   string LabelText = "{0} / {1}";
   label1.Text = string.Format(LabelText, SelectedQuestion.ToString(), ListOfQuestions.Count.ToString());
   if (SelectedQuestion == 1) {
      btnPrevius.Enabled = false;
      btnNext.Focus();
   } else { btnPrevius.Enabled = true; }
   if (SelectedQuestion == ListOfQuestions.Count) {
      btnNext.Enabled = false;
      btnPrevius.Focus();
   } else { btnNext.Enabled = true; }
   FillControlsFromObjects();
```

```
}
private void GetCorrectAnswers() {
   try {
      CorrectAnswers.Clear();
      foreach (Questions q in ListOfQuestions) {
         var answer = ListOfAnswers.First(a => a.eQuestionID == q.ID);
         bool CorrectAnswer = false;
         if(answer.stAnswer1 == q.CorrectAnswer1 && answer.stAnswer2 == q.CorrectAnswer2 && answer.stAnswer3 ==
            q.CorrectAnswer3 && answer.stAnswer4 == q.CorrectAnswer4){
            CorrectAnswer = true;
         }
         CorrectAnswers.Add(q, CorrectAnswer);
      }
   } catch { throw; }
}
private void FillControlsFromObjects() {
   Question = ListOfQuestions[SelectedQuestion - 1];
   Answer = ListOfAnswers.First(a => a.eQuestionID == Question.ID);
   Question.Fill(this);
   Answer.Fill(this);
   chApproved.Checked = CorrectAnswers[Question];
   txtPath.Enabled = Question.AllowUploadFile;
   btnNavigate.Enabled = Question.AllowUploadFile;
   txtPath.Text = Answer.FilePath;
   txtPath.Enabled = false;
}
private void UpdateGradeLabel(){
   string txt = "{0} / {1}";
   int Count = CorrectAnswers.Count;
   int cCount = CorrectAnswers.Where(x => x.Value == true).Count();
   lbCorrectAnswers.Text = string.Format(txt,cCount.ToString(), Count.ToString());
   spinEdit1.Value = Math.Round(Convert.ToDecimal((10*(float)cCount)/(float)Count ),2);
}
private void btnSave_Click(object sender, EventArgs e) {
   try {
      this.Fill(StudentGrade);
      StudentGrade.Save();
      this.Close();
   } catch (Exception ex) { msgs.Error(ex.Message); }
}
private void btnClose_Click(object sender, EventArgs e) {
   this.Close();
}
private void btnNext_Click(object sender, EventArgs e) {
   try {
      if (SelectedQuestion != ListOfQuestions.Count) {
         SelectedQuestion = SelectedQuestion + 1;
         UpdateControls();
      }
   } catch (Exception ex) { msgs.Error(ex.Message); }
}
private void btnPrevius_Click(object sender, EventArgs e) {
   try {
      if (SelectedQuestion != 1) {
         SelectedQuestion = SelectedQuestion - 1;
         UpdateControls();
      }
   } catch (Exception ex) { msgs.Error(ex.Message); }
}
private void chApproved_CheckedChanged(object sender, EventArgs e) {
   CorrectAnswers[Question] = chApproved.Checked;
   UpdateGradeLabel();
```

