Δ Ι Ε Θ Ν Ε Σ Π Α Ν Ε Π Ι Σ Τ Η Μ Ι Ο Τ Η Σ Ε Λ Λ Α Δ Ο Σ
Σ Χ Ο Λ Η Μ Η Χ Α Ν Ι Κ Ω Ν
Τ Μ Η Μ Α Μ Η Χ Α Ν Ι Κ Ω Ν Π Λ Η Ρ Ο Φ Ο Ρ Ι Κ Η Σ ,
Υ Π Ο Λ Ο Γ Ι Σ Τ Ω Ν Κ Α Ι Τ Η Λ Ε Π Ι Κ Ο Ι Ν Ω Ν Ι Ω Ν

ΨΗΦΙΟΠΟΙΗΣΗ ΤΟΥ ΑΡΧΕΙΑΚΟΥ ΥΛΙΚΟΥ ΤΟΥ 1^{ΟΥ} ΓΕΝΙΚΟΥ ΛΥΚΕΙΟΥ ΣΕΡΡΩΝ

Πτυχιακή Εργασία των

Αλέξανδρου Κάτσακα (3934)

Μιχαήλ Τζεμίντιμπη (2923)

Επιβλέπων: Δρ. Ευάγγελος Δημητριάδης

ΣΕΡΡΕΣ, ΑΠΡΙΛΙΟΣ 2022

Υπεύθυνη Δήλωση *: Βεβαιώνουμε ότι είμαστε συγγραφείς αυτής της πτυχιακής εργασίας και ότι κάθε βοήθεια την οποία είχαμε για την προετοιμασία της, είναι πλήρως αναγνωρισμένη και αναφέρεται στην πτυχιακή εργασία. Επίσης έχουμε αναφέρει τις όποιες πηγές από τις οποίες κάναμε χρήση δεδομένων, ιδεών ή λέξεων, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επίσης, βεβαιώνουμε ότι αυτή η πτυχιακή εργασία προετοιμάστηκε από εμάς προσωπικά ειδικά για τις απαιτήσεις του προγράμματος σπουδών του Τμήματος Μηχανικών Πληροφορικής, Υπολογιστών και Τηλεπικοινωνιών του Διεθνούς Πανεπιστημίου της Ελλάδας.*

Περίληψη

Ο στόχος της συγκεκριμένης πτυχιακής εργασίας ήταν η ψηφιοποίηση του αρχειακού υλικού του 1ου ΓΕΛ Σερρών και πιο συγκεκριμένα του μεγάλου όγκου απολυτηρίων που έχουν μαζευτεί στο αρχείο του σχολείου τα οποία είναι πολύ δύσκολο και χρονοβόρο για το προσωπικό να βρει όταν κάποιος απόφοιτος το ζητήσει. Για να διευκολύνουμε τους καθηγητές του σχολείου σαρώσαμε τα απολυτήρια και στη συνέχεια δημιουργήσαμε μια εφαρμογή η οποία επιτρέπει στο προσωπικό του σχολείου να βρει αμέσως το απολυτήριο που χρειάζεται χρησιμοποιώντας το ονοματεπώνυμο, το έτος αποφοίτησης ή το σχολείο αποφοίτησης του μαθητή και να τα μετατρέψει σε PDF ή να εκτυπώσει απευθείας το απολυτήριο, ενώ καταγράφεται και ο φάκελος στον οποίο βρίσκεται στην περίπτωση που ο απόφοιτος επιθυμεί το επίσημο έγγραφο έναντι της εκτύπωσης.

Περιεχόμενα

Πίνακας εικόνων

Εισαγωγή

Το πρόβλημα που κληθήκαμε να αντιμετωπίσουμε στην παρούσα πτυχιακή εργασία ήταν ο μεγάλος όγκος απολυτηρίων που βρίσκονται στο αρχείο του 1ου Γενικού Λυκείου Σερρών και καθιστά την εύρεση ενός απολυτηρίου δύσκολη αλλά και χρονοβόρα για το προσωπικό του σχολείου όταν αυτό ζητηθεί. Για να λύσουμε αυτό το πρόβλημα έπρεπε να σαρώσουμε περίπου 9.000 απολυτήρια και στη συνέχεια να δημιουργήσουμε μια εφαρμογή μέσα από την οποία οι καθηγητές του σχολείου μπορούν εύκολα και γρήγορα να έχουν πρόσβαση στο απολυτήριο που χρειάζονται. Σημαντικοί παράγοντες που έπρεπε να λάβουμε υπόψιν ήταν η ασφάλεια των προσωπικών δεδομένων των μαθητών και η δημιουργία μιας εφαρμογής ικανής να λειτουργήσει στους αρκετά αδύναμους υπολογιστές του λυκείου ανεξάρτητα από το μέγεθος της βάσης δεδομένων της αλλά και του λειτουργικού συστήματος του κάθε υπολογιστή. Για να πετύχουμε τα παραπάνω σχεδιάσαμε την εφαρμογή σε γλώσσα **C**# και τη βάση δεδομένων σε localDB με αποτέλεσμα οι ανάγκες σε μνήμη RAM να μην ξεπερνούν ποτέ τα 15 Megabyte ανεξάρτητα από το μέγεθος της βάσης η οποία ξεπερνά προς το παρόν τα 16 Gigabyte και θα μεγαλώνει όσο περνάει ο καιρός και αυξάνονται οι εγγραφές. Επίσης η εφαρμογή μπορεί να τρέξει σε κάθε λειτουργικό σύστημα της Microsoft από windows xp μέχρι windows 11 ενώ σε περίπτωση που θα χρειαστεί αναβάθμιση της εφαρμογής σε μεταγενέστερα windows αρκεί να γίνει compile της εφαρμογής από το visual studio εκείνης της περιόδου και θα αναβαθμιστεί αυτόματα το .NET Framework στην τελευταία έκδοσή του. Ένα άλλο πλεονέκτημα της εφαρμογής είναι πως σε περίπτωση καταστροφής της βάσης δεδομένων ή της ίδιας της εφαρμογής οι φωτογραφίες αποθηκεύονται στο File System της εφαρμογής στο φάκελο Images με ονοματεπώνυμο και χρονολογία αποφοίτησης ώστε να μπορούν εύκολα να το βρουν και να το εκτυπώσουν μέσα από μια αναζήτηση στα windows ενώ λόγω του τρόπου λειτουργίας της βάσης, η εφαρμογή είναι πολύ εύκολο να μεταφερθεί σε άλλο υπολογιστή αρκεί κάποιος να αντιγράψει τον φάκελο της εφαρμογής.

1. ΕΡΓΑΛΕΙΑ ΠΟΥ ΧΡΗΣΙΜΟΠΟΙΗΘΗΚΑΝ

***Εικόνα 1*:** Λογότυπο του Microsoft Visual Studio

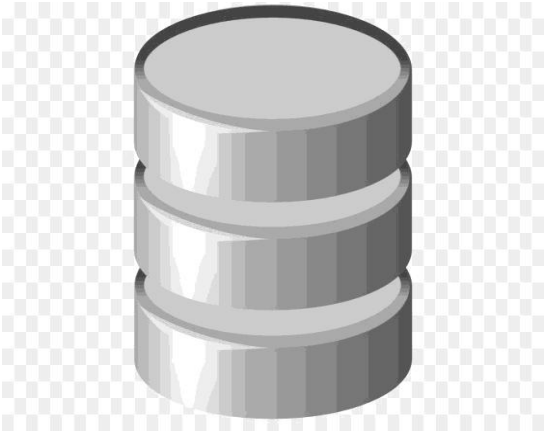
1.1 Το **Microsoft Visual Studio** είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) από τη Microsoft. Χρησιμοποιείται για την ανάπτυξη προγραμμάτων υπολογιστών, καθώς και ιστοσελίδων, εφαρμογών ιστού, υπηρεσιών ιστού και εφαρμογών για κινητά. Το Visual Studio χρησιμοποιεί πλατφόρμες ανάπτυξης λογισμικού της Microsoft όπως το Windows API, τα Windows Forms, το Windows Presentation Foundation, το Windows Store και το Microsoft Silverlight. Μπορεί να παράγει τόσο εγγενή όσο και διαχειριζόμενο κώδικα.

***Εικόνα 2*:** Λογότυπο του Eclipse

1.2 Το Eclipse είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης που χρησιμοποιείται στον προγραμματισμό υπολογιστών. Περιέχει έναν βασικό χώρο εργασίας και ένα επεκτάσιμο σύστημα plug-in για την προσαρμογή του περιβάλλοντος. Είναι γραμμένο κυρίως σε Java και χρησιμοποιείται κυρίως για ανάπτυξη εφαρμογών Java αλλά μπορεί να χρησιμοποιηθεί για την ανάπτυξη εφαρμογών και σε άλλες γλώσσες προγραμματισμού μέσω προσθηκών όπως είναι για παράδειγμα οι Ada, ABAP, C, C++, C#, Clojure, COBOL, PHP, JavaScript, Python και άλλες.

***Εικόνα 3*:** Λογότυπο του SQL Server

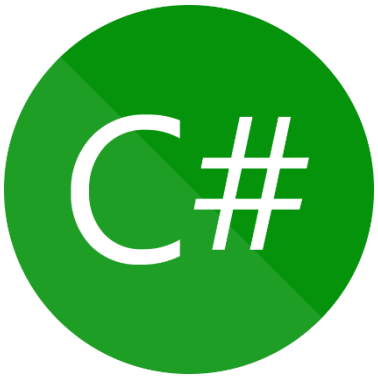
1.3 Ο SQL Server είναι μια σχεσιακή βάση δεδομένων η οποία αναπτύσσεται από τη Microsoft. Βγήκε για πρώτη φορά στην αγορά το 1989 σε συνεργασία με την Sybase και οι κύριες γλώσσες που χρησιμοποιούνται είναι η T-SQL και η ANSI SQL. Η κύρια μονάδα αποθήκευσης στοιχείων είναι μια βάση δεδομένων, η οποία αποτελείται από μια συλλογή πινάκων και κώδικα. Η κεντρική βάση δεδομένων του SQL Server υποστηρίζει διαφορετικούς τύπους, συμπεριλαμβανομένων των ακεραίων αριθμών, αριθμών κινητής υποδιαστολής, δεκαδικών, αλφαριθμητικών, Varchar, δυαδικών αριθμών (για τα μη δομημένα δεδομένα) και κειμένων, ενώ επιτρέπει και καθορισμένους από το χρήστη σύνθετους τύπους δεδομένων (UDTs), δηλαδή τύπους που βασίζονται στους βασικούς τύπους αλλά μπορούν να τροποποιηθούν.



***Εικόνα 4*:** Λογότυπο του DB Browser

1.4 Το DB Browser είναι ένα εργαλείο ανοιχτού κώδικα που χρησιμοποιείται για την σχεδίαση, τη δημιουργία και την επεξεργασία αρχείων βάσεων δεδομένων συμβατών με SQLite. Το χρησιμοποιήσαμε στις δύο πρώτες εφαρμογές που δημιουργήσαμε αν και καταλήξαμε στην τελική εφαρμογή να χρησιμοποιούμαι τον SQL Server καθώς η βάση δεδομένων γράφτηκε σε LocalDB.

2. ΤΕΧΝΟΛΟΓΙΕΣ ΠΟΥ ΧΡΗΣΙΜΟΠΟΙΗΘΗΚΑΝ



***Εικόνα 5*: Λογότυπο της C#**

2.1 Η **C#** (C Sharp) είναι μια γλώσσα προγραμματισμού Η/Υ. Δημιουργήθηκε από την Microsoft μέσα από την πλατφόρμα .NET και αργότερα αναγνωρίστηκε επισήμως από την Ecma (ECMA-334) και την ISO (ISO/IEC 23270:2018). Είναι μια από τις γλώσσες προγραμματισμού που δημιουργήθηκαν για την Κοινή Υποδομή Γλώσσας (Common Language Infrastructure). Ο κύριος σκοπός της γλώσσας είναι να είναι απλή αντικειμενοστρεφής γλώσσα για γενική χρήση.

***Εικόνα 6*:** Λογότυπο της SQLite

Το **SQLite** είναι ένα σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων το οποίο περιέχεται σε μια προγραμματιστική βιβλιοθήκη της C. Δημιουργήθηκε το 2000 από τον D. Richard Hipp κατά τη συνεργασία του με το πολεμικό ναυτικό των Ηνωμένων Πολιτειών της Αμερικής. Ανήκει στην οικογένεια των ενσωματωμένων βάσεων δεδομένων και η σύνταξή του ακολουθεί το πρότυπο PostgreSQL. Είναι συμβατό με τις περισσότερες γλώσσες προγραμματισμού καθώς και με τα περισσότερα προγράμματα περιήγησης.

***Εικόνα 7*:** Λογότυπο του LocalDB

2.3 Το Microsoft SQL Server Express LocalDB αποτελεί μία λειτουργία του SQL Server Express ιδιαίτερα χρήσιμη στους προγραμματιστές. Παρέχεται δωρεάν από τη Microsoft για την ανάπτυξη εφαρμογών όπως και το SQL Server Express με την κύρια μεταξύ τους διαφορά να είναι πως το LocalDB μπορεί να χρησιμοποιηθεί σαν ενσωματωμένη βάση δεδομένων για μικρότερες εφαρμογές ενώ το SQL Server Express αποτελεί μια πιο ισχυρή και πλήρη απομακρυσμένη μηχανή βάσεων δεδομένων που χρησιμοποιείται για μεγαλύτερες εφαρμογές.