```
}
private void btnNavigate_Click(object sender, EventArgs e) {
  try {
    DataTable dt = db.GetDataIntoTable("select * from fsanswers where id ='" + Answer.eFsAnswersID + "'");
    if (!Directory.Exists(exportPath)) {
      Directory.CreateDirectory(exportPath);
    }
    byte[] imageBytes = (byte[])dt.Rows[0]["BinaryFile"];
    if (imageBytes.Length > 0) {
      string path = exportPath + @"\" + Answer.FileName;
      FileStream fs = new FileStream(path, System.IO.FileMode.OpenOrCreate, System.IO.FileAccess.ReadWrite);
      fs.Write(imageBytes, 0, imageBytes.Length);
      fs.Close();
      Process.Start(path);
    }
  }catch(Exception ex) { msg.Error(ex.Message); }
}
private void EvaluationEntryForm_FormClosed(object sender, FormClosedEventArgs e) {
  Directory.CreateDirectory(exportPath);
}
private void EvaluationEntryForm_Load(object sender, EventArgs e)
{
}
}
}
```

Στην παρούσα φόρμα, ξεκινάμε τον κώδικα σταματώντας τις λειτουργίες της φόρμας με το SuspendLayout. Αυτό γίνεται ώστε να δηλώσουμε τις μεταβλητές των λιστών ερωτήσεων και απαντήσεων όπως και να καλέσουμε τις αντίστοιχες κλάσεις χωρίς να ανανεώνει σε κάθε επιλογή που κάνουμε η φόρμα τα στοιχεία, αλλά αφού τελειώσουμε όλες τις επιλογές μας τότε να ενημερώνει το layout της φόρμας. Εδώ δηλώνονται τα ID του σπουδαστή, του καθηγητή καθώς και του διαγωνίσματος. Δημιουργούνται η λίστα ερωτήσεων (ListOfQuestions) επιλέγοντας όλα τα στοιχεία της αντίστοιχης ερώτησης από τον πίνακα της βάσης Questions με σύνδεση στον TestQuestions και η λίστα απαντήσεων (ListOfAnswers) επιλέγοντας από τον πίνακα StudentsAnswers όλα τα στοιχεία των απαντήσεων που έδωσε ο σπουδαστής για το παρών διαγώνισμα.

Έπειτα, δηλώνουμε τις μεταβλητές που θα χρειαστούμε από το σχήμα της βάσης μας, StudentGrade, user, Question, Answer, τις λίστες που δηλώσαμε και παραπάνω ListOfQuestions και ListOfAnswers, καθώς και τις σωστές απαντήσεις CorrectAnswers, μηνύματα msg και το exportPath όπου είναι ο φάκελος που αποθηκεύονται τα αρχεία που ανεβάζουν οι σπουδαστές από τις απαντήσεις τους με τη χρήση FileStream. Η μεταβλητή SelectedQuestion δηλώνεται ως 1 για να κληθεί παρακάτω ως πρώτη κατά σειρά ερώτηση.

Ακολουθεί η κλάση Get Student που μας φέρνει τα στοιχεία του σπουδαστή που απάντησε το διαγώνισμα αντλώντας τα με select από τον πίνακα Users της βάσης δεδομένων μας.

Παρακάτω, η κλάση UpdateControls, έχει να κάνει με τις επιλογές από τα βελάκια για να πάμε σε επόμενη ή προηγούμενη ερώτηση, απενεργοποιώντας το κουμπί Προηγούμενο αν η ερώτηση είναι η πρώτη κατά σειρά (εδώ καλείται το SelectedQuestion για πρώτη φορά που έχουμε δηλώσει ως 1) ή αντίστοιχα αν είναι η τελευταία ερώτηση απενεργοποιεί το κουμπί Επόμενο (δηλαδή το SelectedQuestion ισούται με τον συνολικό αριθμό των ερωτήσεων της λίστας εξ ου και το ListOfQuestions.Count). Εκτελεί επίσης και την κλάση FillControlsFromObjects που θα δούμε παρακάτω.

Μετά έρχεται η κλάση GetCorrectAnswers όπου από τη λίστα των ερωτήσεων ListOfQuestions ελέγχουμε αν είναι σωστή η απάντηση ή λάθος ώστε να χρησιμοποιηθεί για τον αυτόματο υπολογισμό του βαθμού παρακάτω.

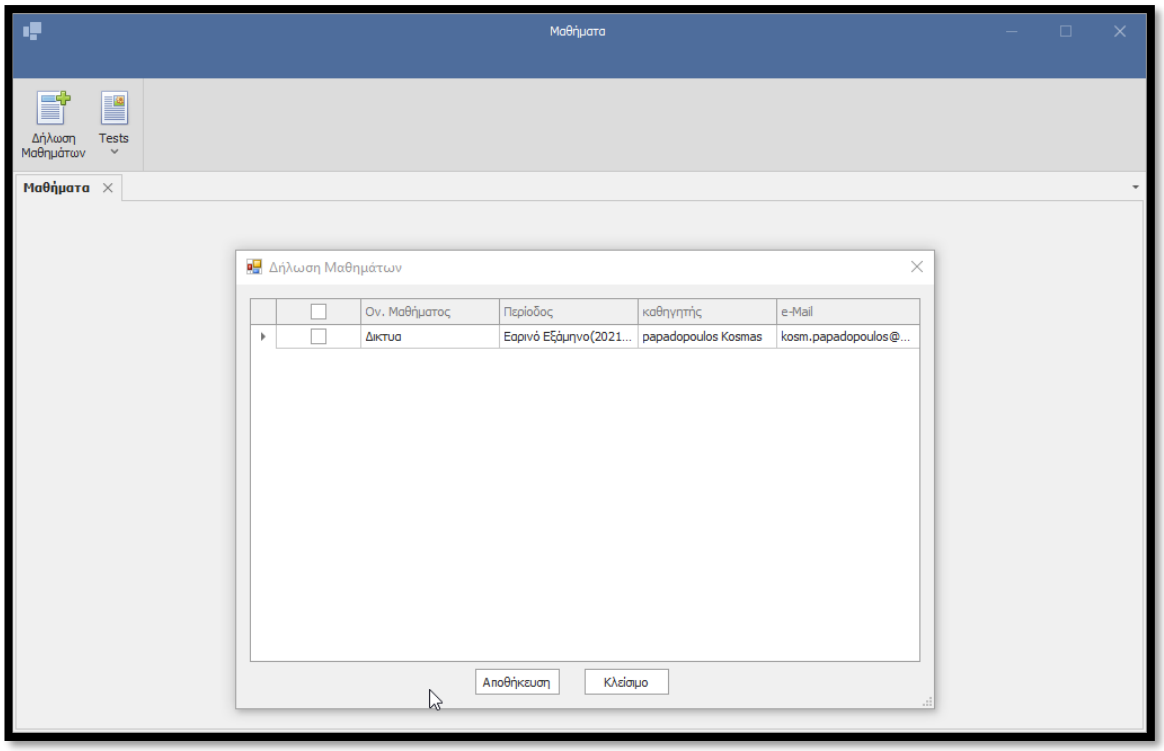
Έπεται η κλάση FillControlsFromObjects που συμπληρώνει όλα τα απαραίτητα πεδία της αντίστοιχης ερώτησης που έχει επιλεχθεί ελέγχοντας με το ID της ερώτησης. Η κλάση αυτή καλείται κάθε φορά που ο καθηγητής πατάει το επόμενο ή προηγούμενο στη φόρμα αξιολόγησης ώστε ανανεώσει το grid με τα στοιχεία της επομένης ερώτησης. Τα στοιχεία αυτά περιλαμβάνουν την ιδιά την ερώτηση, απαντήσεις, επιβεβαίωση σωστής απάντησης, ενεργοποίηση ανεβάσματος αρχείου αλλά και το αρχείο καθώς και τη δυνατότητα επιλογής αναπαραγωγής του αρχείου.

Στη συνέχεια, η κλάση UpdateGradeLabel υπολογίζει καλώντας το CorrectAnswers τις απαντήσεις που δόθηκαν σωστά και το σύνολο όλων των απαντήσεων ώστε να δώσει τον τελικό βαθμό ανά ερώτηση με άριστα το 10. Έχουμε βάλει τη δυνατότητα 2 δεκαδικών ψηφίων για τη βαθμολόγηση στην παρούσα έκδοση.

Ακολουθούν οι κλάσεις των κουμπιών Save, Close, Next, Previous καθώς και του check Αποδοχής Σωστής Απάντησης καθώς και το κουμπί αναπαραγωγής του επισυναπτόμενου αρχείου αν αυτό υπάρχει.

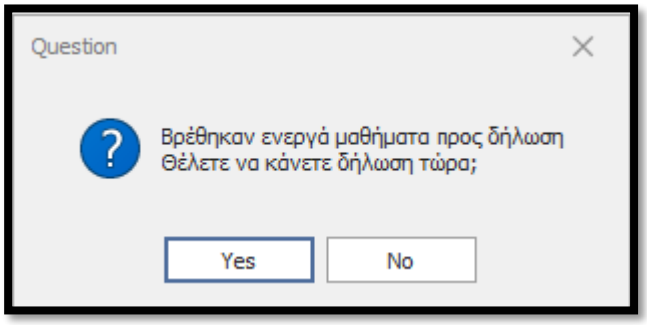
5.3.2 Φόρμες Σπουδαστών

Στην ενότητα αυτή θα δούμε τις φόρμες των σπουδαστών που θα χρησιμοποιούν την εφαρμογή. Ξεκινώντας θα δούμε τη φόρμα δήλωσης μαθήματων προς εξέταση, η οποία ονομάζεται StudentsCoursesReportForm. Ας δούμε πως απεικονίζεται η φόρμα στο πρόγραμμα όταν ο φοιτητής επιλέγει την Δήλωση Μαθημάτων.



Εικόνα 15: Κυρίως Φόρμα Φοιτητών

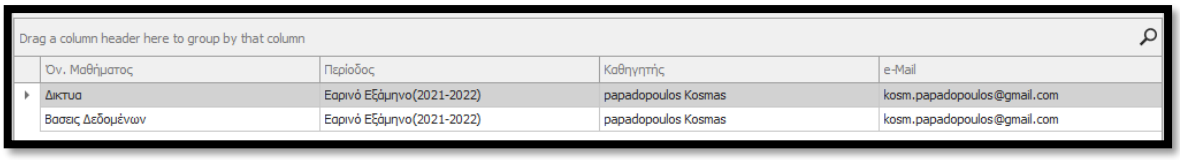
Εδώ απεικονίζεται η φόρμα κατά τη δήλωση νέων μαθημάτων. Όταν πατάει τη Δήλωση Μαθημάτων ο σπουδαστής και υπάρχουν νέα μαθήματα τα οποία έχει συμμετάσχει και έχει επιτρέψει ο καθηγητής την δήλωσή τους λαμβάνει το παρακάτω μήνυμα:



Εικόνα 16: Αναδυόμενη ειδοποίηση για Δήλωση Μαθημάτων

Πατώντας το Yes προχωράει στο βήμα που απεικονίζεται στην προηγουμένη εικόνα με όλα τα διαθέσιμα μαθήματα προς δήλωση για εξέταση.

Τα μαθήματα που έχουν επιλεγεί ήδη από τον σπουδαστή απεικονίζονται στη φόρμα ως:



Εικόνα 17: Φόρμα Δηλωμένων Μαθήματων Φοιτητή

Ας δούμε όμως και τον κώδικα της φόρμας:

```
namespace Thanos.Reports {
public partial class StudentsCoursesReportForm : Form {
public StudentsCoursesReportForm() {
   InitializeComponent();
   user = MainForm.user;
   GetDeclaredCourses();
}
bool IsFormLoading = true;
Users user;
DataBaseActions db = MainForm.db;
Messages msg = new Messages();
DataTable NonDeclaredCourse = new DataTable();
DataTable DeclaredCourse = new DataTable();

private void GetNonDeclaredCourses() {
NonDeclaredCourse = db.GetDataIntoTable(" SELECT " +
                                        "    c.ID " +
                                        " FROM courses c " +
                                        " JOIN dbo.CourseName cn " +
                                        "    ON cn.ID = c.eCourseNameID " +
                                        " JOIN dbo.ePeriods p " +
                                        "    ON  p.ID = c.ePeriodsID " +
                                        " JOIN dbo.Users u " +
                                        "    ON u.ID = c.eUserID " +
                                        " JOIN eAcademicYears ay " +
                                        "    ON ay.ID = c.eAcademicYearID " +
                                        " LEFT JOIN dbo.StudentsCourses sc " +
                                        "    ON sc.eCoursesID = c.ID " +
                                        " WHERE cast(GETDATE() AS date) BETWEEN eStartDate AND eEndDate " +
                                        " AND cn.ID NOT IN (SELECT " +
                                        "                                    CourseName.ID " +
                                        "                                    FROM dbo.StudentsCourses " +
                                        "                                    JOIN dbo.Courses " +
                                        "                                    ON Courses.ID = StudentsCourses.eCoursesID " +
                                        "                                    JOIN dbo.CourseName " +
                                        "                                    ON CourseName.ID = Courses.eCourseNameID " +
                                        "                                    WHERE StudentsCourses.eUserID = '" + user.ID + "')");