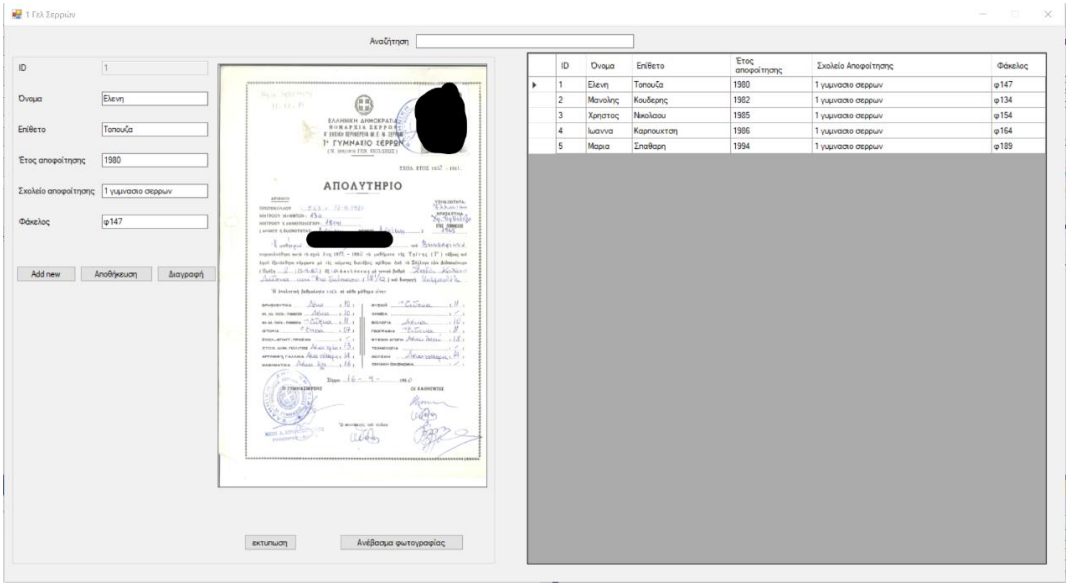
ΨΗΦΙΟΠΟΙΗΣΗ

Το πρώτο μέρος αυτής της εργασίας είναι η ψηφιοποίηση του αρχειακού υλικού του 1ου ΓΕΛ. Με το πέρασμα των χρόνων το 1ο λύκειο έχει μαζέψει έναν μεγάλο αριθμό απολυτηρίων καθώς μέχρι και την δεκαετία του 80 τα απολυτήρια όλων των σχολείων της πόλης και πολλών σχολείων από κοντινά χωριά του νομού Σερρών αποθηκεύονταν στο αρχείο του 1ου λυκείου. Για να εξυπηρετήσουμε τις ανάγκες του σχολείου αλλά και των παλιών αποφοίτων ξεκινήσαμε από το έτος 1970 και σαρώσαμε όλα τα απολυτήρια μέχρι και το 2000, όταν ξεκίνησε η επίσημη ψηφιοποίηση των αρχείων από το υπουργείο παιδείας. Η σάρωση των περίπου 9000 απολυτηρίων έγινε με τη χρήση του φωτοαντιγραφικού Ricoh Aficio MP C3502 που μας δόθηκε από το σχολείο με την επιλεγμένη ανάλυση στα 600 dpi και τύπο πρωτοτύπου ασπρόμαυρο κείμενο/σχέδιο ώστε να έχουμε την καθαρότερη δυνατή εικόνα. Το συγκεκριμένο φωτοαντιγραφικό μας έδωσε τη δυνατότητα να αποθηκεύουμε τις εικόνες σε εξωτερικό μέσο αποθήκευσης (κάρτα microSD) σε μορφή jpeg ενώ η ταχύτητα σάρωσης που δεν ξεπερνούσε τα 10 δευτερόλεπτα ανά σελίδα βελτίωσε πολύ τον χρόνο ολοκλήρωσης της όλης διαδικασίας η οποία πέρα από χρονοβόρα αποδείχτηκε αρκετά επώδυνη για τους μύες της μέσης και θα παροτρύναμε οποιονδήποτε αναλάβει κάποια αντίστοιχη εργασία να αφήνει μία κενή μέρα ανάμεσα στις μέρες σάρωσης. Ο συνολικός χρόνος σάρωσης ενός φακέλου 200 σελίδων που συμπεριλαμβάνει την ίδια τη σάρωση, την αποθήκευση στην κάρτα microSD και την δημιουργία αντιγράφων ασφαλείας σε υπολογιστές του σχολείου κυμαίνεται ανάμεσα σε 70-90 λεπτά που σημαίνει ότι αναλόγως του μεγέθους των φακέλων μπορούσαμε να ολοκληρώσουμε δύο με τρεις φακέλους τη μέρα και λαμβάνοντας υπόψιν τις κενές ενδιάμεσες μέρες η όλη διαδικασία της σάρωσης διήρκεσε περίπου 6 βδομάδες.

ΠΡΩΤΕΣ ΠΡΟΣΠΑΘΕΙΕΣ

Η πρώτη εφαρμογή που δημιουργήσαμε γράφτηκε σε JAVA μέσα από το περιβάλλον ανάπτυξης Eclipse ενώ η βάση δεδομένων δημιουργήθηκε σε SQLite. Παρότι η εφαρμογή ήταν γρήγορη και αρκετά εύχρηστη αναγκαστήκαμε γρήγορα να την παρατήσουμε, καθώς ο υπολογιστής που μας διατέθηκε από το σχολείο για να εγκαταστήσουμε την εφαρμογή ήταν χαμηλών δυνατοτήτων και η εφαρμογή θα ήταν αδύνατο να τρέξει όσο θα αυξανόταν το μέγεθος της βάσης δεδομένων.

Η δεύτερη εφαρμογή είχε γραφτεί σε C#(C sharp) με χρήση του Visual studio 2019 community και η βάση δεδομένων της σε SQLite. Στη συγκεκριμένη εφαρμογή λύθηκε το κύριο πρόβλημα της πρώτης εφαρμογής καθώς οι απαιτήσεις της σε μνήμη RAM ήταν πολύ χαμηλότερες αλλά το γραφικό περιβάλλον (εικόνα 8) δεν κρίθηκε εύχρηστο από το προσωπικό του σχολείου.



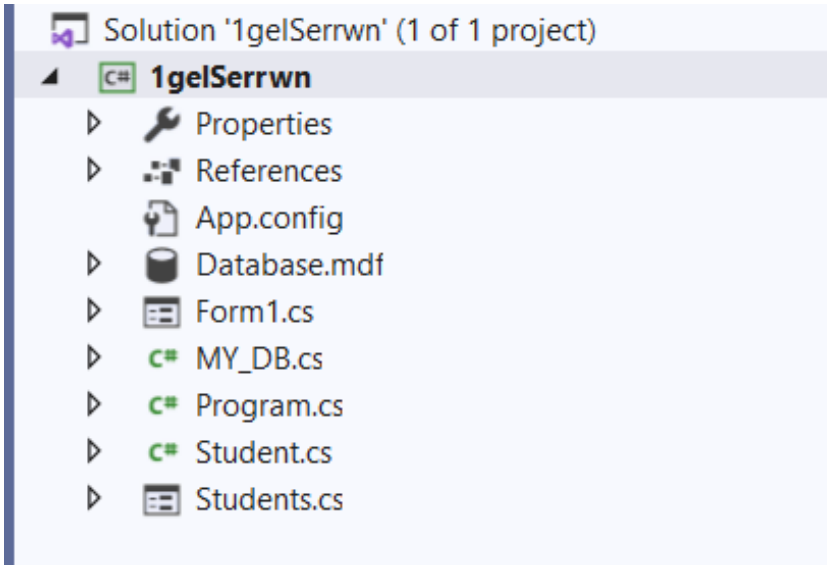
***Εικόνα 8*:** Γραφικό περιβάλλον της δεύτερης εφαρμογής

Η τελική έκδοση της εφαρμογής υλοποιήθηκε σε C# με τη βάση δεδομένων σε localDB της οποίας οι ανάγκες σε μνήμη RAM δεν ξεπερνούν τα 15 Megabyte ανεξάρτητα από το μέγεθος της βάσης ενώ παράλληλα μπορεί να τρέξει σε κάθε λειτουργικό σύστημα της Microsoft από windows xp μέχρι windows 11 αλλά και σε μεταγενέστερα windows αρκεί να γίνει compile της εφαρμογής από το visual studio εκείνης της περιόδου και θα αναβαθμιστεί αυτόματα το .NET Framework στην τελευταία έκδοσή του.

3. ΤΕΧΝΙΚΗ ΑΝΑΛΥΣΗ

3.1 ΚΛΑΣΕΙΣ

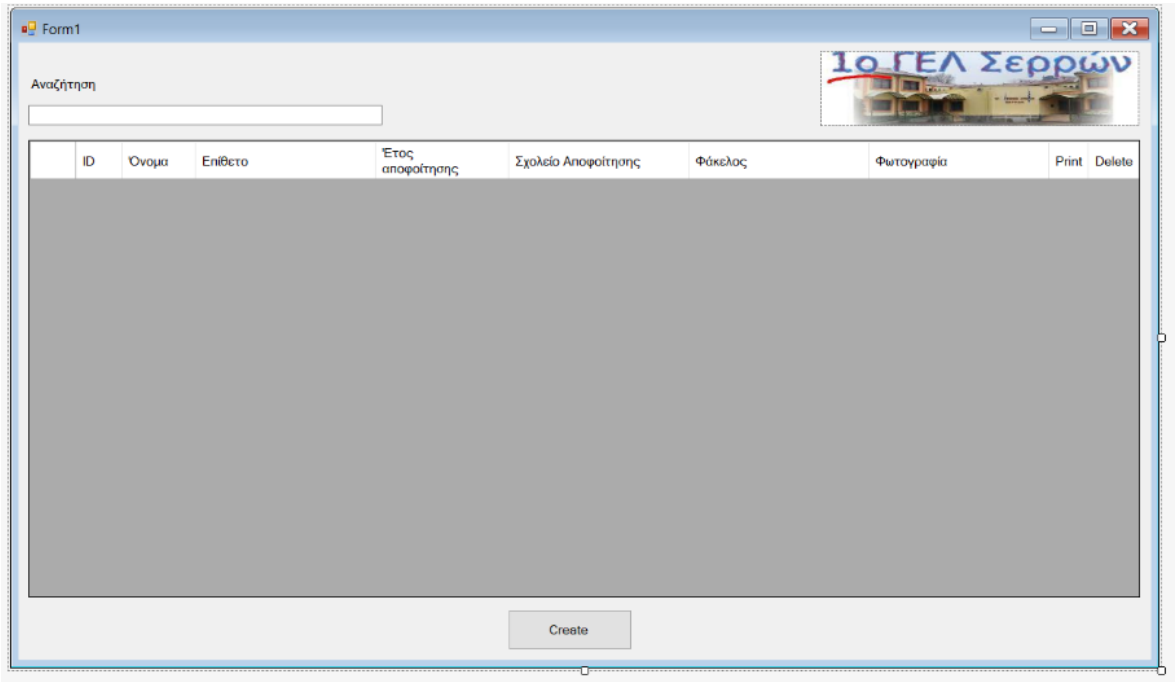
Στην εικόνα 9 βλέπουμε όλες τις κλάσεις του προγράμματος μαζί και τη βάση δεδομένων .



***Εικόνα 9*:** Κλάσεις και βάση δεδομένων

3.2 Η ΦΟΡΜΑ Form1

Η Form1 είναι η πρώτη και βασική φόρμα του συστήματος. Εδώ πέρα εκτελούνται η αναζήτηση , η διαγραφή και η εκτύπωση των εγγραφών.



***Εικόνα 10*:** Η φόρμα FORM1

3.3 ΒΑΣΗ ΔΕΔΟΜΕΝΩΝ

Μετά από τη δημιουργία της βασικής φόρμας έχουμε τη δημιουργία της βάσης δεδομένων η οποία περιλαμβάνει Id, όνομα, επίθετο, χρονιά αποφοίτησης, σχολείο αποφοίτησης, φάκελο και φωτογραφία του αποφοίτου στον οποίο ανήκει το απολυτήριο. Η βάση είναι γραμμένη σε sql και είναι αποθηκευμένη στο αρχείο Database.mdf .

	Name	Data Type	Allow Nulls	Default
🔑	Id	int	☐	
	FirstName	nvarchar(30)	☐	
	LastName	nvarchar(30)	☐	
	GraduationYear	int	☐	
	GraduationSchool	nvarchar(50)	☐	
	Folder	nvarchar(10)	☐	
	Photo	nvarchar(MAX)	☐	
			☐	

Design | T-SQL

```
1  CREATE TABLE [dbo].[Students] (
2      [Id]               INT            IDENTITY (1, 1) NOT NULL,
3      [FirstName]        NVARCHAR (30)  NOT NULL,
4      [LastName]         NVARCHAR (30)  NOT NULL,
5      [GraduationYear]   INT            NOT NULL,
6      [GraduationSchool] NVARCHAR (50)  NOT NULL,
7      [Folder]           NVARCHAR (10)  NOT NULL,
8      [Photo]            NVARCHAR (MAX) NOT NULL,
9      PRIMARY KEY CLUSTERED ([Id] ASC)
10 );
11
12
```

Εικόνα 11: Βάση δεδομένων

3.3.1 ΚΛΑΣΗ MY_DB

Η κλάση MY_DB.cs ασχολείται με την σύνδεση της βάσης δεδομένων με το πρόγραμμα και την κατάσταση στην οποία βρίσκεται η βάση δεδομένων.

Η μεταβλητή conn διαχειρίζεται την σύνδεση με τη βάση και ορίζει σε ποια θέση υπάρχει το αρχείο της βάσης Database.mdf ώστε να μπορέσει να επιτευχθεί η σύνδεση.

Η μέθοδος SqlConnection επιστρέφει την θέση της βάσης δεδομένων .

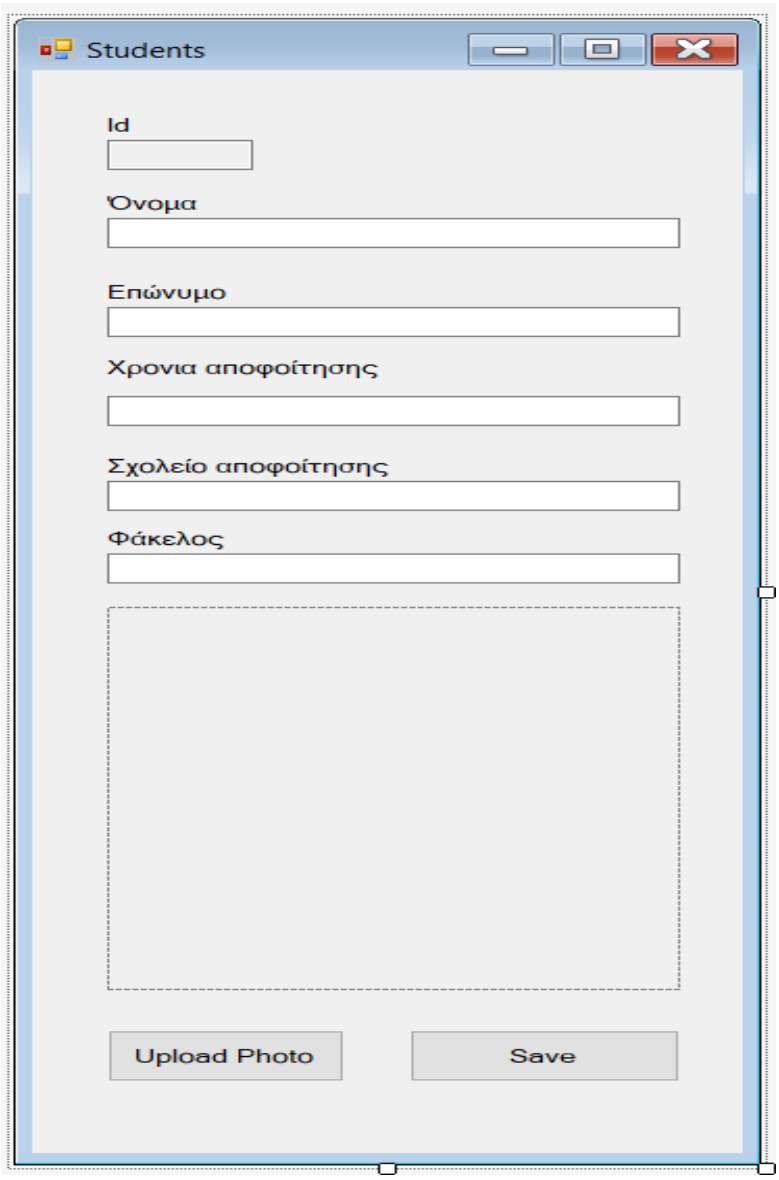
Η μέθοδος openConnection() ελέγχει αν η σύνδεση με τη βάση είναι κλειστή και την ανοίγει .

Η μέθοδος closeConnection() ελέγχει αν η σύνδεση με τη βάση είναι ανοιχτή και την κλείνει .

```
namespace _1gelSerrwn
{
   class MY_DB
   {
     private  SqlConnection conn = new SqlConnection(@"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=|DataDirectory|\Database.m
df;Integrated Security=True");

       public SqlConnection GetConnection
       {
          get
          {
             return conn;
          }
       }

       public void openConnection()
       {
          if (conn.State==System.Data.ConnectionState.Closed)
          {
             conn.Open();
          }
       }

       public void closeConnection()
       {
          if (conn.State == System.Data.ConnectionState.Open)
          {
             conn.Close();
          }
       }
   }
}
```

3.4 ΦΟΡΜΑ Students

Στην εικόνα 13 βλέπουμε την φόρμα Students.cs η οποία είναι υπεύθυνη για την δημιουργία χρηστών ή την αλλαγή στοιχείων για τους υπάρχοντες χρήστες.



***Εικόνα 12*:** Η Φόρμα Students

Στην κλάση Students οι βασικές μεταβλητές είναι η filepath η οποία μηδενίζεται στην αρχή, είναι τύπου String και είναι το μέρος όπου στο τέλος αποθηκεύονται οι φωτογραφίες του χρήστη. Η imagepath η οποία είναι public ώστε να μπορεί να επικοινωνεί με τις άλλες φόρμες είναι σε μορφή string και παίρνει την θέση της φωτογραφίας του επιλεγμένου χρήστη ώστε να μπορεί να την επεξεργαστεί στο μέλλον.

```
public Students()
      {
         InitializeComponent();
      }

      string filepath=null;
      public string imagepath;
```

3.4.1 ΜΕΘΟΔΟΣ saveb

Η μέθοδος saveb είναι τύπου bool και ελέγχει εαν τα πεδία τιμών των Ονομα(Nametb),Επίθετο(Lnametb),Χρονολογία(Yeartb),Σχολείο(Schtb), Φάκελος(Folder) και Φωτογραφία(Picturetb) είναι κενά και αν είναι επιστρέφει την τιμή false αλλιώς την τιμή true.

```
bool saveb()
      {
         if ((Nametb.Text.Trim() == "") || (Lnametb.Text.Trim() == "") ||
(Yeartb.Text.Trim() == "") || (Schtb.Text.Trim() == "") || (Foldertb.Text.Trim() == "")
|| (Picturetb.Image == null))
         {
            return false;
         }
         else
         {
            return true;
         }
      }
```

3.4.2 ΜΕΘΟΔΟΣ Savebt_Click

Η μέθοδος Savebt_Click εκτελείται όταν ο χρήστης πατήσει το κουμπί αποθήκευση στη φόρμα. Η μεταβλητή student είναι τύπου Student και θα χρησιμοποιηθεί αργότερα για την αποθήκευση ή επεξεργασία του χρήστη. Πρώτα η συνθήκη ελέγχει αν το κείμενο του κουμπιού είναι ίδιο με το Save ή το Update. Αν το κείμενο είναι ίδιο με το Save τότε ελέγχει για αρχή αν όλα τα πεδία είναι κενά μέσω της συνάρτησης saveb. Αν είναι κενά βγάζει το μήνυμα ότι ο χρήστης πρέπει να γεμίσει τα πεδία, αλλιώς δημιουργούνται 7 μεταβλητές εκ των οποίων οι 6 είναι τύπου string και η 1 τύπου int για να αποθηκεύσει τις τιμές από τα textbox. Η μεταβλητή fullname συνδυάζει τα πεδία fname, lname και year χρησιμοποιώντας το _ και το επίθεμα .jpg για να αποθηκεύσει το καινούργιο όνομα της φωτογραφίας του χρήστη. Στην μεταβλητή path τύπου string αποθηκεύεται η διαδρομή του γονικού καταλόγου της εφαρμογής. Μετά το πρόγραμμα ελέγχει αν υπάρχει στη διαδρομή του γονικού καταλόγου της εφαρμογής ο φάκελος Images ο οποίος αν δεν υπάρχει δημιουργείται στην επόμενη εντολή. Εφόσον πια υπάρχει ο φάκελος Images το πρόγραμμα αντιγράφει τη διαδρομή του φακέλου μαζί με όνομα του αρχείου στο αρχείο filepath ώστε το πρόγραμμα να ξέρει σε ποια θέση βρίσκεται η φωτογραφία του χρήστη. Στη συνέχεια καλείται η συνάρτηση insStudent που βρίσκεται στην κλάση Student για να γίνει η δημιουργία του χρήστη, εμφανίζεται το κατάλληλο μήνυμα και η φόρμα κλείνει .

```
private void Savebt_Click(object sender, EventArgs e)
    {
      Student student = new Student();
      if (Savebt.Text == "Save")
      {
        if (saveb())
        {
          string fname = Nametb.Text.Trim();
          string lname = Lnametb.Text.Trim();
          int year = Convert.ToInt32(Yeartb.Text.Trim());
          string fullname = fname + "_" + lname + "_" + year + ".jpg";
          string school = Schtb.Text.Trim();
          string folder = Foldertb.Text.Trim();
          string path = Application.StartupPath;
          if (!Directory.Exists(Path.Combine(path, "Images")))
          {
```

```
                Directory.CreateDirectory(Path.Combine(path, "Images"));
            }
            System.IO.File.Copy(filepath, path + "\\Images\\" + fullname);
            if (student.insStudent(fname, lname, year, school, folder, "\\Images\\" +
fullname))
            {
                MessageBox.Show("Προστέθηκε νεος μαθητης", "Add student",
MessageBoxButtons.OK, MessageBoxIcon.Information);
                this.Close();
            }
            else
            {
                MessageBox.Show("Πρόβλημα δεν προστέθηκε νέος μαθητής", "Add
student", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }
        else
        {
            MessageBox.Show("Παρακαλώ γεμίστε τα κενά πεδία", "Add student",
MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
        }
    }
```

Σε περίπτωση που το κείμενο του κουμπιού είναι Update το πρόγραμμα δημιουργεί μια καινούργια μεταβλητή που είναι το id και είναι τύπου string , παίρνει την τιμή που υπάρχει στο Textbox με όνομα Idtb και την μετατρέπει σε Int32 για να είναι συμβατή. Μετά μέσω της συνάρτησης saveb το πρόγραμμα ελέγχει αν όλα τα πεδία είναι κενά και βγάζει το κατάλληλο μήνυμα. Στη συνέχεια παίρνει όλα τα δεδομένα από τα Textbox και τα βάζει στις κατάλληλες μεταβλητές. Μετά δημιουργεί τη μεταβλητή fullname που συνδυάζει τα πεδία fname, lname και year χρησιμοποιώντας το _ και το επίθεμα .jpg για να αποθηκεύσει το καινούργιο όνομα της φωτογραφίας του χρήστη. Σε περίπτωση που η μεταβλητή imagepath είναι διαφορετική από το μέρος που βρίσκεται η φωτογραφία και το filepath είναι κενό τότε για να μπορέσει να αντιγράψει την καινούργια φωτογραφία το πρόγραμμα σβήνει την παλιά φωτογραφία από το picturebox και μεταφέρει την καινούργια φωτογραφία στο imagepath που αποτελείται από το αρχείο που βρίσκεται μέσα στο φάκελο Images του γονικού καταλόγου. Αν το filepath δεν είναι κενό αντιγράφει σε αυτό τη θέση του αρχείου μαζί με το αρχείο. Έπειτα καλείται η συνάρτηση updateStudents που βρίσκεται στην

κλάση Student με τα κατάλληλα ορίσματα, γίνεται η ενημέρωση του μαθητή και μετά κλείνει η φόρμα.

```
else if (Savebt.Text == "Update")
        {
           int id = Convert.ToInt32(Idtb.Text);
           if (saveb())
           {
              string fname = Nametb.Text.Trim();
              string lname = Lnametb.Text.Trim();
              int year = Convert.ToInt32(Yeartb.Text.Trim());
              string fullname = fname + "_" + lname + "_" + year + ".jpg";
              string school = Schtb.Text.Trim();
              string folder = Foldertb.Text.Trim();
              string path = Application.StartupPath;
              if ((imagepath != path + "\\Images\\" + fullname) && (filepath == null))
              {
                 Picturetb.Image.Dispose();
                 File.Move(imagepath, path + "\\Images\\" + fullname);
              }
              if (filepath != null)
              {
                 System.IO.File.Copy(filepath, path + "\\Images\\" + fullname, true);

              }
              if (student.updateStudents(id, fname, lname, year, school, folder,
"\\Images\\" + fullname))
              {
                 MessageBox.Show("Ο μαθητής ενημερώθηκε!", "Edit student",
MessageBoxButtons.OK, MessageBoxIcon.Information);
                 this.Close();
              }
              else
              {
                 MessageBox.Show("Πρόβλημα! Δεν μπόρεσε να γίνει ενημέρωση
μαθητή!", "Edit student", MessageBoxButtons.OK, MessageBoxIcon.Error);
              }
           }
           else
           {
              MessageBox.Show("Παρακαλώ γεμίστε τα κενά πεδία", "Edit student",
MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
           }
        }
     }
```

3.4.3 Η ΣΥΝΑΡΤΗΣΗ Uploadbt_click

Η συνάρτηση Uploadbt_click εκτελείται όταν ο χρήστης πατήσει το κουμπί ανέβασμα φωτογραφίας. Η συνάρτηση αυτή ανοίγει ένα παράθυρο διαλόγου για να επιλέξουμε μια φωτογραφία η οποία υπακούει στους τύπους αρχείου που έχουμε επιλέξει. Μετά αν το picturebox έχει φωτογραφία την αδειάζει για να μπει η καινούργια φωτογραφία . Μετά το πρόγραμμα φτιάχνει την μεταβλητή img τύπου Image και αποθηκεύει την φωτογραφία. Στη συνέχεια αποθηκεύει το μέρος που βρίσκεται τώρα η φωτογραφία στο filepath και προβάλει την φωτογραφία στο picturebox.

```
private void Uploadbt_Click(object sender, EventArgs e)
    {
      OpenFileDialog dlg = new OpenFileDialog();
      dlg.Filter = "JPG Files(*.jpg)|*.jpg|PNG Files(*.png)|*.png|ALL
      Files(*.*)|*.*";
      if (dlg.ShowDialog() == DialogResult.OK)
      {
        if (Picturetb.Image != null)
        {
          Picturetb.Image.Dispose();
        }
        Image img = Image.FromFile(dlg.FileName);
        filepath = dlg.FileName;
        Picturetb.Image = img;
      }
      dlg.Dispose();
    }
```

3.4.4 Η συνάρτηση Yeartb_KeyPress

Η συνάρτηση Yeartb_KeyPress ελέγχει στο textbox Yeartb όλους τους χαρακτήρες και επιτρέπει μόνο τους αριθμούς και το κουμπί backspace και delete.

```
private void Yeartb_KeyPress(object sender, KeyPressEventArgs e)
    {
      char ch = e.KeyChar;
      if (!Char.IsDigit(ch) && ch != 8 && ch != 46)
      {
        e.Handled = true;
      }
    }
```

3.5 ΚΛΑΣΗ Student

Η κλάση Student είναι η κλάση στην οποία γίνεται η σύνδεση τον στοιχείων της βάσης δεδομένων με τον κώδικα . Για αρχή δημιουργείται η μεταβλητή db που είναι τύπου MY_DB η οποία θα χρησιμοποιηθεί για τη σύνδεση με τη βάση . Η πρώτη συνάρτηση είναι η insStudent η οποία εισάγει τα δεδομένα που πήρε από τις άλλες κλάσεις στην βάση δεδομένων . Στη συνέχεια καλείται η εντολή command που είναι ερώτημα sql και εισάγει τα δεδομένα που πήρε στη βάση στις κατάλληλες θέσεις δηλαδή το fn στο FirstName, ln στο LastName, year στο GraduationYear, sch στο GraduationSchool, fol στο Folder και pic στο Photo. Στη συνέχεια με τη βοήθεια της εντολής command.Parameters.Add γίνεται η σύνδεση τον μεταβλητών της κλάσης με τα στοιχεία της βάσης δεδομένων όπου εκτελείται η σύνδεση με τη βάση δεδομένων και γίνεται έλεγχος για τις σειρές που έχουν αλλαχθεί στη βάση δεδομένων .

```
class Student
  {
    MY_DB db = new MY_DB();

    public bool insStudent(String fname, String lname, int year, String school, String
folder, String photo)
    {
      SqlCommand command = new SqlCommand("INSERT INTO
Students(FirstName,LastName,GraduationYear,GraduationSchool,Folder,Photo)
VALUES (@fn,@ln,@year,@sch,@fol,@pic)", db.GetConnection);
      command.Parameters.Add("@fn", System.Data.SqlDbType.NVarChar).Value
= fname;
      command.Parameters.Add("@ln", System.Data.SqlDbType.NVarChar).Value
= lname;
      command.Parameters.Add("@year", System.Data.SqlDbType.Int).Value =
year;
      command.Parameters.Add("@sch",
System.Data.SqlDbType.NVarChar).Value = school;
      command.Parameters.Add("@fol",
System.Data.SqlDbType.NVarChar).Value = folder;
      command.Parameters.Add("@pic",
System.Data.SqlDbType.NVarChar).Value = photo;

      db.openConnection();
      if (command.ExecuteNonQuery() == 1)
      {
        db.closeConnection();
        return true;
      }
```

```
        else
        {
            db.closeConnection();
            return false;
        }
    }
```

3.5.1 Η συνάρτηση getStudents

Η συνάρτηση getStudents είναι τύπου Datatable και στην ουσία επιστρέφει ένα πίνακα με εγγραφές ανάλογα με το sql ερώτημα που του έχουμε δώσει . Η κλάση αυτή είναι απαραίτητη για την αναζήτηση που γίνεται στην αρχική φόρμα και την ανανέωση.

```
        public DataTable getStudents(SqlCommand command)
    {
        command.Connection = db.GetConnection;
        SqlDataAdapter sqlDataAdapter = new SqlDataAdapter(command);
        DataTable table = new DataTable();
        sqlDataAdapter.Fill(table);
        return table;
    }
```

3.5.2 Η συνάρτηση updateStudent

Η συνάρτηση updateStudent αλλάζει τα δεδομένα που πήρε από τις άλλες κλάσεις στη βάση δεδομένων. Για αρχή καλείται η εντολή command που είναι ερώτημα sql και εισάγει τα δεδομένα που πήρε στις κατάλληλες θέσεις της βάσης δεδομένων. Στη συνέχεια με τη βοήθεια της εντολής command.Parameters.Add γίνεται η σύνδεση τον μεταβλητών της κλάσης με τα στοιχεία της βάσης δεδομένων και τέλος εκτελείται η σύνδεση με τη βάση δεδομένων και γίνεται έλεγχος για τις σειρές που έχουν αλλαχθεί.

```
public bool updateStudents(int id,String fname, String lname, int year, String school,
String folder, String photo)
    {
        SqlCommand command = new SqlCommand("UPDATE Students SET
FirstName=@fn , LastName=@ln , GraduationYear=@year ,
GraduationSchool=@sch , Folder=@fol , Photo=@pic WHERE id=@Id",
db.GetConnection);
```

```
        command.Parameters.Add("@id", System.Data.SqlDbType.Int).Value = id;
        command.Parameters.Add("@fn", System.Data.SqlDbType.NVarChar).Value
= fname;
        command.Parameters.Add("@ln", System.Data.SqlDbType.NVarChar).Value
= lname;
        command.Parameters.Add("@year", System.Data.SqlDbType.Int).Value =
year;
        command.Parameters.Add("@sch",
System.Data.SqlDbType.NVarChar).Value = school;
        command.Parameters.Add("@fol",
System.Data.SqlDbType.NVarChar).Value = folder;
        command.Parameters.Add("@pic",
System.Data.SqlDbType.NVarChar).Value = photo;

        db.openConnection();
        if (command.ExecuteNonQuery() == 1)
        {
           db.closeConnection();
           return true;
        }
        else
        {
           db.closeConnection();
           return false;
        }
    }
```