}
private void GetDeclaredCourses() {
DeclaredCourse = db.GetDataIntoTable(" SELECT " +
                                     "    sc.ID " +
                                     "    , cn.Name CourseName " +
                                     "    , p.PeriodName + '(' + ay.AcademicYear + ')' Period " +
                                     "    , u.Surname + ' ' + u.Name Teacher " +
                                     "    , u.email " +
                                     " FROM courses c " +
                                     " JOIN dbo.CourseName cn " +
                                     "    ON cn.ID = c.eCourseNameID " +
```

```
                        " JOIN dbo.ePeriods p " +
                        "    ON  p.ID = c.ePeriodsID " +
                        " JOIN dbo.Users u " +
                        "    ON u.ID = c.eUserID " +
                        " JOIN eAcademicYears ay " +
                        "    ON ay.ID = c.eAcademicYearID " +
                        " JOIN dbo.StudentsCourses sc " +
                        "    ON sc.eCoursesID = c.ID " +
                        " WHERE sc.eUserID = '" + user.ID + "'");
gcStudentsCourses.DataSource = DeclaredCourse;
}
private void FixDeclaredCourse() {
   Dictionary<string, string> Translate = new Dictionary<string, string>();
   Translate.Add("CourseName", "Όν. Μαθήματος");
   Translate.Add("Period", "Περίοδος");
   Translate.Add("Teacher", "Καθηγητής");
   Translate.Add("email", "e-Mail");
   gvStudentsCourses.ColumnsTranslate(Translate);
   gvStudentsCourses.HideColumns();
   gvStudentsCourses.OptionsBehavior.ReadOnly = true;
   gvStudentsCourses.OptionsBehavior.Editable = false;
}
private void StudentsCoursesReportForm_Load(object sender, EventArgs e) {
   try {
      DeclareCourses(IsFormLoading);
      FixDeclaredCourse();
   } catch (Exception ex) { msg.Error(ex.Message); }
}
private void btnDelete_Click(object sender, EventArgs e) {
   try {
        gvStudentsCourses.DeleteSelectedRow(typeof(StudentsCourses), GetDeclaredCourses);
   } catch (Exception ex) { msg.Error(ex.Message); }
}
private void btnAdd_Click(object sender, EventArgs e) {
   try {
      IsFormLoading = false;
      DeclareCourses(IsFormLoading);
   } catch (Exception ex) { msg.Error(ex.Message); }
}
private void DeclareCourses(bool IsFormLoading) {
   try {
      GetNonDeclaredCourses();
      if (IsFormLoading) {
         if (NonDeclaredCourse.Rows.Count > 0) {
            var dr = msg.QuestionYesNo("Βρέθηκαν ενεργά μαθήματα προς δήλωση. \nΘέλετε να κάνετε δήλωση τώρα;");
            if (dr == DialogResult.Yes) {
               MassDeclareCoursesForm nForm = new MassDeclareCoursesForm();
               nForm.ShowDialog(this);
            }
            GetDeclaredCourses();
         }
      }else{
         if (NonDeclaredCourse.Rows.Count > 0) {
               MassDeclareCoursesForm nForm = new MassDeclareCoursesForm();
               nForm.ShowDialog(this);
               GetDeclaredCourses();
         } else {
            msg.Information("Δεν βρέθηκαν ενεργά μαθήματα προς δήλωση");
         }
      }
   }catch(Exception ex) { msg.Error(ex.Message); }
}
```

```
private void panel2_Paint(object sender, PaintEventArgs e)
{
}
}
}
```

Ξεκινώντας, στη φόρμα καλούμε την κλάση GetDeclaredCourses που όπως θα δούμε παρακάτω φέρνει στη φόρμα τα μαθήματα που έχουμε ήδη δηλώσει και είναι προς εξέταση.

Ακολουθεί η κλάση GetNonDeclaredCourses η οποία αντλεί το id από τη βάση δεδομένων SQL από πίνακα Courses που γίνεται join με τους πίνακες CourseName, ePeriods, Users, eAcademicYears και StudentsCourses με κριτήριο την ημερομηνία δήλωσης μαθήματος που έχει θέσει ο καθηγητής καθώς και το να μην υπάρχει ήδη δηλωμένο το μάθημα στα μαθήματα του σπουδαστή. Όπως θα δούμε αυτό το id το οποίο εισάγεται στο DataTable NonDeclaredCourse θα χρησιμοποιηθεί παρακάτω.

Έπειτα η κλάση GetDeclaredCourses αντλεί από τη βάση και εισάγει στο DataTable DeclaredCourse τα στοιχεία των δηλωμένων μαθημάτων του σπουδαστή (Ον. Μαθήματος, Περίοδος, Ονοματεπώνυμο Καθηγητή και email Καθηγητή). Εδώ εισάγει και τα στοιχεία με DataSource κατευθείαν στη φόρμα.

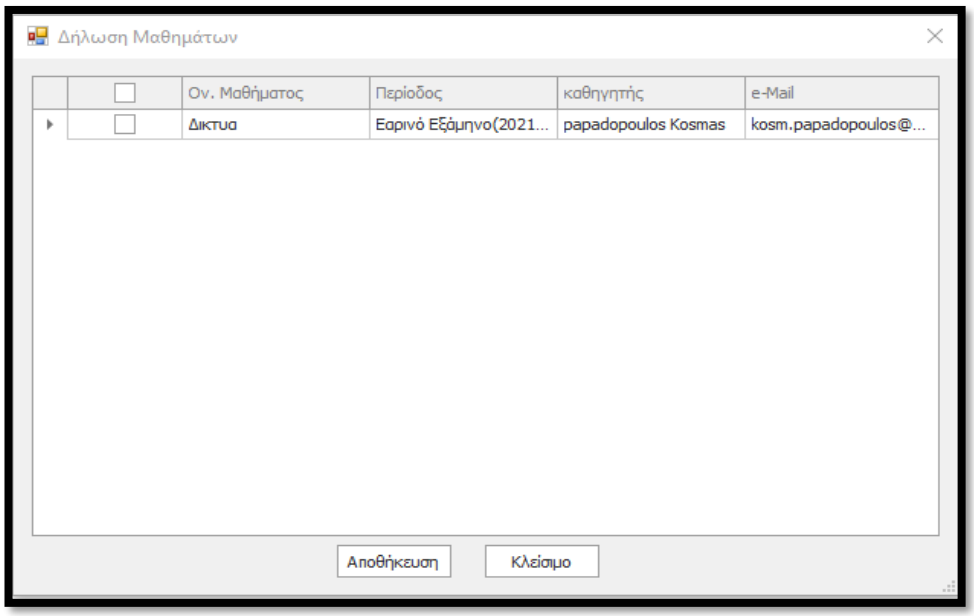
Εν συνεχεία, έχουμε την κλάση FixDeclaredCourse όπου χρησιμοποιώντας Dictionary μεταφράζει τις στήλες της φόρμας στα Ελληνικά, καθώς και κρύβει περιττές στήλες που δεν καλούνται στη φόρμα. Επίσης και εδώ κάνει τις στήλες read only και non editable.

Μετά έχουμε τη κλάση StudentsCoursesReportForm_Load όπου απλά φορτώνει την φόρμα κατά την επιλογή και καλεί την FixDeclaredCourse για να κάνει μετάφραση των στηλών. Τα κουμπιά Delete και Add όπου αντίστοιχα σβήνουν ένα μάθημα ή επαναφέρουν τη φόρμα ώστε να δηλώσουμε κάποιο μάθημα εκ νέου.

Τέλος, έχουμε τη κλάση DeclareCourses η οποία χρησιμοποιώντας τη Boolean μεταβλητή IsFormLoading ελέγχει ποσά μαθήματα επιστρέφει η κλάση GetNonDeclaredCourses και αν είναι μεγαλύτερο από 0 τότε βγάζει μήνυμα στον σπουδαστή «Βρέθηκαν ενεργά μαθήματα προς δήλωση. Θέλετε να κάνετε δήλωση τώρα;». Εάν επιλέξουμε ναι, τότε καλεί την φόρμα δηλώσεις νέων μαθήματων MassDeclareCoursesForm. Στην περίπτωση else η οποία συμβαίνει μόνο αν κάποιος διαγράψει μάθημα και θέλει να ξαναδηλώσει πατώντας το Add button ξαναφορτώνει τη φόρμα MassDeclareCoursesForm ώστε να επιλέξει το μάθημα εκ νέου. Σε

περίπτωση που δεν υπάρχουν μαθήματα προς δήλωση πατώντας το κουμπί Add βγάζει μήνυμα «Δεν βρέθηκαν ενεργά μαθήματα προς δήλωση».

Παρακάτω θα δούμε την φόρμα MassDeclareCoursesForm. Είναι η φόρμα που βγαίνει όταν πατάμε το Add ή επιλέγουμε Ναι στο pop up για τη δήλωση νέων μαθημάτων προς εξέταση. Η φόρμα εμφανίζεται ως εξής:



Εικόνα 18: Φόρμα Μαθήματων Προς Δήλωση από Φοιτητή

Ας δούμε τώρα και τον κώδικα της φόρμας MassDeclareCoursesForm:

```
namespace Thanos.Reports {
public partial class MassDeclareCoursesForm : Form {
public MassDeclareCoursesForm() {
   InitializeComponent();
   user = MainForm.user;
   GetUnDeclaredCourse();
   FixGrid();
}
DataBaseActions db = MainForm.db;
DataTable UnDeclaredCourse = new DataTable();
Messages msg = new Messages();
Users user;
Int32[] SelectedRows;

private void FixGrid() {
   Dictionary<string, string> Translates = new Dictionary<string, string>();
   Translates.Add("CourseName", "Ον. Μαθήματος");
   Translates.Add("Teacher", "καθηγητής");
   Translates.Add("Period", "Περίοδος");
   Translates.Add("email", "e-Mail");
```

```
   gvUnDeclaredCourses.ColumnsTranslate(Translates);
   gvUnDeclaredCourses.HideColumns(new string[] { "ID", "CourseNameID" });
   gvUnDeclaredCourses.OptionsSelection.MultiSelect = true;
      gvUnDeclaredCourses.OptionsSelection.MultiSelectMode =
      DevExpress.XtraGrid.Views.Grid.GridMultiSelectMode.CheckBoxRowSelect;
   gvUnDeclaredCourses.OptionsBehavior.Editable = false;
   gvUnDeclaredCourses.SelectionChanged += GvUnDeclaredCourses_SelectionChanged;

}
private void GetUnDeclaredCourse() {
UnDeclaredCourse = db.GetDataIntoTable(" SELECT " +
                          "       c.ID " +
                          "     ,cn.Name CourseName " +
                          "       , p.PeriodName + '(' + ay.AcademicYear + ')' Period " +
                          "     ,u.Surname + ' ' + u.Name[Teacher] " +
                          "       ,u.email " +
                          "    ,cn.ID CourseNameID" +
                          " FROM courses c " +
                          " JOIN dbo.CourseName cn " +
                          "       ON cn.ID = c.eCourseNameID " +
                          " JOIN dbo.ePeriods p " +
                          "       ON p.ID = c.ePeriodsID " +
                          " JOIN dbo.Users u " +
                          "       ON u.ID = c.eUserID " +
                          " JOIN eAcademicYears ay " +
                          "       ON ay.ID = c.eAcademicYearID " +
                          " LEFT JOIN dbo.StudentsCourses sc " +
                          "       ON sc.eCoursesID = c.ID " +
                          " WHERE cast(GETDATE() AS date) BETWEEN eStartDate AND eEndDate " +
                          " AND cn.ID NOT IN (SELECT " +
                          "                                       CourseName.ID " +
                          "                                       FROM dbo.StudentsCourses " +
                          "                                       JOIN dbo.Courses " +
                          "                                       ON Courses.ID = StudentsCourses.eCoursesID " +
                          "                                      JOIN dbo.CourseName " +
                          "                                      ON CourseName.ID = Courses.eCourseNameID " +
                          "                                      WHERE StudentsCourses.eUserID = '" + user.ID + "')");
gcUnDeclaredCourses.DataSource = UnDeclaredCourse;
gvUnDeclaredCourses.RefreshData();
}
private void btnSave_Click(object sender, EventArgs e) {
try {
   var dr = msg.QuestionYesNo("θέλετε να αποθηκεύσετε τα επιλεγμένα μαθήματα;");
if (dr == DialogResult.Yes) {
      Int32[] SelectedRows = gvUnDeclaredCourses.GetSelectedRows();

      List<Guid> Guids = new List<Guid>();
      bool Exists = false;
      if (SelectedRows.Count() > 0) {
         foreach (int i in SelectedRows) {
            var g = Guid.Parse(gvUnDeclaredCourses.GetRowCellValue(i, "CourseNameID").ToString());
            if (!Guids.Contains(g)) {
               Guids.Add(g);
            } else {
               Exists = true;
               break;
            }
         }
      }
      if(!Exists){
         foreach (int i in SelectedRows) {
            StudentsCourses studentsCourses = new StudentsCourses();
```

```
            studentsCourses.eCoursesID = Guid.Parse(gvUnDeclaredCourses.GetRowCellValue(i, "ID").ToString());
            studentsCourses.eUserID = user.ID;
            studentsCourses.eTimeStamp = DateTime.Now;
            studentsCourses.Save();
          }
          msg.Information("Η δήλωση μαθημάτων ολοκληρώθηκε επιτυχώς!");
          this.Close();
        } else{
          msg.Error("Δεν μπορείτε να επιλέξετε το ίδιο μάθημα από διαφορετικό καθηγητή!");
        }
      }
    } catch (Exception ex) {msg.Error(ex.Message); }
    }
    private void btnClose_Click(object sender, EventArgs e) {
      this.Close();
    }
    private void GvUnDeclaredCourses_SelectionChanged(object sender, DevExpress.Data.SelectionChangedEventArgs e) {
    try {
      if (e.Action == CollectionChangeAction.Add) {
        Int32[] CurSelectedRows = gvUnDeclaredCourses.GetSelectedRows();
        List<Guid> Guids = new List<Guid>();
        bool Exists = false;
        if (CurSelectedRows.Count() > 0) {
          foreach (int i in CurSelectedRows) {
            var g = Guid.Parse(gvUnDeclaredCourses.GetRowCellValue(i, "CourseNameID").ToString());
            if (!Guids.Contains(g)) {
              Guids.Add(g);
            } else {
              Exists = true;
              break;
            }
          }
        }
        if (Exists) {
          gvUnDeclaredCourses.ClearSelection();
          if(SelectedRows != null){
            foreach (int i in SelectedRows) {
              gvUnDeclaredCourses.SelectRow(i);
            }
          }
          msg.Error("Δεν μπορείτε να επιλέξετε το ίδιο μάθημα από διαφορετικό καθηγητή!");
        } else {
          SelectedRows = CurSelectedRows;
        }
      }
    }catch{ }
    }
    private void gcUnDeclaredCourses_Click(object sender, EventArgs e)
    {
    }
    private void MassDeclareCoursesForm_Load(object sender, EventArgs e)
    {
    }
    }
    }
```

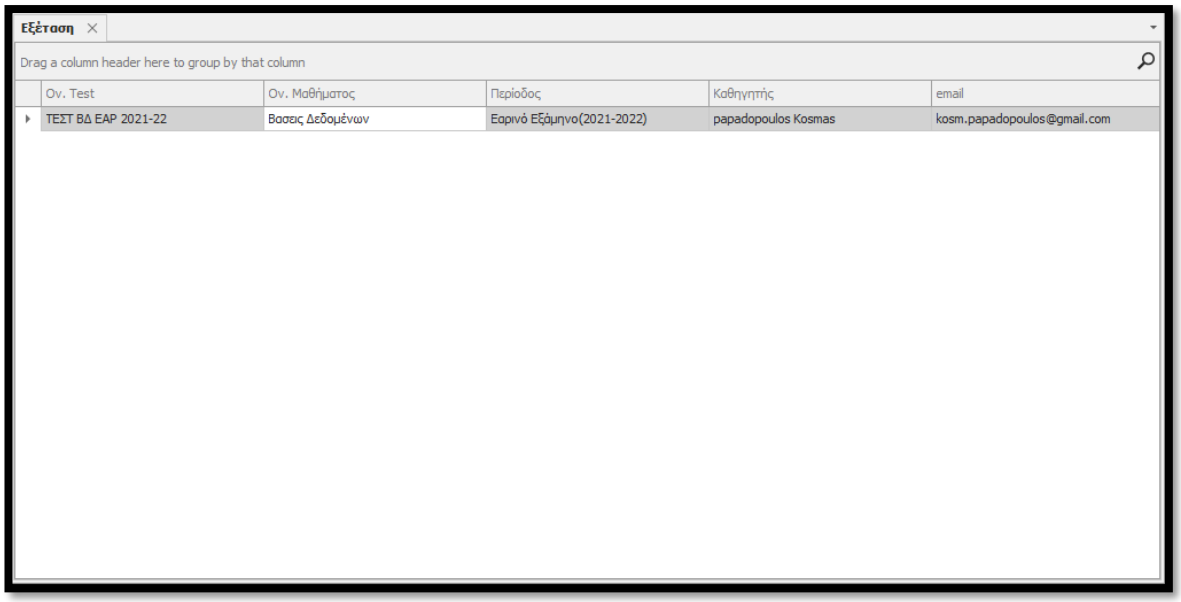
Στην παρούσα φόρμα, χρησιμοποιείται η κλάση GetUnDeclaredCourse για να εμφανίσει στο σπουδαστή τα διαθέσιμα μαθήματα προς εξέταση που δεν έχει δηλώσει.

Ακολουθεί η κλάση FixGrid η οποία μεταφράζει και εδώ τις στήλες της φόρμας στα Ελληνικά, και κρύβει τις στήλες ID και CourseNameID. Επίσης, στην περίπτωση αυτή δίνει την δυνατότητα επιλογής πολλαπλών γραμμών, με το MultiSelectMode επιτρέπει την επιλογή γραμμών και checkbox για τα μαθήματα. Επίσης υπάρχει και φίλτρο για την περίπτωση που κάποιος προσπαθήσει να διαλέξει το ίδιο μάθημα από άλλο καθηγητή ενώ το έχει ήδη δηλώσει.

Παρακάτω, η κλάση GetUnDeclaredCourse αντλεί όλα τα απαραίτητα στοιχεία από τη βάση δεδομένων και τα εμφανίζει μέσω DataSource στη φόρμα απευθείας. Αντλεί το ID του μαθήματος από τον πίνακα Courses, όνομα μαθήματος Coursename από αντίστοιχο πίνακα, περίοδο και ακαδημαϊκό έτος από τους πίνακες ePeriods και eAcademicYears για τα αντίστοιχα μαθήματα που έχουν δηλωθεί από τους καθηγητές προς εξέταση στον πίνακα StudentCourses. Γίνεται έλεγχος ότι η περίοδος δήλωσης είναι έγκυρη και ότι δεν υπάρχει ήδη δηλωμένο το μάθημα.

Έπειτα, οι κλάσεις Save και GvUnDeclaredCourses_SelectionChanged, η πρώτη ελέγχει κατά την επιλογή μαθημάτων προς εξέταση αν υπάρχουν ήδη, αν δεν υπάρχουν προχωράει βγάζοντας την ερώτηση «Θέλετε να αποθηκεύσετε τα επιλεγμένα μαθήματα;» αν πατήσουμε το ναι βγάζει το μήνυμα «Η δήλωση μαθημάτων ολοκληρώθηκε επιτυχώς». Αν όμως υπάρχει ήδη το μάθημα δηλωμένο από άλλο καθηγητή μας βγάζει το μήνυμα λάθους «Δεν μπορείτε να επιλέξετε το ίδιο μάθημα από διαφορετικό καθηγητή». Αντίστοιχα το ίδιο κάνει και η GvUnDeclaredCourses_SelectionChanged αλλά όταν στην υπάρχουσα φόρμα γίνονται αλλαγές, όπως όταν επιλεγεί μάθημα ή διαγραφεί και προσπαθήσουμε να το επιλέξουμε ξανά ή αν υπάρχει ήδη από άλλο καθηγητή.

Στην επόμενη ενότητα θα δούμε τη φόρμα TestParticipationReportform που είναι η φόρμα εξέτασης των σπουδαστών. Ας δούμε μια εικόνα της φόρμας.



Εικόνα 19: Φόρμα Εξέτασης Μαθημάτων από Φοιτητή

Όπως βλέπουμε είναι η φόρμα της εξέτασης όπου υπάρχει η λίστα με τα διαθέσιμα διαγωνίσματα κατηγοριοποιημένα με Όνομα διαγωνίσματος, μαθήματος, Περίοδο, Ονοματεπώνυμο καθηγητή καθώς και mail επικοινωνίας. Ο κώδικας της φόρμας ακολουθεί:

```
namespace Thanos.Reports {
public partial class TestParticipationReportForm : Form {
   public TestParticipationReportForm() {
      InitializeComponent();
      user = MainForm.user;
      GridRefresh();
      FixGridView();
   }
Users user;
DataBaseActions db = MainForm.db;
Messages msgs = new Messages();