3.5.3 Η συνάρτηση deleteStudent

Η συνάρτηση deleteStudent καλείται όταν χρειάζεται η διαγραφή ή εγγραφή ενός χρήστη. Μέσα από την μεταβλητή command που είναι τύπου sql βρίσκεται ο χρήστης που είναι προς διαγραφή μέσω του ερωτήματος sql.

```
public bool deleteStudent(int id)
    {
        SqlCommand command = new SqlCommand("DELETE FROM Students
WHERE id=@STDID", db.GetConnection);
        command.Parameters.Add("@STDID", System.Data.SqlDbType.Int).Value =
id;

        db.openConnection();
        if (command.ExecuteNonQuery() == 1)
        {
           db.closeConnection();
```

```
            return true;
        }
        else
        {
            db.closeConnection();
            return false;
        }
    }
```

3.6 ΚΛΑΣΗ Form1

Όπως έχουμε ήδη αναφέρει η κλάση Form1 είναι η βασική κλάση του προγράμματος στην οποία γίνεται η αναζήτηση, η εμφάνιση, η διαγραφή και η εκτέλεση εκτύπωσης. Με την συνάρτηση Form1_Load καλείται η συνάρτηση filldatagrid που γεμίζει τον πίνακα της εφαρμογής με εγγραφές. Στη συνέχεια η συνάρτηση Create_Click σε περίπτωση που ο χρήστης πατήσει το κουμπί Create δημιουργεί έναν νέο μαθητή και ανοίγει η φόρμα Students για την εισαγωγή των δεδομένων. Μόλις κλείσει η φόρμα Students γίνεται ανανέωση των εγγραφών του πίνακα από την συνάρτηση filldatagrid.

```
public partial class Form1 : Form
  {
    public Form1()
    {
        InitializeComponent();
    }
    private void Form1_Load(object sender, EventArgs e)
    {
        filldatagrid();
    }
    private void Create_Click(object sender, EventArgs e)
    {
        Students std = new Students();
        std.ShowDialog();
        filldatagrid
    }
```

3.6.1 Η συνάρτηση dataGridView1_CellContentClick

Η συνάρτηση dataGridView1_CellContentClick χωρίζεται σε δύο μέρη, στην διαγραφή δεδομένων και στην εκτύπωση της φωτογραφίας της εγγραφής. Στην αρχή δημιουργείται μια μεταβλητή student τύπου Student που χρησιμοποιείται για να καλέσει συναρτήσεις από την κλάση Student. Μετά ελέγχει αν έχουμε πατήσει το κουμπί Delete ή το κουμπί Print στη σειρά που έχουμε επιλέξει. Αν έχουμε πατήσει το Delete τότε ξεκινάει το πρώτο μέρος της συνάρτησης και στην μεταβλητή id που είναι τύπου int μπαίνει ο αριθμός της σειράς στην οποία βρισκόμαστε . Μετά με ανάλογο μήνυμα ρωτάμε τον χρήστη αν θέλει να διαγράψει την εγγραφή. Αν ο χρήστης πατήσει ναι τότε καλείται από την κλάση Students η συνάρτηση deleteStudent και διαγράφει την εγγραφή. Στη συνέχεια διαγράφετε η φωτογραφία και εμφανίζεται το κατάλληλο μήνυμα ότι ο μαθητής διαγράφηκε. Σε περίπτωση που έχουμε πατήσει print τρέχει το δεύτερο μέρος της συνάρτησης και τρέχει η εντολή διαλόγου PrintDialog. Εκεί πέρα μας γίνεται η απεικόνιση της φωτογραφίας που θα στείλουμε για εκτύπωση η θα μετατρέψουμε σε pdf μέσα από τα εργαλεία της PrintDialog. Στη συνέχεια θα γίνει ανανέωση των εγγραφών του πίνακα μέσω της συνάρτησης filldatagrid.

```
private void dataGridView1_CellContentClick(object sender,
DataGridViewCellEventArgs e)
      {
         Student student = new Student();
         if (dataGridView1.Columns[e.ColumnIndex].Name == "Delete")
         {
            int id =
Convert.ToInt32(dataGridView1.CurrentRow.Cells[2].Value.ToString());
            if (MessageBox.Show("Θέλετε να διαγραφεί ο μαθητής?", "Delete
Student", MessageBoxButtons.YesNo, MessageBoxIcon.Question) ==
DialogResult.Yes)
            {
               if (student.deleteStudent(id))
               {
                  string path = Application.StartupPath;
                  File.Delete(Path.Combine(path +
dataGridView1.CurrentRow.Cells[8].Value.ToString()));
                  MessageBox.Show("Ο μαθητής διαγράφηκε!", "Delete Student",
MessageBoxButtons.OK, MessageBoxIcon.Information);
               }
               else
               {
```

```
                MessageBox.Show("Πρόβλημα δεν μπόρεσε να διαγραφεί ο
μαθητής!", "Delete Student", MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
            }
        }
        else if (dataGridView1.Columns[e.ColumnIndex].Name == "Print")
        {
            PrintDialog printdlg = new PrintDialog();
            if (printdlg.ShowDialog() == DialogResult.OK)
            {
                PrintDocument printDocument1 = new PrintDocument();
                PrintPreviewDialog printPrvDlg = new PrintPreviewDialog();
                printDocument1.PrintPage += new
PrintPageEventHandler(this.printDocument1_PrintPage);
                printPrvDlg.Document = printDocument1;
                printPrvDlg.ShowDialog();
            }
        }
        filldatagrid();
    }
```

3.6.2 Η συνάρτηση printDocument1_PrintPage

Η συνάρτηση printDocument1_PrintPage βρίσκει την εικόνα μέσα από τον πίνακα από το κελί 8 τις γραμμής που έχουμε επιλέξει, μετατρέπει την εικόνα σε Bitmap μέσω της εντολής Bitmap, ετοιμάζει την φωτογραφία για εκτύπωση ή μετατροπή σε pdf και την στέλνει στην προεπισκόπηση του PrintDialog. Τέλος μέσα από την εντολή myBitmap1.Dispose ελευθερώνουμε την μνήμη που έπιανε η φωτογραφία.

```
private void printDocument1_PrintPage(object sender, PrintPageEventArgs e)
    {
        string path = Application.StartupPath;
string photo = path + dataGridView1.CurrentRow.Cells[8].Value.ToString();
        Bitmap myBitmap1 = new Bitmap(photo);
        e.Graphics.DrawImage(myBitmap1, e.PageBounds);
        myBitmap1.Dispose();
    }
```

3.6.3 Η συνάρτηση dataGridView1_DoubleClick

Η συνάρτηση dataGridView1_DoubleClick εκτελείται όταν ο χρήστης πατήσει διπλό κλικ πάνω σε κάποια εγγραφή. Στη συνέχεια δημιουργείται η μεταβλητή std τύπου Students που χρησιμοποιείται για να μπορέσουμε να ανοίξουμε μία καινούργια φόρμα Students και να περάσουμε μέσα τα δεδομένα της εγγραφής. Αυτό γίνεται με την αντιγραφή των δεδομένων κάθε κελιού της εγγραφής μετά από τη μετατροπή τους σε String και τα δεδομένα εμφανίζονται στα κατάλληλα Textbox της φόρμας Student. Στη συνέχεια αντιγράφεται η διαδρομή της φωτογραφίας και μέσω της εντολής Image.FromFile εμφανίζουμε την εικόνα στο picturebox της δεύτερης φόρμας. Τέλος αλλάζουμε το όνομα του κουμπιού αποθήκευσης σε Update ώστε ο χρήστης να ξέρει ότι θα γίνει ενημέρωση των στοιχείων όταν το πατήσει. Μόλις κλείσει η φόρμα του Students γίνεται ανανέωση των στοιχείων του πίνακα και διαγράφεται η προσωρινή εικόνα που είχε δημιουργηθεί για να εμφανιστεί στη φόρμα Students ώστε να ελευθερωθεί μνήμη.

```
private void dataGridView1_DoubleClick(object sender, EventArgs e)
 {
    Students std = new Students();
    std.Idtb.Text = dataGridView1.CurrentRow.Cells[2].Value.ToString().Trim();
    std.Nametb.Text =
    dataGridView1.CurrentRow.Cells[3].Value.ToString().Trim();
    std.Lnametb.Text =
    dataGridView1.CurrentRow.Cells[4].Value.ToString().Trim();
    std.Yeartb.Text =
    dataGridView1.CurrentRow.Cells[5].Value.ToString().Trim();
    std.Schtb.Text =
    dataGridView1.CurrentRow.Cells[6].Value.ToString().Trim();
    std.Foldertb.Text =
    dataGridView1.CurrentRow.Cells[7].Value.ToString().Trim();
    string path = Application.StartupPath;
    std.Picturetb.Image =
    Image.FromFile(path+dataGridView1.CurrentRow.Cells[8].Value.ToString())
    ;
    std.imagepath = path +
    dataGridView1.CurrentRow.Cells[8].Value.ToString();
    std.Savebt.Text = "Update";
    std.ShowDialog();
    filldatagrid();
      if (std.Picturetb.Image != null)
```

```
            {
                std.Picturetb.Image.Dispose();
            }
    }
```

3.6.4 Η συνάρτηση Textbox1_TextChanged

Η συνάρτηση Textbox1_TextChanged είναι η αναζήτηση εγγραφών της φόρμας. Αρχικά ορίζεται μια μεταβλητή τύπου sql και στη συνέχεια ένα πίνακας μεταβλητών τύπου String που αποτελείται από "FirstName", "LastName", "GraduationYear", "GraduationSchool" και "Folder". Στην αρχή η συνάρτηση ελέγχει αν το textbox της αναζήτησης είναι κενό και κάνει ανανέωση του πίνακα. Στη συνέχεια μέσω ενός ερωτήματος Sql ψάχνει αν οι λέξεις ή τα γράμματα που γράψαμε, που θα πρέπει να είναι πάνω από 1 υπάρχουν μέσα στον πίνακα. Εάν υπάρχουν εμφανίζει τις εγγραφές και στη συνέχεια ελέγχει αν μετα από το κενό υπάρχει άλλο γράμμα ή λέξη. Η αναζήτηση σταματάει όταν έχει βρει την εγγραφή που έχει παρόμοιο συνδυασμό γραμμάτων ή είναι οι ίδιες λέξεις.

```
private void textBox1_TextChanged(object sender, EventArgs e)
    {
      SqlCommand command = new SqlCommand();
      string sql;
      string[] collumnname = { "FirstName", "LastName", "GraduationYear",
"GraduationSchool", "Folder" };
      try
      {
        if (string.IsNullOrEmpty(textBox1.Text.Trim()))
        {
          filldatagrid();
          return;
        }
        sql = "SELECT * FROM Students WHERE ";
        string[] words = textBox1.Text.Trim().Split(' ');
        if (words.Length >= 1)
        {
          foreach (string word in words)
          {
            sql += "(";
            foreach (string column in collumnname) sql += column + " LIKE N'%"
+ word + "%' OR ";
```

```
                sql = sql.Substring(0, sql.Length - 3);
                sql += ") AND ";
            }

            sql = sql.Substring(0, sql.Length - 5);
            sql += " ORDER BY Id ASC";
            command.CommandText = sql;
            command.Parameters.Clear();
            filldatagrid(command);
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show("ERROR" + ex.Message.ToString());
    }
    finally
    {
        textBox1.Focus();
    }
}
```

3.6.5 Η συνάρτηση filldatagrid

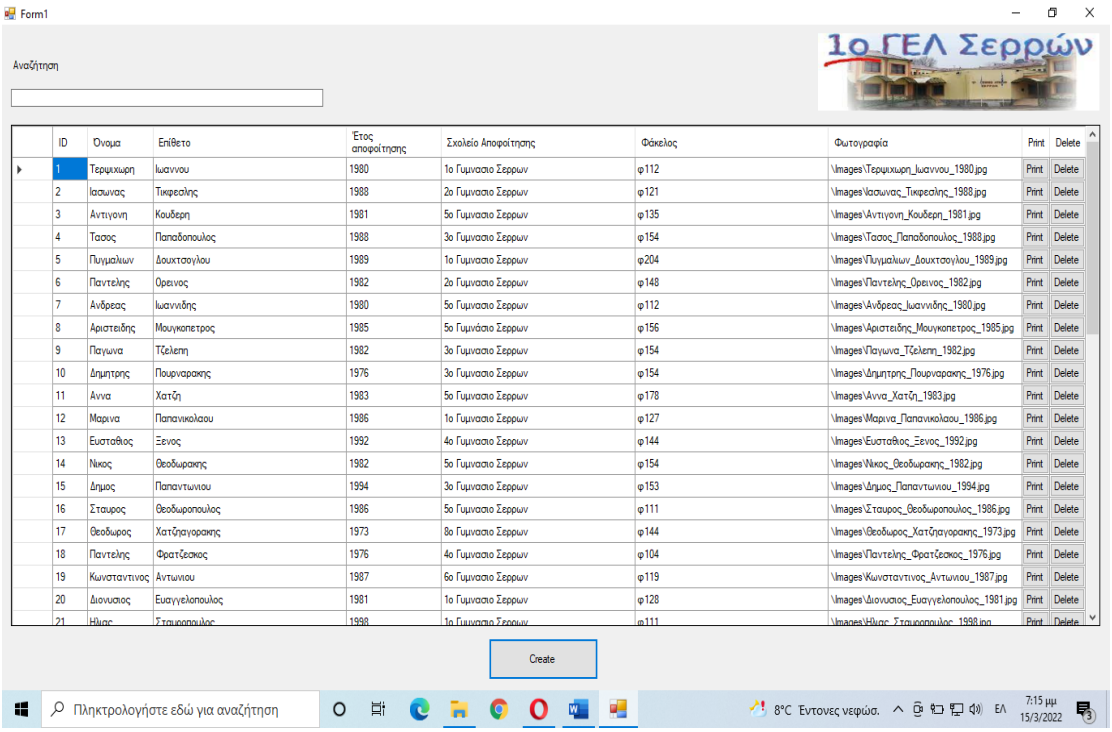
Η συνάρτηση filldatagrid δημιουργεί μια μεταβλητή με όνομα student και τύπου Student. Μέσα στη μεταβλητή sql τύπου string αποθηκεύεται ένα ερώτημα αναζήτησης όλων των εγγραφών σε μορφή Sql. Μετα ελέγχει αν υπάρχει ήδη sql ερώτημα προς εκτέλεση για τη συγκεκριμένη συνάρτηση ή αν είναι κενό. Σε περίπτωση που είναι κενό αντιγράφει το ερώτημα στην μεταβλητή command που είναι μορφής sql και το στέλνει στη συνάρτηση getStudents που βρίσκεται στην κλάση Student. Έπειτα γίνεται ενημέρωση των εγγραφών του πίνακα με βάση την εντολή sql.

```
public void filldatagrid(SqlCommand cmd = null)
    {
        Student student = new Student();
```

```
        SqlCommand command = new SqlCommand();

        string sql = "SELECT * FROM Students ORDER BY Id ASC";

        if (cmd == null)
        {
            command.CommandText = sql;

        }
        else
        {
            command = cmd;
        }

        dataGridView1.DataSource = student.getStudents(command);
    }
```

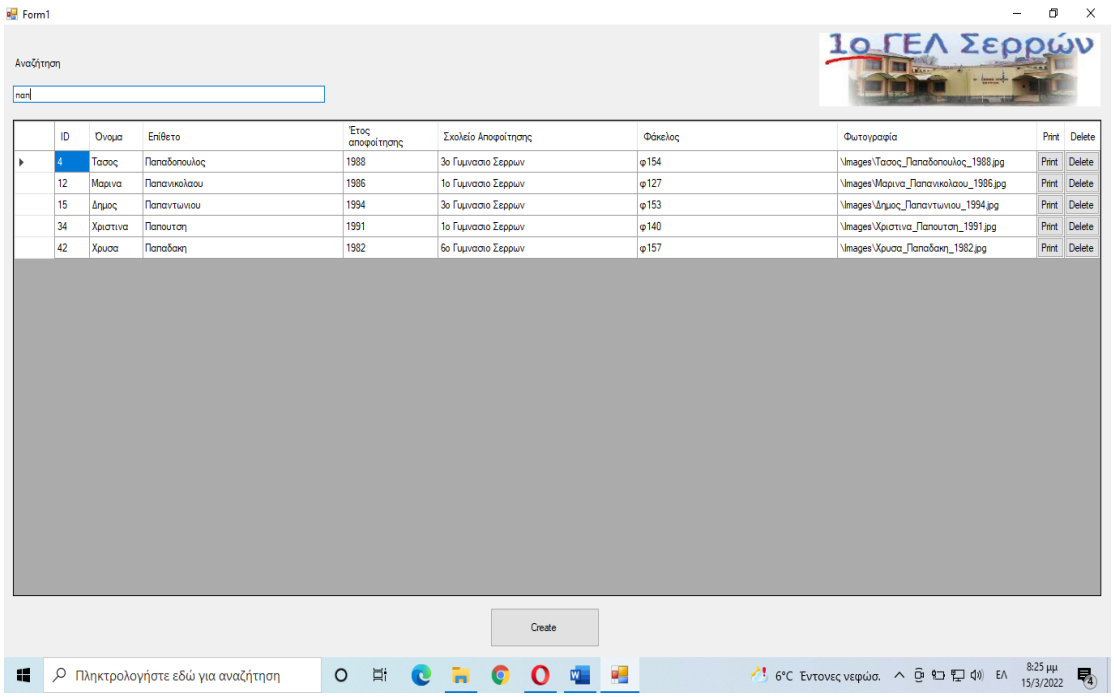
4. ΕΓΧΕΙΡΙΔΙΟ ΛΕΙΤΟΥΡΓΙΑΣ

Η πρώτη εικόνα που θα συναντήσει ο χρήστης ανοίγοντας την εφαρμογή είναι η λίστα των εγγραφών και του δίνεται η δυνατότητα να δημιουργήσει μία καινούρια εγγραφή ή να αναζητήσει την εγγραφή που χρειάζεται.



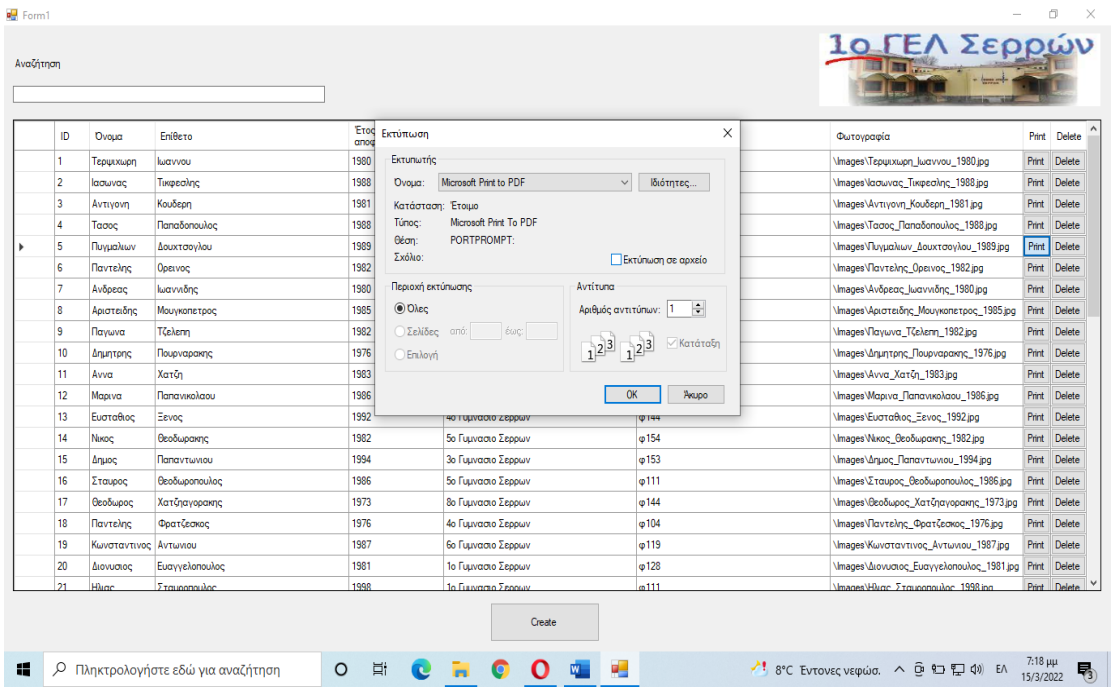
***Εικόνα 13*:** Πρώτη επαφή του χρήστη με την εφαρμογή

Εάν ο χρήστης θέλει να βρει κάποιο από τα υπάρχοντα απολυτήρια θα πρέπει να πληκτρολογήσει στην μπάρα αναζήτησης οποιοδήποτε από τα στοιχεία του μαθητή διαθέτει και όσο πληκτρολογεί θα εμφανίζονται στην οθόνη μόνο οι εγγραφές που ταιριάζουν ώστε να βρει εύκολα την εγγραφή που ψάχνει. Αφού επιλέξει την κατάλληλη εγγραφή μπορεί να την επεξεργαστεί, να τη διαγράψει ή να την εκτυπώσει.



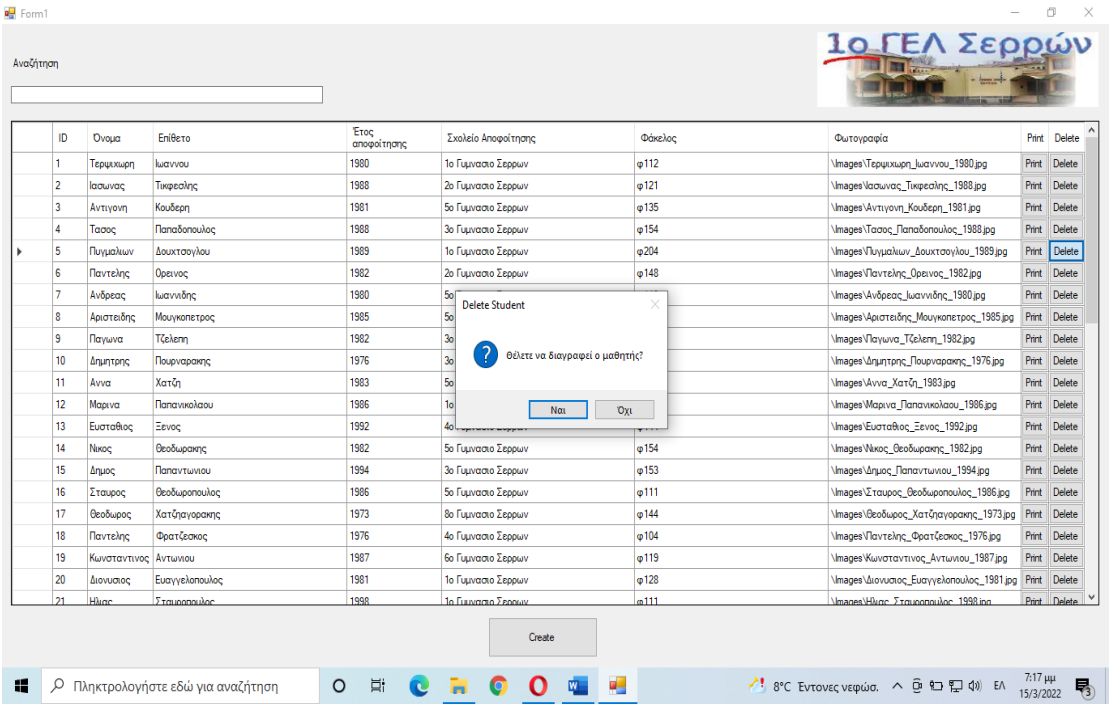
***Εικόνα 14*: Εμφάνιση των εγγραφών που ταιριάζουν με την αναζήτηση**

Αφού ο χρήστης βρει την εγγραφή που ψάχνει θα πρέπει να πατήσει το πλήκτρο Print και στη συνέχεια να επιλέξει εαν θέλει να την αποθηκεύσει σε ψηφιακή μορφή ή να την εκτυπώσει.



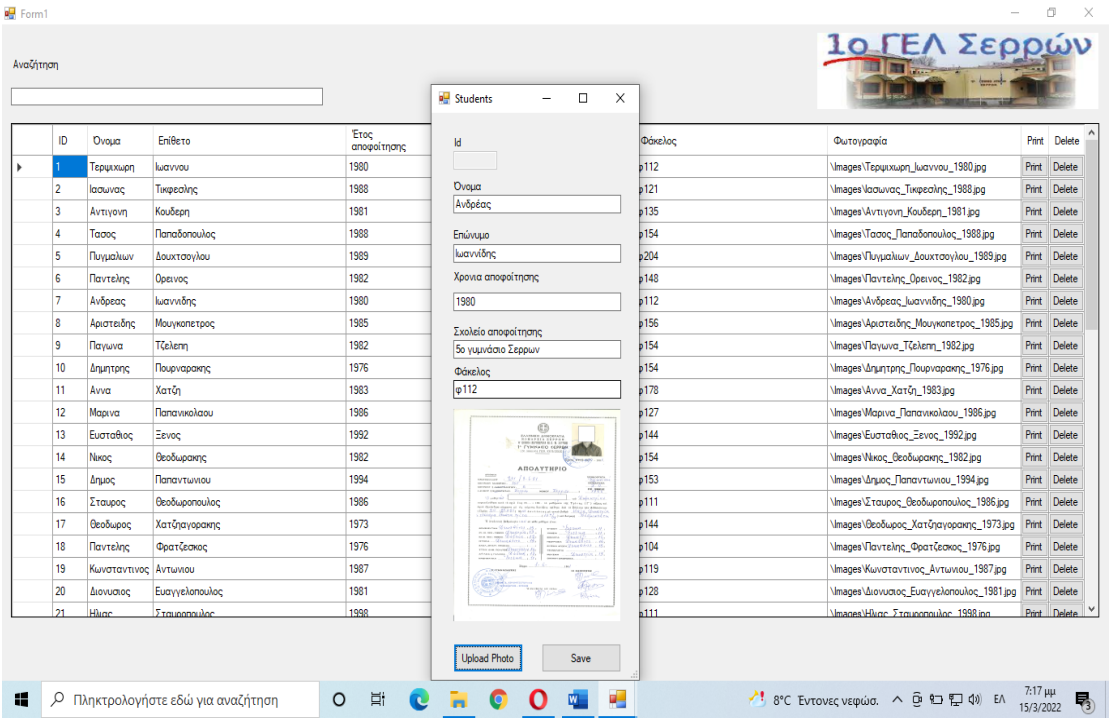
***Εικόνα 15*: Επιλογή εκτύπωσης ή αποθήκευσης της εικόνας σε μορφή PDF**

Για να διαγράψει την εγγραφή θα πρέπει να πατήσει το κουμπί delete και στη συνέχεια να πατήσει <u>ναι</u> στο παράθυρο με το αντίστοιχο μήνυμα.



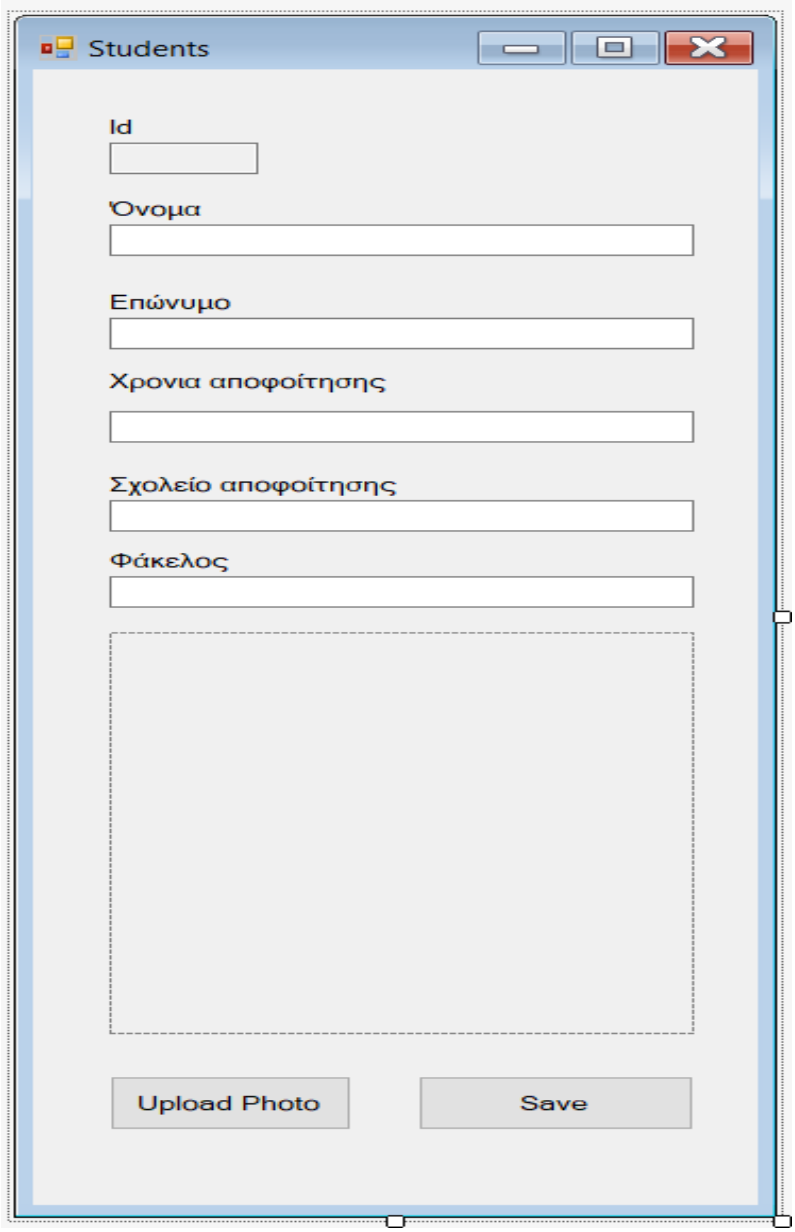
***Εικόνα 16*:** Μενού διαγραφής εγγραφών

Για να επεξεργαστεί την εγγραφή που επιθυμεί αρκεί να πατήσει διπλό κλικ πάνω της και θα ανοίξει το μενού επεξεργασίας από όπου μπορεί να αλλάξει οποιοδήποτε στοιχείο της εγγραφής.