private void GridRefresh() {
gcParticipationTests.DataSource = db.GetDataIntoTable("SELECT " +
                                    "    t.ID " +
                                    "    , t.TestName " +
                                    "    , cn.Name " +
                                    "    , p.PeriodName + '(' + ay.AcademicYear + ')'[Period] " +
                                    "    , u.Surname + ' ' + u.Name[Teacher] " +
                                    "    , u.email " +
                                    "FROM dbo.StudentsCourses sc " +
                                    "JOIN dbo.Courses c " +
                                    "    ON c.ID = sc.eCoursesID " +
                                    "JOIN dbo.CourseName cn " +
                                    "    ON cn.ID = c.eCourseNameID " +
                                    "JOIN dbo.Tests t " +
                                    "    ON t.eCoursesID = sc.eCoursesID " +
                                    "JOIN dbo.Users u " +
                                    "    ON u.ID = c.eUserID " +
                                    "JOIN dbo.ePeriods p " +
```

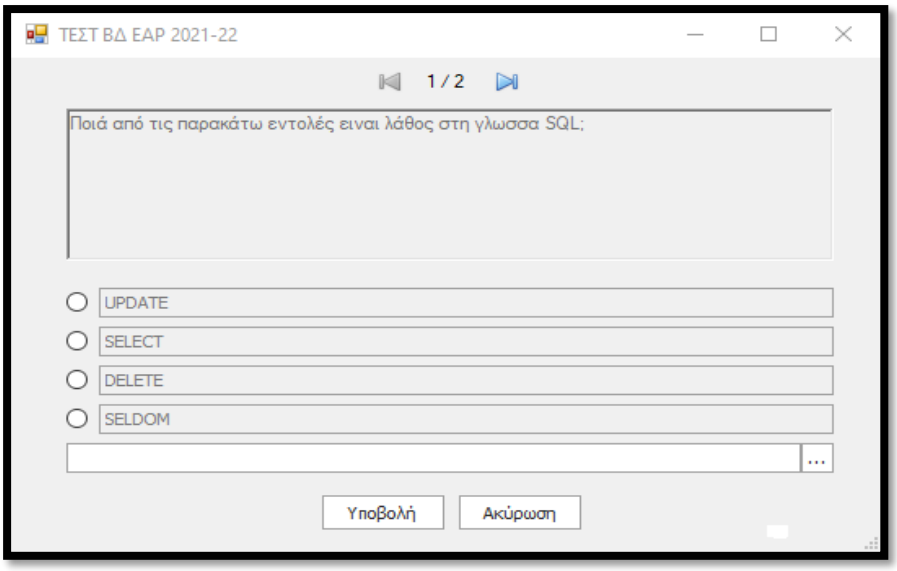
```
                                    "    ON p.ID = c.ePeriodsID " +
                                    "JOIN dbo.eAcademicYears ay " +
                                    "    ON ay.ID = c.eAcademicYearID " +
                                    "WHERE t.Published = 1 " +
                                    "AND sc.eUserID = '"+user.ID+"' " +
                                    "AND t.ID NOT IN (SELECT distinct " +
                                    "                 eTestID " +
                                    "               FROM StudentsAnswers " +
                                    "               WHERE eUserID = '"+ user.ID + "')");
    }
    private void FixGridView() {
      Dictionary<string, string> Translate = new Dictionary<string, string>();
      Translate.Add("Name", "Ον. Μαθήματος");
      Translate.Add("TestName", "Ον. Test");
      Translate.Add("Period", "Περίοδος");
      Translate.Add("Teacher", "Καθηγητής");
      gvParticipationTests.ColumnsTranslate(Translate);
      gvParticipationTests.HideColumns();
      gvParticipationTests.OptionsBehavior.ReadOnly = true;
      gvParticipationTests.OptionsBehavior.Editable = false;
    }
    private void gvParticipationTests_DoubleClick(object sender, EventArgs e) {
      try {
        gvParticipationTests.EditOnDoubleClick(this, typeof(Tests), typeof(StudentsTestEntryForm), GridRefresh);
      }catch(Exception ex) { msgs.Error(ex.Message); }
    }
    private void panel2_Paint(object sender, PaintEventArgs e)
    {
    }
    }
    }
```

Ξεκινάμε τη φόρμα με το GridRefresh όπου φέρνει με DataSource άμεσα στη φόρμα τα απαραίτητα στοιχεία από τη βάση δεδομένων. ID, όνομα Test, περίοδο, ονοματεπώνυμο καθηγητή και το email του. Αντλεί τα δεδομένα με join των πινάκων StudentCourses, Courses, CourseName, Tests, Users, ePeriods, eAcademicyears και ελέγχει ότι ο καθηγητής χέει κάνει δημοσίευση το διαγώνισμα (t.Published = 1) και ότι το userID που έχει ο σπουδαστής αντιστοιχεί με αυτό στη βάση στον πίνακα StudentCourses καθώς και ότι δεν υπάρχει ήδη το ID του test στα απαντημένα διαγωνίσματα στον πίνακα StudentAnswers.

Παρακάτω, έχουμε τη κλάση FixGridView όπου μεταφράζει τις εμφανιζόμενες στήλες, κρύβει τις υπόλοιπες και τις κάνει read only και non editable.

Τέλος, έχουμε την κλάση gvParticipationTests_DoubleClick ή οποία καλεί την φόρμα StudentsTestEntryForm για να δώσει ο σπουδαστής το μάθημα.

Η φόρμα StudentsTestEntryForm, η οποία είναι για την απάντηση των ερωτήσεων του διαγωνίσματος, απεικονίζεται ως εξής:



Εικόνα 20: Φόρμα Απάντησης Ερωτήματος από Φοιτητή

Ο σπουδαστής λαμβάνει με τη σειρά ερωτήσεις με πολλαπλής επιλογής απαντήσεις και από κάτω μπορεί να επισυνάψει ένα video λίγων δευτερολέπτων για την επεξήγηση της απάντησης.

Ας δούμε τον κώδικα της φόρμας.

```
namespace Thanos.EntriesForms {
public partial class StudentsTestEntryForm : Form {
public StudentsTestEntryForm(Tests obj) {
   InitializeComponent();
   user = MainForm.user;
   test = obj;
   user = MainForm.user;
   string qListOfTestQuestions = "SELECT * from dbo.TestQuestions WHERE eTestID = '" + test.ID + "'";
   ListOfTestQuestions =
db.GetDataIntoTable(qListOfTestQuestions).TurnToListOfObjects(typeof(TestQuestions)).Cast<TestQuestions>().ToList();
   FillLists();
   Question = ListOfQuestions[1];
   this.Text = test.TestName;
   UpdateControls();

}
Users user;
Tests test;
DataBaseActions db = MainForm.db;
List<TestQuestions> ListOfTestQuestions = new List<TestQuestions>();
Questions Question;
StudentsAnswers Answer;
List<Questions> ListOfQuestions = new List<Questions>();
List<StudentsAnswers> ListOfAnswers = new List<StudentsAnswers>();
Messages msg = new Messages();
int SelectedQuestion = 1;

private void FillLists() {
   foreach (TestQuestions tq in ListOfTestQuestions) {
      Questions q = new Questions();
```

```
            q.GetByID(tq.eQuestionID);
            ListOfQuestions.Add(q);
            StudentsAnswers stAnswers = new StudentsAnswers();
            stAnswers.eTestID = tq.eTestID;
            stAnswers.eTestQuestionID = tq.ID;
            stAnswers.eUserID = user.ID;
            stAnswers.eQuestionID = q.ID;
            ListOfAnswers.Add(stAnswers);
        }
    }
    private void UpdateControls() {
        string LabelText = "{0} / {1}";
        UpdateAnswers();
        label1.Text = string.Format(LabelText, SelectedQuestion.ToString(), ListOfQuestions.Count.ToString());
        if (SelectedQuestion == 1) {
            btnPrevius.Enabled = false;
            btnNext.Focus();
        } else { btnPrevius.Enabled = true; }
        if (SelectedQuestion == ListOfQuestions.Count) {
            btnNext.Enabled = false;
            btnPrevius.Focus();
        } else { btnNext.Enabled = true; }
        FillControlsFromObjects();
    }
    private void UpdateAnswers(){
        Answer = ListOfAnswers.First(a => a.eQuestionID == Question.ID);
        this.Fill(Answer);
    }
    private void FillControlsFromObjects() {
        Question = ListOfQuestions[SelectedQuestion - 1];
        Answer = ListOfAnswers.First(a => a.eQuestionID == Question.ID);
        Question.Fill(this);
        Answer.Fill(this);
        txtPath.Enabled = Question.AllowUploadFile;
        btnNavigate.Enabled = Question.AllowUploadFile;
        txtPath.Text = Answer.FilePath;

    }
    private void btnPrevius_Click(object sender, EventArgs e) {
        try {
            if (SelectedQuestion != 1) {
                SelectedQuestion = SelectedQuestion - 1;
                UpdateControls();
            }
        } catch (Exception ex) { msg.Error(ex.Message); }
    }
    private void btnNext_Click(object sender, EventArgs e) {
        try {
            if (SelectedQuestion != ListOfQuestions.Count) {
                SelectedQuestion = SelectedQuestion + 1;

                UpdateControls();
            }
        } catch (Exception ex) { msg.Error(ex.Message); }
    }
    private void btnSubbmit_Click(object sender, EventArgs e) {
        try {
            var dr = msg.QuestionYesNo("ΠΡΟΣΟΧΗ! Με την υποβολή του Test δε μπορείτε να αλλάξετε τις απαντήσεις
            σας.\nΘέλετε να κάνετε ολοκλήρωση του Test;");
            if (dr == DialogResult.Yes) {
                foreach (StudentsAnswers obj in ListOfAnswers) {
                if(!string.IsNullOrEmpty(obj.FilePath)){
```

```
                FsAnswers fs = new FsAnswers();
                fs.eStudentID = obj.eUserID;
                fs.StudentAnswerID = obj.ID;
                fs.FileName = obj.FileName;
                fs.FilePath = obj.FilePath;
                fs.Save();
                obj.eFsAnswersID = fs.ID;
            }
              obj.Save();
            }
            this.Close();
          }
      } catch (Exception ex) { msg.Error(ex.Message); }
    }
    private void btnCancel_Click(object sender, EventArgs e) {
      this.Close();
    }
    private void btnNavigate_Click(object sender, EventArgs e) {
      OpenFileDialog ofd = new OpenFileDialog();
      if(ofd.ShowDialog() == DialogResult.OK){
        txtPath.Text = ofd.FileName;
      }
    }
    private void txtPath_EditValueChanged(object sender, EventArgs e) {
      Answer.FilePath = txtPath.Text;
      Answer.FileName = Path.GetFileName(txtPath.Text);
    }
    private void StudentsTestEntryForm_Load(object sender, EventArgs e)
    {
    }
    }
    }
```

Αρχικά, η φόρμα StudentsTestEntryForm τραβάει στη μεταβλητή ListOfTestQuestions τις ερωτήσεις του αντίστοιχου διαγωνίσματος που έχει επιλεχθεί.

Αμέσως μετά, δηλώνονται οι μεταβλητές που θα χρησιμοποιηθούν στις επόμενες κλάσεις. Η μεταβλητή SelectedQuestion παίρνει την τιμή 1.

Ακολουθεί η κλάση FillLists, η οποία για κάθε ερώτηση που απαντάται από τον σπουδαστή αποθηκεύεται στη λίστα ListOfAnswers.

Η κλάση που ακολουθεί, UpdateControls, χρησιμεύει ώστε να κάνει ενεργά ή ανενεργά τα κουμπιά του προηγουμένου/επομένου στο διαγώνισμα. Όταν η τιμή του SelectedQuestion είναι 1 τότε το προηγούμενο είναι ανενεργό καθώς πρόκειται για την πρώτη ερώτηση και όταν το SelectedQuestion ισούται με το σύνολο των ερωτήσεων του διαγωνίσματος τότε το επόμενο είναι ανενεργό καθώς έχουμε φτάσει στην τελευταία ερώτηση.

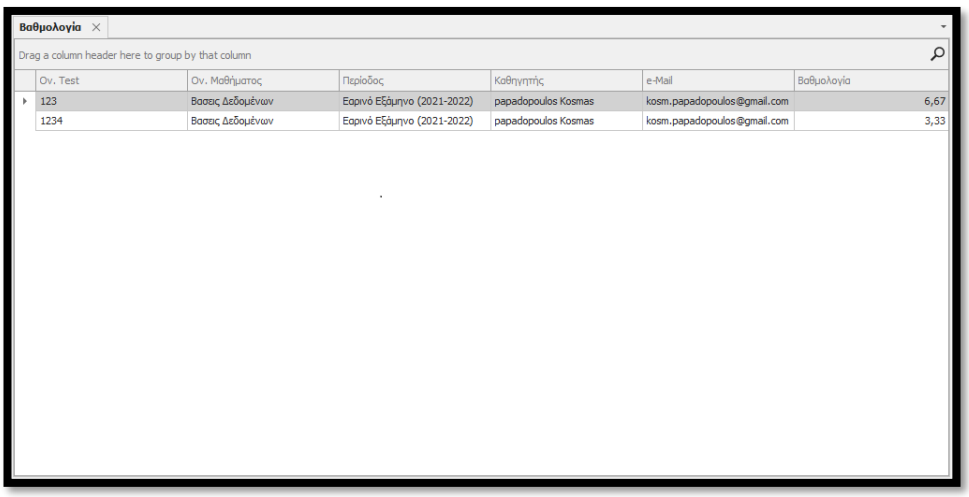
Έπειτα, η κλάση UpdateAnswers, χρησιμοποιείται όταν ο σπουδαστής αλλάζει ή επιλεγεί μια απάντηση στο διαγώνισμα.

Η επόμενη κλάση FillControlFromObjects, ενημερώνει τα πεδία των ερωτήσεων απαντήσεων και το αν το ανέβασμα αρχείου είναι ενεργοποιημένο για την εκάστοτε ερώτηση όταν ο φοιτητής λαμβάνει την ερώτηση.

Οι υπόλοιπες κλάσεις είναι τα κουμπιά της φόρμας, Previous, Next, Submit, Cancel, Navigate και EditValue. Τα Previous/Next είναι για να μετάβουμε σε προηγουμένη ή επόμενη αντίστοιχα ερώτηση. Το Submit υποβάλλει και καταχωρεί το διαγώνισμα βγάζοντας το προειδοποιητικό μήνυμα «ΠΡΟΣΟΧΗ! Με την υποβολή του Test δε μπορείτε να αλλάξετε τις απαντήσεις σας. Θέλετε να κάνετε ολοκλήρωση του Test;» . Αν πατήσουμε ναι προχωράει αλλιώς μας επιστρέφει στη φόρμα του διαγωνίσματος για να συνεχίσουμε. Το Cancel κλείνει τη φόρμα, ωστόσο μπορούμε να την ξανανοίξουμε αργότερα. Το Navigate μας αφήνει να επιλέξουμε αρχείο video για να ανεβάσουμε για την απάντηση. Τέλος το EditValue μας επιτρέπει να αλλάξουμε με το χέρι το πεδίο του φακέλου που βρίσκεται το αρχείο.

Τέλος, έχουμε τη φόρμα Βαθμολόγησης που λαμβάνουν οι σπουδαστές η οποία είναι η CompletedTestReportForm. Στη φόρμα αυτή ο φοιτητής λαμβάνει τα στοιχεία ονόματος διαγωνίσματος και ονόματος μαθήματος που έδωσε, καθώς και την περίοδο που διεξήχθη αλλά και τα στοιχεία του καθηγητή (ονοματεπώνυμο και e-mail) και βέβαια τη βαθμολογία που πηρέ στο εκάστοτε διαγώνισμα.

Παρακάτω, η εικόνα της φόρμας.



Εικόνα 21: Φόρμα Βαθμολογίας των Ολοκληρωμένων Διαγωνισμάτων

Ας δούμε και τον κώδικα της φόρμας της Βαθμολογίας.

```
namespace Thanos.Reports {
public partial class CompletedTestReportForm : Form {
public CompletedTestReportForm() {
   InitializeComponent();
   user = MainForm.user;
   GridRefresh();
   FixGridView();
}
Users user;
DataBaseActions db = MainForm.db;
Messages msgs = new Messages();