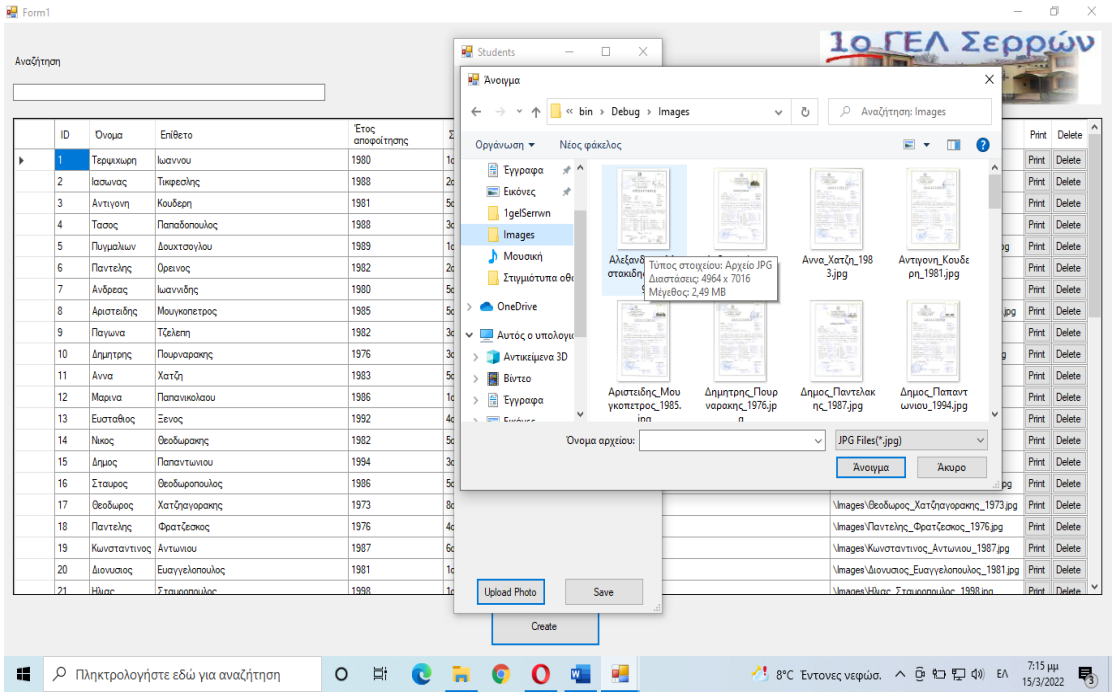
***Εικόνα 17*:** Μενού επεξεργασίας εγγραφών

Εάν ο χρήστης επιλέξει να δημιουργήσει μια καινούρια εγγραφή τότε ανοίγει η φόρμα students και ο χρήστης καλείται να συμπληρώσει το ονοματεπώνυμο, την χρονιά και το σχολείο αποφοίτησης καθώς και τον φάκελο στον οποίο βρίσκεται το εν λόγω απολυτήριο.



***Εικόνα 18*:** Η φόρμα Students όπου εισάγουμε τα στοιχεία του μαθητή

Στη συνέχεια αφού πατήσει το κουμπί upload photo ανοίγει το μενού των εικόνων και ο χρήστης πρέπει να επιλέξει την κατάλληλη σάρωση.

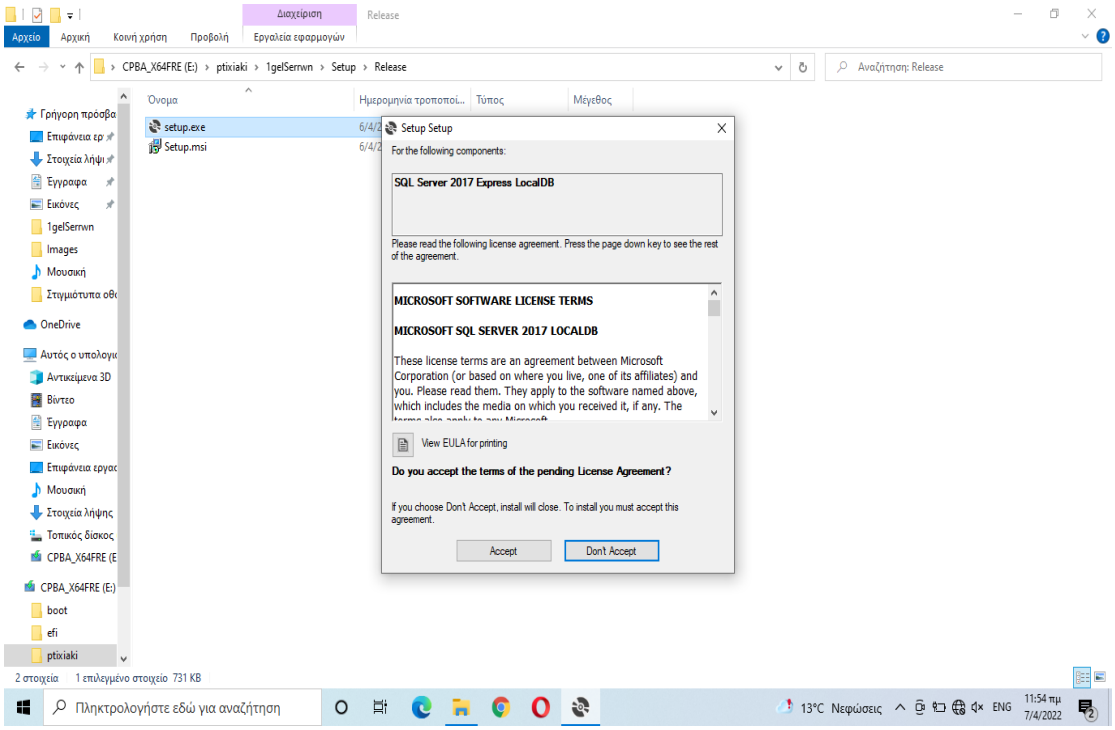


***Εικόνα 19*:** Επιλογή της εικόνας του απολυτηρίου

Τέλος πατώντας το κουμπί save η νέα εγγραφή αποθηκεύεται στη βάση και η εικόνα αποθηκεύεται στον φάκελο images με όνομα εικόνας το ονοματεπώνυμο του μαθητή και την χρονολογία αποφοίτησης ώστε να μπορεί το προσωπικό του σχολείου να βρει εύκολα την εγγραφή ακόμα και σε περίπτωση που το πρόγραμμα δεν ανταποκρίνεται.

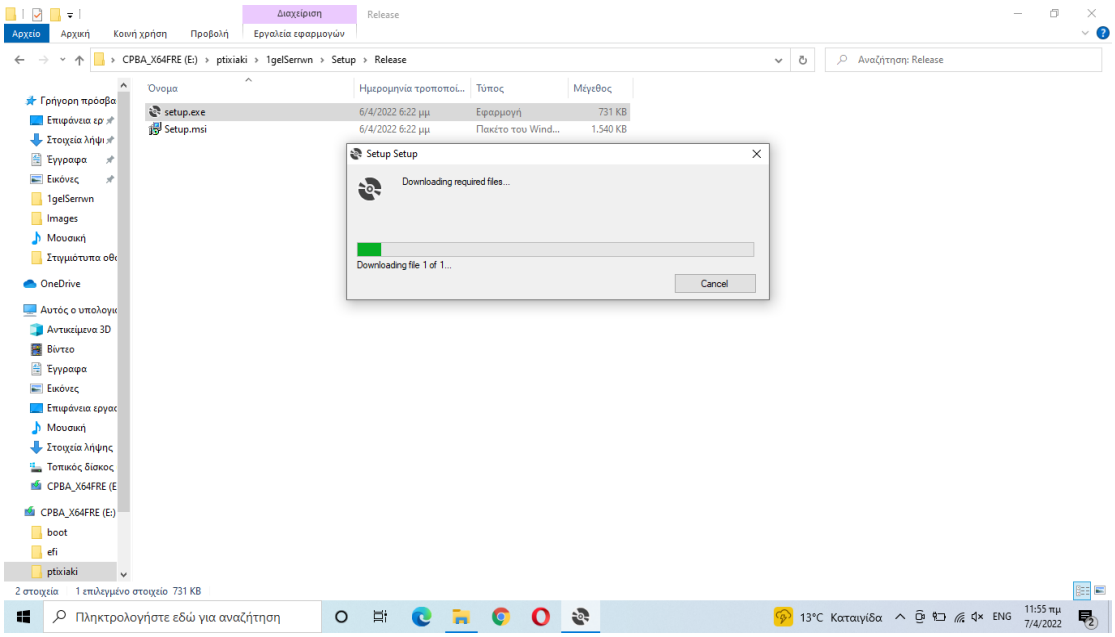
5. ΕΓΚΑΤΑΣΤΑΣΗ ΕΦΑΡΜΟΓΗΣ

Για να μπορέσει να εγκατασταθεί η εφαρμογή μας πρέπει στον υπολογιστή να υπάρχει το .NET FRAMEWORK version 4.7.2 και το SQL Server 2017 Express LocalDB. Για να αρχίσει η εγκατάσταση τρέχουμε το setup.exe και ανοίγει το παράθυρο εγκατάστασης πρόσθετων εφαρμογών.



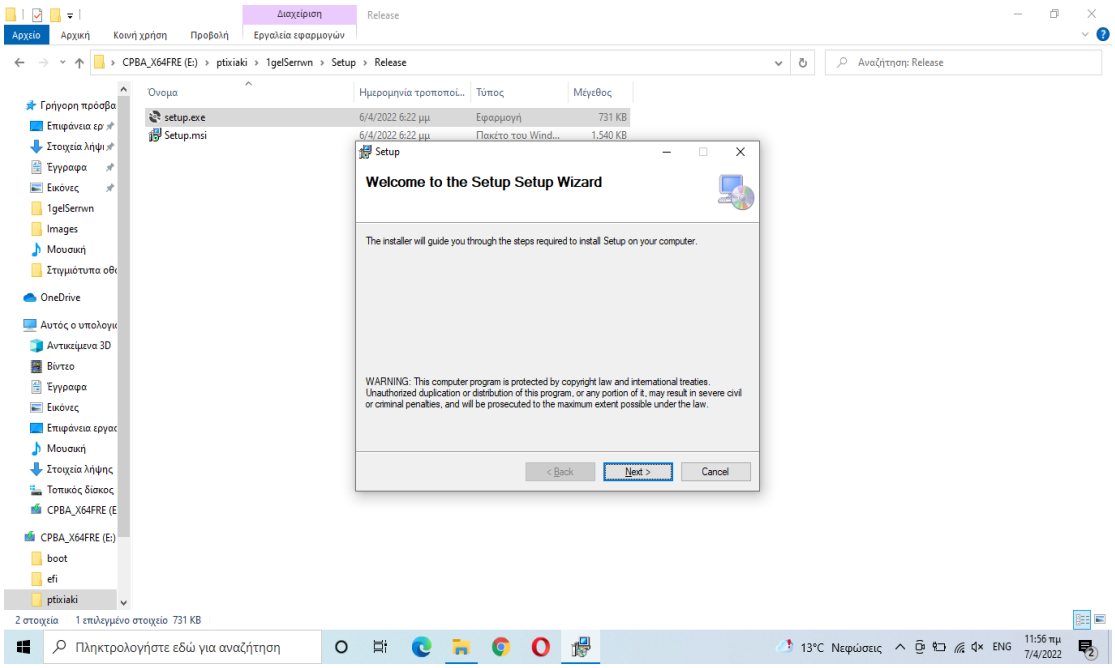
Εικόνα 20: Παράθυρο εγκατάστασης πρόσθετων εφαρμογών

Αφού πατήσουμε Accept οι προαναφερθείσες εφαρμογές αρχίζουν να κατεβαίνουν (*εικόνα 21*).



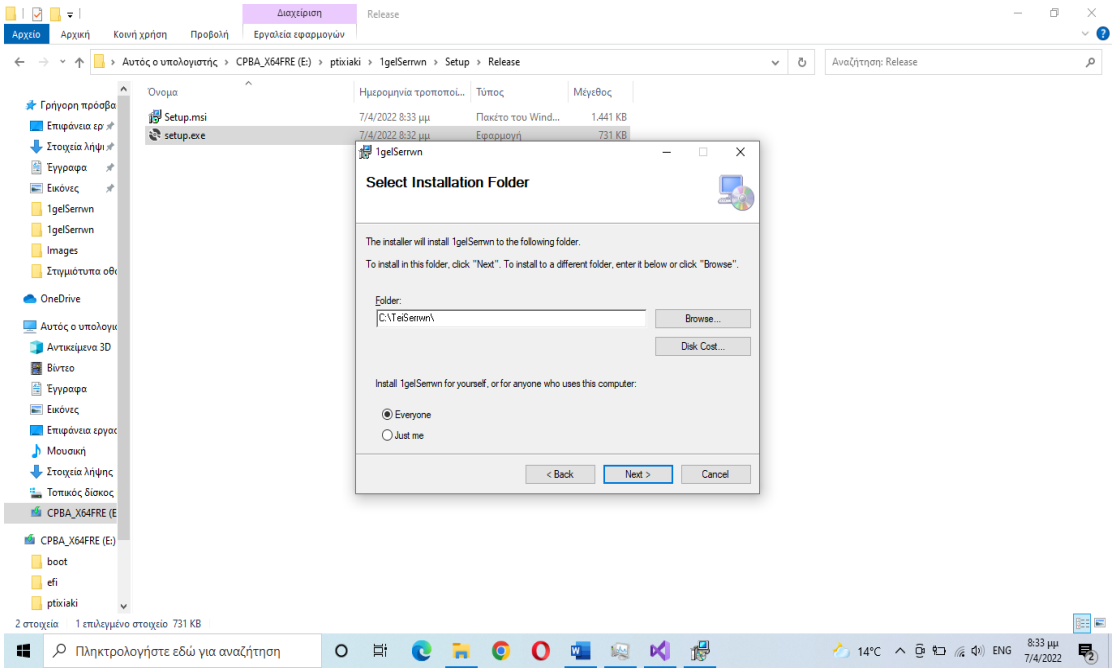
Εικόνα 21: κατέβασμα εφαρμογών

Εφόσον οι εφαρμογές κατέβουν με επιτυχία ανοίγει ο setup wizard



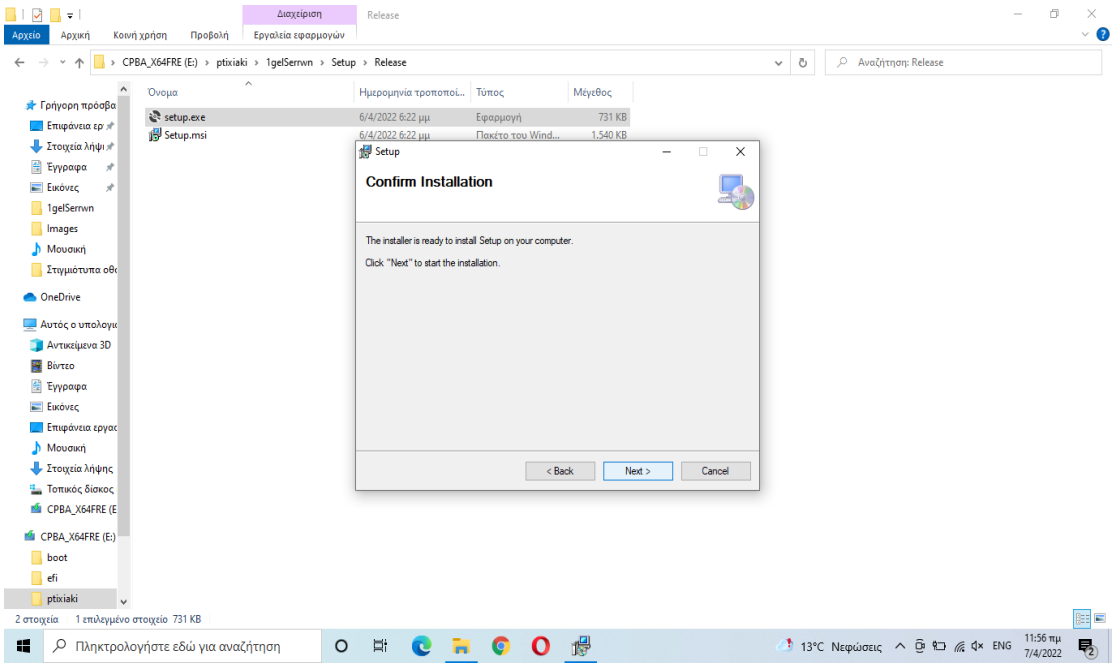
Εικόνα 22: setup wizard

Αφού πατήσουμε next ανοίγει νέο παράθυρο όπου επιλέγουμε το χώρο στον οποίο θα γίνει η εγκατάσταση και ιδανικά την αποθηκεύουμε στον δίσκο C. Μετά επιλέγουμε αν στην εφαρμογή θα έχουν πρόσβαση όλοι οι χρήστες του υπολογιστή ή μόνο ο χρήστης που κάνει την εγκατάσταση.



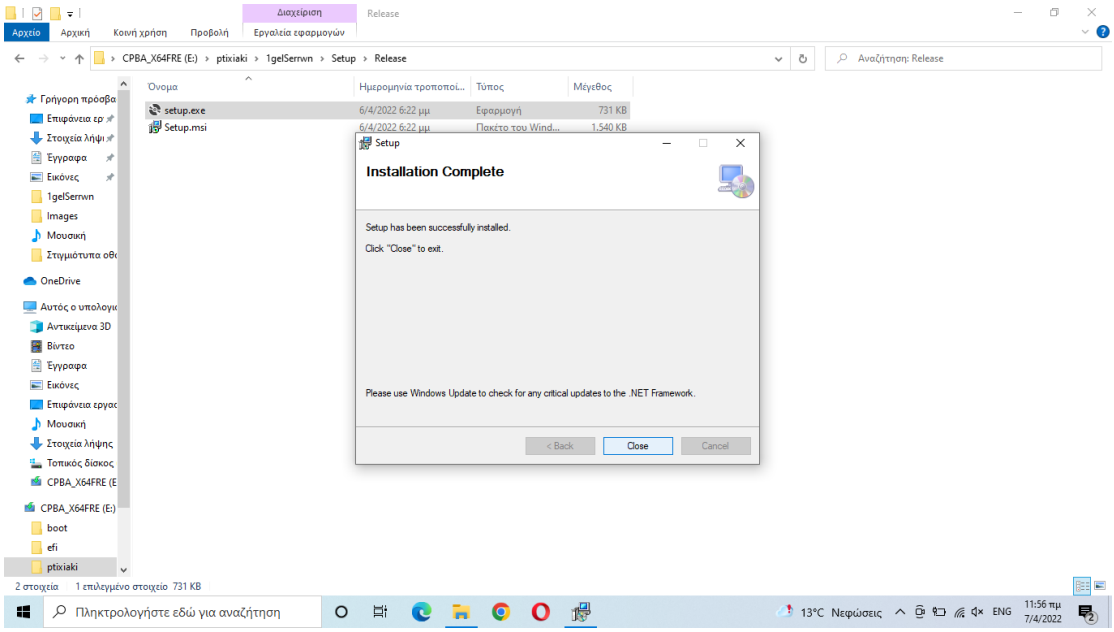
Εικόνα 23: επιλογή φακέλου εγκατάστασης

Αφού ο χρήστης πατήσει next καλείται να επιβεβαιώσει τις επιλογές του



Εικόνα 24: επιβεβαίωση επιλογών εγκατάστασης

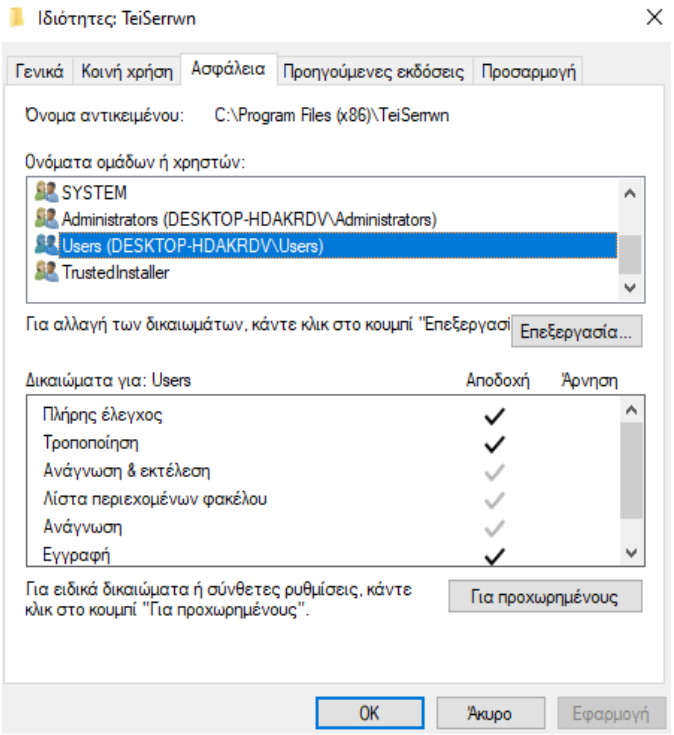
Τέλος, αφού ολοκληρωθεί η εγκατάσταση πατάμε το κουμπί close.



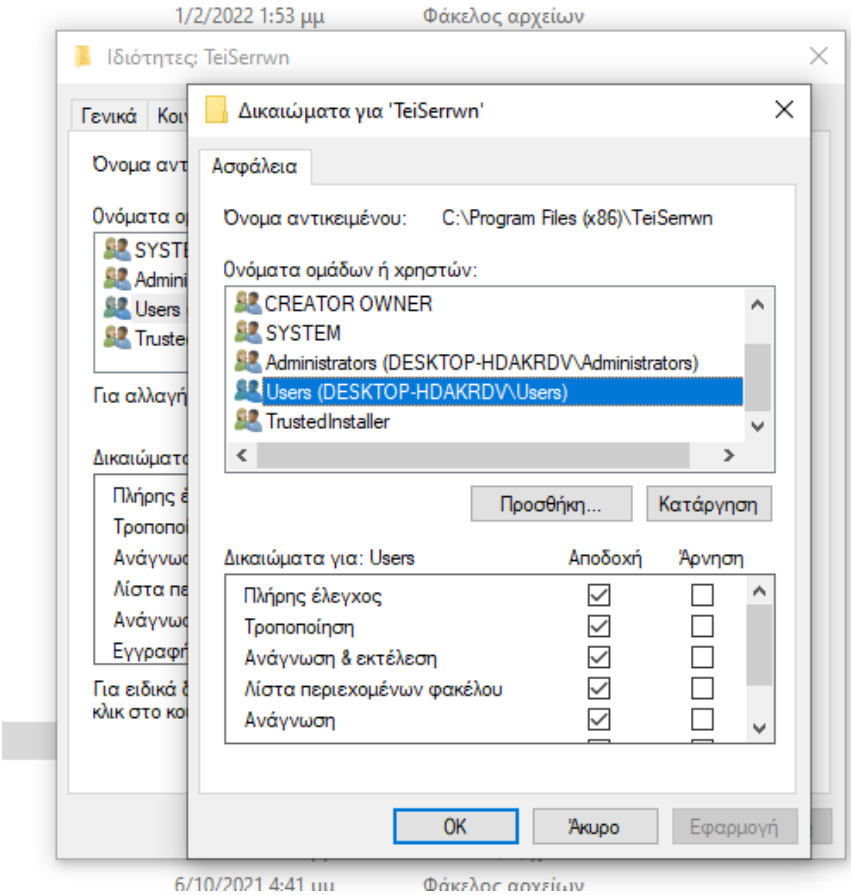
Εικόνα 25: ολοκλήρωση εγκατάστασης

ΠΡΟΣΟΧΗ

Στην περίπτωση που ο χρήστης θέλει να αποθηκεύσει την εφαρμογή στον φάκελο αρχεία εφαρμογών (x86) ή αρχεία εφαρμογών, θα πρέπει μετά να πάμε στον φάκελο της εφαρμογής να πατήσουμε δεξί κλικ/ιδιότητες/ασφάλεια/επεξεργασία να επιλέξουμε τον χρήστη users και να του δώσουμε πλήρη έλεγχο όπως φαίνεται στις εικόνες 26 και 27.



Εικόνα 26: δικαιώματα εφαρμογής



Εικόνα 27: αλλαγή δικαιωμάτων εφαρμογής

ΠΑΡΟΜΟΙΕΣ ΕΦΑΡΜΟΓΕΣ

Directory opus

Εικονα 28: λογοτυπο του directory opus

Το Directory Opus είναι ένα πρόγραμμα διαχείρισης αρχείων που γράφτηκε για το υπολογιστικό σύστημα Amiga στις αρχές της δεκαετίας του 1990 ενώ η εμπορική έκδοση για το Amiga σταμάτησε στα τέλη της ίδιας δεκαετίας. Η ανάπτυξη και η πώληση του DOpus συνεχίζονται μέχρι και σήμερα από την GPSoftware για το λειτουργικό σύστημα Microsoft Windows ενώ από το 2000 κυκλοφορεί και μία έκδοση ανοιχτού κώδικα του Directory Opus 4. Το Directory Opus δημιουργήθηκε και αναπτύσετε μέχρι και σήμερα από τον Jonathan Potter για την θυγατρική εταιρεία της Amiga Inovatronics, ενώ από το 1994 και μετά την εμπορική έκδοση του προγράμματος ανέλαβε η GPSoftware.

Total Commander

Εικόνα 29: λογότυπο του Total Commander

Το Total Commander είναι ένα πρόγραμμα διαχείρισης αρχείων κοινής χρήσης για Windows, Windows Phone, Windows Mobile/Windows CE και Android, που αναπτύχθηκε από τον Christian Ghisler. Αρχικά είχε γραφτεί με χρήση της γλώσσας προγραμματισμου Delphi, αλλά οι μεταγενέστερες εκδόσεις για Windows 64-bit αναπτύχθηκαν με τη βοήθεια του ολοκληρωμένου περιβάλλοντος ανάπτυξης Lazarus. Περιέχει έναν ενσωματωμένο FTP client, πολλαπλές διεπαφές, σύγκριση αρχείων και εργαλείο ταυτόχρονης μετονομασίας πολλαπλών αρχείων ενώ είναι συμβατό και με Linux με τη χρήση Wine.

XYplorer

Εικόνα 30: λογότυπο του XYplorer

Ο XYplorer είναι ένας διαχειριστής αρχείων για windows με συμβατότητα από τα windows xp μέχρι και τα 11. Ο XYplorer είναι ένας υβριδικός διαχειριστής αρχείων που συνδυάζει χαρακτηριστικά που βρίσκονται σε διαχειριστές πλοήγησης και ορθόδοξους διαχειριστές αρχείων. Εκτός από τα παράθυρα διπλών φακέλων διαθέτει και μία διεπαφή με καρτέλες που υποστηρίζει μεταφορά αρχείων μεταξύ καρτελών και παραθύρων.

xplorer²

Εικόνα 31: λογότυπο του xplorer²

Ο xplorer² είναι ένας διαχειριστής αρχείων διπλού παραθύρου για λειτουργικά συστήματα Microsoft Windows που αναπτύχθηκε από τον Νίκο Βοζίνη. Προσφέρει τη λειτουργικότητα των ορθοδόξων διαχειριστών αρχείων μέσω μιας διεπαφής παρόμοιας με τη γνωστή Εξερεύνηση των Windows. Ορισμένες από τις δυνατότητες του είναι η διαχείριση αρχείων με καρτέλες σε κάθε παράθυρο, η προβολή και η επεξεργασία αρχείων κειμένου, η αναζήτηση αρχείων με χρήση αυθαίρετων κριτηρίων, η δυνατότητα σύγκρισης και συγχρονισμού φακέλων και η δυνατότητα εκτέλεσης σε σειρά ή ταυτόχρονων λειτουργιών αντιγραφής και μετακίνησης με διαχείριση σφαλμάτων. Οι νεότερες εκδόσεις υποστηρίζουν την περιήγηση σε πολλούς φακέλους, ανιχνεύουν διπλότυπα και παρόμοια αρχεία και εικόνες για τον καθαρισμό του χώρου στο σκληρό δίσκο.

Altap Salamander

Εικόνα 32: λογότυπο του Altap Salamander

Το Altap Salamander είναι ένας δωρεάν ορθόδοξος διαχειριστής αρχείων για Microsoft Windows, εμπνευσμένος από το Norton Commander. Σε αντίθεση με αρκετούς άλλους διαχειριστές αρχείων, έχει μια διεπαφή χρήστη με επίγνωση του περιβάλλοντος του, για παράδειγμα, η κάτω λίστα λειτουργιών αλλάζει με το πάτημα των πλήκτρων τροποποίησης, εμφανίζοντας απλώς το τρέχον διαθέσιμο σύνολο πλήκτρων πρόσβασης. Η ανάπτυξή του ξεκίνησε το 1996 από τον Petr Solin και κυκλοφόρησε ως δωρεάν λογισμικό το 1997. Αρχικά γράφτηκε σε Watcom C++ και αργότερα σε Microsoft Visual C++ 6.0. Το Salamander 2.0 περιλάμβανε υποστήριξη για προσθήκες προβολής και αρχειοθέτησης. Κατά την ανάπτυξη της έκδοσης 2.5, η αρχιτεκτονική των πρόσθετων επεκτάθηκε για την υποστήριξη FTP και άλλων πρωτοκόλλων. Το SDK για την έκδοση 2.5 επιτρέπει στους προγραμματιστές να δημιουργούν νέες προσθήκες προβολής, προσθήκες αρχειοθέτησης, προσθήκες συστήματος αρχείων και εργαλεία (όπως πολλαπλή μετονομασία ή αρχείο σύγκρισης).

Οι παραπάνω εφαρμογές ,κάποιες εκ των οποίων έχουν και δωρεάν εκδόσεις, παρότι είναι πολύ εύχρηστες για την εύρεση αρχείων σε λειτουργικά συστήματα Windows δεν πληρούν τις προαναφερθείσες προϋποθέσεις μας όσων αφορά τις ελάχιστες δυνατές απαιτήσεις σε μνήμη RAM αλλά και την ευκολία μεταφοράς των δεδομένων σε άλλους υπολογιστές.

6. ΠΙΘΑΝΕΣ ΠΡΟΣΘΗΚΕΣ

Λόγω των ευαίσθητων προσωπικών δεδομένων που διαχειρίζεται η εφαρμογή, μας ζητήθηκε από τη διεύθυνση του σχολείου να δημιουργήσουμε μία μη δικτυακή εφαρμογή για να πετύχουμε τη μέγιστη δυνατή ασφάλεια για τις πληροφορίες των αποφοίτων. Παρόλα αυτά είναι εφικτό να προστεθεί κώδικας στον οποίο το σχολείο μπορεί να αποθηκεύει το e-mail και τον κωδικό του αποφοίτου και να στέλνει σε όποιον θέλει το απολυτήριο σε ηλεκτρονική μορφή. Σε περίπτωση που αποφασίσουν να κάνουν online εφαρμογή μπορούμε να πάρουμε κομμάτια του πηγαίου κώδικα και να τα προσαρμόσουμε σε κάποια ιστοσελίδα έτσι ώστε μετά από επικοινωνία του αποφοίτου με το λύκειο, το σχολείο να του στείλει έναν κωδικό που να του επιτρέπει να μπει για συγκεκριμένο χρονικό διάστημα στην ιστοσελίδα και να έχει πρόσβαση στο δικό του απολυτήριο από όποια συσκευή θέλει.