private void GridRefresh() {
gcCompletedTests.DataSource = db.GetDataIntoTable
                    ("SELECT " +
                     "    t.ID " +
                     "    , t.TestName " +
                     "    , cn.Name " +
                     "    , p.PeriodName + ' (' + ay.AcademicYear + ')' Period " +
                     "    , ut.Surname + ' ' + ut.Name [Teacher] " +
                     "    , ut.email [e-Mail] " +
                     "    , sg.Grade " +
                     " FROM Tests t " +
                     " JOIN dbo.Courses c " +
                     "    ON c.ID = t.eCoursesID " +
                     " JOIN dbo.CourseName cn " +
                     "    ON cn.ID = c.eCourseNameID " +
                     " JOIN dbo.ePeriods p " +
                     "    ON  p.ID = c.ePeriodsID " +
                     " JOIN dbo.eAcademicYears ay " +
                     "    ON ay.ID = c.eAcademicYearID " +
                     " JOIN(SELECT DISTINCT eTestID, eUserID FROM dbo.StudentsAnswers) sta ON t.ID = sta.eTestID " +
                     " JOIN users u " +
                     "    ON u.ID = sta.eUserID " +
                     " JOIN StudentsGrading sg " +
                     "    ON u.ID = sg.StudentID AND sg.eTestID = t.ID"+
                     " JOIN users ut " +
                     "    ON ut.ID = sg.TeacherID ");
}
private void FixGridView() {
   Dictionary<string, string> Translate = new Dictionary<string, string>();
   Translate.Add("Name", "Ον. Μαθήματος");
   Translate.Add("TestName", "Ον. Test");
   Translate.Add("Period", "Περίοδος");
   Translate.Add("Teacher", "Καθηγητής");
   Translate.Add("Grade", "Βαθμολογία");
   gvCompletedTests.ColumnsTranslate(Translate);
   gvCompletedTests.HideColumns();
   gvCompletedTests.OptionsBehavior.ReadOnly = true;
   gvCompletedTests.OptionsBehavior.Editable = false;
}
private void panel2_Paint(object sender, PaintEventArgs e)
{
}
}
}
```

Η φόρμα αυτή είναι πολύ απλή, έχει μόνο την κλάση GridRefresh, η οποία με χρήση DataSource εντολής εισάγει τα δεδομένα που θέλουμε από τη βάση δεδομένων απευθείας στη φόρμα. Τραβάει με select το id και όνομα του

διαγωνίσματος από τον πίνακα Test και τα συνδέει με join με τους πίνακες Courses και CourseName από τους οποίους αντλεί το όνομα μαθήματος, από τους πίνακες ePeriods και eAcademicYears αντλεί την περίοδο (χειμερινό/εαρινό και έτος περιόδου), από τον πίνακα Users αντλεί το ονοματεπώνυμο και e-mail του καθηγητή και από τον StudentsGrading τον βαθμό που έχει καταχωρηθεί στον σπουδαστή.

Τέλος, έχει την κλάση FixGridView που κάνει μετάφραση τις επιλεγμένες στήλες, κρύβει τις υπόλοιπες και τις κάνει read only και non-editable.

6 Συμπεράσματα

Από τη δημιουργία της παραπάνω εφαρμογής και τις αναλύσεις των χρησιμοποιούμενων εργαλείων, προκύπτουν κάποιες σημαντικές διαπιστώσεις και παρατηρήσεις. Πρώτον, η διεξαγωγή των διαδικτυακών εξετάσεων είναι ακόμα σε πρώιμο στάδιο και χρήζουν πολλών βελτιώσεων. Δεύτερον, θα ήταν χρήσιμο να εξετάσουμε τους τρόπους βελτίωσης των συνθήκων των διαδικτυακών εξετάσεων ώστε να διευκολυνθούν και οι εξεταστές αλλά και οι εξεταζόμενοι. Τρίτον, κρίνεται απαραίτητη η δυνατότητα εξέτασης με μεγαλύτερη ακρίβεια, δηλαδή να μπορούν οι εξεταστές να αναγνωρίζουν αν οι εξεταζόμενοι όντως καταλαβαίνουν το εκάστοτε μάθημα και την εξεταζόμενη ύλη. Η χρήση μιας εφαρμογής όπως αυτή που παρουσιάσαμε, θα βελτίωνε δραστικά την εμπειρία της εξέτασης των φοιτητών και θα βοηθούσε να βαθμολογήσουν πιο αξιοκρατικά αλλά και πιο ευκολά οι καθηγητές τις απαντήσεις των διαγωνισμάτων.

7 Βιβλιογραφία

[1] G. O. I. a. Ş. Oruç, «Effect of the use of multimedia on students' performance: A case study of social studies class,» *AcademicJournals,* τόμ. 11, αρ. 8, p. 878, 4 March 2016.

[2] A. G. a. D. Mahapatra, «INTERACTIVE MEDIA AND ITS IMPACT ON EDUCATION,» *International Journal of Current Research,* τόμ. 8, αρ. 12, p. 43964, 30 December 2016.

[3] Wikipedia, «Wikipedia,» Wikimedia, 20 April 2022. [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Microsoft_SQL_Server. [Πρόσβαση 25 May 2022].

[4] Wikipedia, «ACID,» WikiMedia, 2018. [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/ACID. [Πρόσβαση 25 January 2022].

[5] S. E. S. Sumathi, «Fundamentals of Relational Database Management Systems,» σε *Fundamentals of Relational Database Management Systems*, Springer Science & Business Media, 2007, p. 114.

[6] Microsoft, «Microsoft Build,» Microsoft, 4 1 2022. [Ηλεκτρονικό]. Available: https://docs.microsoft.com/en-us/analysis-services/analysis-services-overview?view=asallproducts-allversions&redirectedfrom=MSDN&viewFallbackFrom=sql-server-ver15. [Πρόσβαση 18 4 2022].

[7] Microsoft, «Microsoft Build,» Microsoft, 20 1 2022. [Ηλεκτρονικό]. Available: https://docs.microsoft.com/en-us/sql/reporting-services/create-deploy-and-manage-mobile-and-paginated-reports?view=sql-server-ver15&redirectedfrom=MSDN&viewFallbackFrom=sql-server-2014. [Πρόσβαση 18 4 2022].

[8] Microsoft, «Microsoft Build,» Microsoft, 29 11 2021. [Ηλεκτρονικό]. Available: https://docs.microsoft.com/en-us/sql/relational-databases/blob/filestream-sql-server?view=sql-server-ver15#recommendations-and-guidelines-for-improving-filestream-performance. [Πρόσβαση 23 3 2022].

[9] Wikipedia, «Wikipedia,» Wikimedia, 2022. [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Microsoft_Visual_Studio. [Πρόσβαση 18 4 2022].

[10] Wikipedia, «Wikipedia,» Wikimedia, 2022. [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/C_Sharp_(programming_language). [Πρόσβαση 18 4 2022].

[11] Wikipedia, «Wikipedia,» Wikimedia, 2022. [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Windows_Forms. [Πρόσβαση 18 4 2022].

[12] Microsoft, «Microsoft,» Microsoft, 2022. [Ηλεκτρονικό]. Available: https://docs.microsoft.com/en-us/dotnet/desktop/winforms/overview/?view=netdesktop-6.0. [Πρόσβαση 18 4 2022].

Προτάσεις Επέκτασης Εργασίας:

- Δυνατότητα εισαγωγής ειδοποιήσεων στους φοιτητές για λήψη διαγωνισμάτων/δήλωση μαθήματων
- Πιθανή αυτοματοποίηση εγγραφής Καθηγητών από το σύστημα, ακόμα ίσως και των φοιτητών.
- Δυνατότητα ενσωμάτωσης σε πρόγραμμα απομακρυσμένης εξέτασης μέσω Video, οπού θα μπορούσαν να συμπεριληφθούν πολλαπλές δυνατότητες που δε προσφέρουν οι σημερινές πλατφόρμες όπως η δυνατότητα να μην ακούν οι εξεταζόμενοι ο ένας τον άλλον κατά τη διάρκεια της διαδικτυακής εξέτασης αλλά μόνο τον καθηγητή.
- Δυνατότητα μαζικής εισαγωγής ερωτήσεων από Excel.
- Δυνατότητα περιορισμού των καθηγητών σχετικά με ποια μαθήματα μπορούν να δηλώσουν προς εξέταση. Θα πρέπει να μπορούν να δηλώσουν μόνο τα δικά τους.
- Προσθήκη από πριν των μαθημάτων που υπάρχουν στο εκάστοτε Πανεπιστήμιο και δημιουργία προεπιλεγμένης λίστας για τους καθηγητές και τα μαθήματα που θα μπορούν να δηλώσουν προς εξέταση.
- Αυτόματη προσθήκη λίστας και λογαριασμών φοιτητών στο σύστημα ώστε να μπορεί να γίνει έλεγχος αν ο εκάστοτε φοιτητής δικαιούται να εξεταστεί στο μάθημα.