7. ΣΥΜΠΕΡΑΣΜΑΤΑ

Σε αυτή την πτυχιακή εργασία δημιουργήσαμε μία εύχρηστη εφαρμογή με τις λιγότερες δυνατές απαιτήσεις από τους υπολογιστές που θα την φιλοξενήσουν, πολύ εύκολη μεταφορά της εφαρμογής από υπολογιστή σε υπολογιστή, εύκολη και γρήγορη αναβάθμιση για τη συμβατότητά της με μελλοντικά λειτουργικά συστήματα και δικλείδες ασφαλείας για το ενδεχόμενο καταστροφής της βάσης δεδομένων ή και της ίδιας της εφαρμογής. Χάρη σε αυτήν τόσο το προσωπικό του 1ου Γενικού Λυκείου Σερρών όσο και οι παλιοί απόφοιτοι αυτού θα μπορέσουν μελλοντικά να εξοικονομήσουν πολύτιμο χρόνο και ενέργεια, ενώ αποτέλεσε και ένα πολύ χρήσιμο μάθημα για εμάς όσον αφορά το κομμάτι της συνεργασίας, μεταξύ μας αλλά και με το προσωπικό του σχολείου, ενώ αποκτήσαμε και κάποιες χρήσιμες γνώσεις και εμπειρίες τόσο στο κομμάτι της συγγραφής κώδικα όσο και στην επιλογή της κατάλληλης γλώσσας προγραμματισμού και περιβάλλοντος ανάπτυξης ανάλογα με τις ανάγκες των χρηστών, τις δυνατότητες του εξοπλισμού τους και τους τυχόν περιορισμούς όπως ήταν στην περίπτωσή μας η προστασία των προσωπικών δεδομένων των αποφοίτων.

ΚΩΔΙΚΑΣ

Form1.cs

```
using System;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Drawing.Printing;
using System.IO;
using System.Windows.Forms;

namespace _1gelSerrwn
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
      }
      private void Form1_Load(object sender, EventArgs e)
      {
         filldatagrid();
      }
      private void Create_Click(object sender, EventArgs e)
      {
         Students std = new Students();
         std.ShowDialog();
         filldatagrid();
         if (std.Picturetb.Image!=null)
         {
            std.Picturetb.Image.Dispose();
         }
      }
      private void dataGridView1_CellContentClick(object sender, DataGridViewCellEventArgs e)
      {
         Student student = new Student();
         if (dataGridView1.Columns[e.ColumnIndex].Name == "Delete")
         {
            int id = Convert.ToInt32(dataGridView1.CurrentRow.Cells[2].Value.ToString());
            if (MessageBox.Show("Θέλετε να διαγραφεί ο μαθητής?", "Delete Student",
MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)
            {
               if (student.deleteStudent(id))
               {
                  string path = Application.StartupPath;
                  File.Delete(Path.Combine(path +
dataGridView1.CurrentRow.Cells[8].Value.ToString()));
                  MessageBox.Show("Ο μαθητής διαγράφηκε!", "Delete Student",
MessageBoxButtons.OK, MessageBoxIcon.Information);
               }
               else
               {
                  MessageBox.Show("Πρόβλημα δεν μπόρεσε να διαγραφεί ο μαθητής!", "Delete
Student", MessageBoxButtons.OK, MessageBoxIcon.Error);
               }
```

```
            }
        }
        else if (dataGridView1.Columns[e.ColumnIndex].Name == "Print")
        {
            PrintDialog printdlg = new PrintDialog();
            if (printdlg.ShowDialog() == DialogResult.OK)
            {
                PrintDocument printDocument1 = new PrintDocument();
                PrintPreviewDialog printPrvDlg = new PrintPreviewDialog();
                printDocument1.PrintPage += new
PrintPageEventHandler(this.printDocument1_PrintPage);
                printPrvDlg.Document = printDocument1;
                printPrvDlg.ShowDialog();
            }
        }
        filldatagrid();
    }

    private void printDocument1_PrintPage(object sender, PrintPageEventArgs e)
    {
        string path = Application.StartupPath;
        string photo = path + dataGridView1.CurrentRow.Cells[8].Value.ToString();
        Bitmap myBitmap1 = new Bitmap(photo);
        e.Graphics.DrawImage(myBitmap1, e.PageBounds);
        myBitmap1.Dispose();
    }

    private void dataGridView1_DoubleClick(object sender, EventArgs e)
    {
        Students std = new Students();
        std.Idtb.Text = dataGridView1.CurrentRow.Cells[2].Value.ToString().Trim();
        std.Nametb.Text = dataGridView1.CurrentRow.Cells[3].Value.ToString().Trim();
        std.Lnametb.Text = dataGridView1.CurrentRow.Cells[4].Value.ToString().Trim();
        std.Yeartb.Text = dataGridView1.CurrentRow.Cells[5].Value.ToString().Trim();
        std.Schtb.Text = dataGridView1.CurrentRow.Cells[6].Value.ToString().Trim();
        std.Foldertb.Text = dataGridView1.CurrentRow.Cells[7].Value.ToString().Trim();
        string path = Application.StartupPath;
        std.Picturetb.Image =
Image.FromFile(path+dataGridView1.CurrentRow.Cells[8].Value.ToString());
        std.imagepath = path + dataGridView1.CurrentRow.Cells[8].Value.ToString();
        std.Savebt.Text = "Update";
        std.ShowDialog();
        filldatagrid();

        if (std.Picturetb.Image != null)
        {
            std.Picturetb.Image.Dispose();
        }
    }

    private void textBox1_TextChanged(object sender, EventArgs e)
    {
        SqlCommand command = new SqlCommand();
        string sql;
        string[] collumnname = { "FirstName", "LastName", "GraduationYear", "GraduationSchool",
"Folder" };


        try
        {
```

```
            if (string.IsNullOrEmpty(textBox1.Text.Trim()))
            {
                filldatagrid();
                return;
            }
            sql = "SELECT * FROM Students WHERE ";
            string[] words = textBox1.Text.Trim().Split(' ');

            if (words.Length >= 1)
            {
                foreach (string word in words)
                {
                    sql += "(";
                    foreach (string column in collumnname) sql += column + " LIKE N'%" + word + "%'
OR ";
                    sql = sql.Substring(0, sql.Length - 3);
                    sql += ") AND ";
                }

                sql = sql.Substring(0, sql.Length - 5);
                sql += " ORDER BY Id ASC";
                command.CommandText = sql;
                command.Parameters.Clear();
                filldatagrid(command);
            }
        }

        catch (Exception ex)
        {
            MessageBox.Show("ERROR" + ex.Message.ToString());

        }
        finally
        {
            textBox1.Focus();
        }
    }


    public void filldatagrid(SqlCommand cmd = null)
    {
        Student student = new Student();

        SqlCommand command = new SqlCommand();

        string sql = "SELECT * FROM Students ORDER BY Id ASC";

        if (cmd == null)
        {
            command.CommandText = sql;

        }
        else
        {
            command = cmd;
        }

        dataGridView1.DataSource = student.getStudents(command);
    }
```

```
    }
}
```

MY_DB.cs

```
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Linq;
using System.Text;
using System.Threading.Tasks;


namespace _1gelSerrwn
{
    class MY_DB
    {
      private  SqlConnection conn = new SqlConnection(@"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=|DataDirectory|\Database.mdf;Integrated
Security=True");

        public SqlConnection GetConnection
        {
            get
            {
                return conn;
            }
        }

        public void openConnection()
        {
            if (conn.State==System.Data.ConnectionState.Closed)
            {
                conn.Open();
            }
        }

        public void closeConnection()
        {
            if (conn.State == System.Data.ConnectionState.Open)
            {
                conn.Close();
            }
        }
    }
}
```

Student.cs

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.SqlClient;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace _1gelSerrwn
{
   class Student
   {
      MY_DB db = new MY_DB();

      public bool insStudent(String fname, String lname, int year, String school, String folder, String
photo)
      {
         SqlCommand command = new SqlCommand("INSERT INTO
Students(FirstName,LastName,GraduationYear,GraduationSchool,Folder,Photo) VALUES
(@fn,@ln,@year,@sch,@fol,@pic)", db.GetConnection);
         command.Parameters.Add("@fn", System.Data.SqlDbType.NVarChar).Value = fname;
         command.Parameters.Add("@ln", System.Data.SqlDbType.NVarChar).Value = lname;
         command.Parameters.Add("@year", System.Data.SqlDbType.Int).Value = year;
         command.Parameters.Add("@sch", System.Data.SqlDbType.NVarChar).Value = school;
         command.Parameters.Add("@fol", System.Data.SqlDbType.NVarChar).Value = folder;
         command.Parameters.Add("@pic", System.Data.SqlDbType.NVarChar).Value = photo;

         db.openConnection();
         if (command.ExecuteNonQuery() == 1)
         {
            db.closeConnection();
            return true;
         }
         else
         {
            db.closeConnection();
            return false;
         }
      }

      public DataTable getStudents(SqlCommand command)
      {
         command.Connection = db.GetConnection;
         SqlDataAdapter sqlDataAdapter = new SqlDataAdapter(command);
         DataTable table = new DataTable();
         sqlDataAdapter.Fill(table);
         return table;
      }

      public bool updateStudents(int id,String fname, String lname, int year, String school, String
folder, String photo)
      {
         SqlCommand command = new SqlCommand("UPDATE Students SET FirstName=@fn ,
LastName=@ln , GraduationYear=@year , GraduationSchool=@sch , Folder=@fol , Photo=@pic
WHERE id=@Id", db.GetConnection);
         command.Parameters.Add("@id", System.Data.SqlDbType.Int).Value = id;
         command.Parameters.Add("@fn", System.Data.SqlDbType.NVarChar).Value = fname;
```

```
            command.Parameters.Add("@ln", System.Data.SqlDbType.NVarChar).Value = lname;
            command.Parameters.Add("@year", System.Data.SqlDbType.Int).Value = year;
            command.Parameters.Add("@sch", System.Data.SqlDbType.NVarChar).Value = school;
            command.Parameters.Add("@fol", System.Data.SqlDbType.NVarChar).Value = folder;
            command.Parameters.Add("@pic", System.Data.SqlDbType.NVarChar).Value = photo;

            db.openConnection();
            if (command.ExecuteNonQuery() == 1)
            {
                db.closeConnection();
                return true;
            }
            else
            {
                db.closeConnection();
                return false;
            }
        }

        public bool deleteStudent(int id)
        {
            SqlCommand command = new SqlCommand("DELETE FROM Students WHERE
id=@STDID", db.GetConnection);
            command.Parameters.Add("@STDID", System.Data.SqlDbType.Int).Value = id;

            db.openConnection();
            if (command.ExecuteNonQuery() == 1)
            {
                db.closeConnection();
                return true;
            }
            else
            {
                db.closeConnection();
                return false;
            }
        }
    }
}
```

Students.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace _1gelSerrwn
{
    public partial class Students : Form
```

```
{
    public Students()
    {
        InitializeComponent();
    }

    string filepath=null;
    public string imagepath;

    private void Savebt_Click(object sender, EventArgs e)
    {
        Student student = new Student();
        if (Savebt.Text == "Save")
        {
            if (saveb())
            {
                string fname = Nametb.Text.Trim();
                string lname = Lnametb.Text.Trim();
                int year = Convert.ToInt32(Yeartb.Text.Trim());
                string fullname = fname + "_" + lname + "_" + year + ".jpg";
                string school = Schtb.Text.Trim();
                string folder = Foldertb.Text.Trim();
                string path = Application.StartupPath;
                if (!Directory.Exists(Path.Combine(path, "Images")))
                {
                    Directory.CreateDirectory(Path.Combine(path, "Images"));
                }
                System.IO.File.Copy(filepath, path + "\\Images\\" + fullname);
                if (student.insStudent(fname, lname, year, school, folder, "\\Images\\" + fullname))
                {
                    MessageBox.Show("Προστέθηκε νεος μαθητης", "Add student",
MessageBoxButtons.OK, MessageBoxIcon.Information);
                    this.Close();
                }
                else
                {
                    MessageBox.Show("Πρόβλημα δεν προστέθηκε νέος μαθητής", "Add student",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
            }
            else
            {
                MessageBox.Show("Παρακαλώ γεμίστε τα κενά πεδία", "Add student",
MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
            }
        }
        else if (Savebt.Text == "Update")
        {
            int id = Convert.ToInt32(Idtb.Text);
            if (saveb())
            {
                string fname = Nametb.Text.Trim();
                string lname = Lnametb.Text.Trim();
                int year = Convert.ToInt32(Yeartb.Text.Trim());
                string fullname = fname + "_" + lname + "_" + year + ".jpg";
                string school = Schtb.Text.Trim();
                string folder = Foldertb.Text.Trim();
                string path = Application.StartupPath;
                if ((imagepath != path + "\\Images\\" + fullname) && (filepath == null))
                {
```

```
                Picturetb.Image.Dispose();
                File.Move(imagepath, path + "\\Images\\" + fullname);
            }
            if (filepath != null)
            {
                System.IO.File.Copy(filepath, path + "\\Images\\" + fullname, true);

            }
            if (student.updateStudents(id, fname, lname, year, school, folder, "\\Images\\" + fullname))
            {
                MessageBox.Show("Ο μαθητής ενημερώθηκε!", "Edit student",
MessageBoxButtons.OK, MessageBoxIcon.Information);
                this.Close();
            }
            else
            {
                MessageBox.Show("Πρόβλημα! Δεν μπόρεσε να γίνει ενημέρωση μαθητή!", "Edit
student", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }
        else
        {
            MessageBox.Show("Παρακαλώ γεμίστε τα κενά πεδία", "Edit student",
MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
        }
    }
}

bool saveb()
{
    if ((Nametb.Text.Trim() == "") || (Lnametb.Text.Trim() == "") || (Yeartb.Text.Trim() == "") ||
(Schtb.Text.Trim() == "") || (Foldertb.Text.Trim() == "") || (Picturetb.Image == null))
    {
        return false;
    }
    else
    {
        return true;
    }
}

private void Uploadbt_Click(object sender, EventArgs e)
{
    OpenFileDialog dlg = new OpenFileDialog();
    dlg.Filter = "JPG Files(*.jpg)|*.jpg|PNG Files(*.png)|*.png|ALL Files(*.*)|*.*";
    if (dlg.ShowDialog() == DialogResult.OK)
    {
        if (Picturetb.Image != null)
        {
            Picturetb.Image.Dispose();
        }
        Image img = Image.FromFile(dlg.FileName);
        filepath = dlg.FileName;
        Picturetb.Image = img;
    }
    dlg.Dispose();
}

private void Yeartb_KeyPress(object sender, KeyPressEventArgs e)
{
```

```
            char ch = e.KeyChar;
            if (!Char.IsDigit(ch) && ch != 8 && ch != 46)
            {
                e.Handled = true;
            }
        }
    }
}
```

Database.mdf

```
CREATE TABLE [dbo].[Students] (
    [Id]              INT          IDENTITY (1, 1) NOT NULL,
    [FirstName]       NVARCHAR (30)  NOT NULL,
    [LastName]        NVARCHAR (30)  NOT NULL,
    [GraduationYear]  INT          NOT NULL,
    [GraduationSchool] NVARCHAR (50)  NOT NULL,
    [Folder]          NVARCHAR (10)  NOT NULL,
    [Photo]           NVARCHAR (MAX) NOT NULL,
    PRIMARY KEY CLUSTERED ([Id] ASC)
);
```

Βιβλιογραφία

1) C# Fundamentals: C# 9 and .NET 5 by Adam Seebeck
2) Exploring Blazor: Creating Hosted, Server-side, and Client-side Applications with C# 1st ed. Edition by Taurius Litvinavicius
3) C# 8.0 and .NET Core 3.0 – Modern Cross-Platform Development: Build applications with C#, .NET Core, Entity Framework Core, ASP.NET Core, and ML.NET using Visual Studio Code, 4th Edition Illustrated Edition by Mark J. Price
4) Hands-On Parallel Programming with C# 8 and .NET Core 3: Build solid enterprise software using task parallelism and multithreading by Shakti Tanwar