**ПРОФЕСИОНАЛНА ГИМНАЗИЯ**

**„ПРОФ. Д-Р АСЕН ЗЛАТАРОВ”**

**ВИДИН**



**ТЕМА НА ДИПЛОМНИЯ ПРОЕКТ:**

**РАЗРАБОТКА НА ПРИЛОЖЕНИЕ БАЗА ДАННИ ЗА ЗАВЪРШИЛИ УЧЕНИЦИ**

**Ивайло Георгиев Петров**

(трите имена на ученика/ученичката)

Професия 481030 „Приложен програмист“

Специалност 4810301 „Приложно програмиране“

**…………………………………………**

(име и фамилия на ръководителя/консултанта)

**СЪДЪРЖАНИЕ**

I. УВОД (Heading 1)

II. ОСНОВНА ЧАСТ (Heading1)

1. ТЕОРЕТИЧНА ЧАСТ (Heading 2)

1.1. Точка 1 (heading 3)

1.2. Точка 2 (heading 3)

1.3. Точка 3 (heading 3)

1.4. Точка 4 (heading 3)

2. ПРАКТИЧЕСКА ЧАСТ (Heading 2)

III. ЗАКЛЮЧЕНИЕ (Heading 1)

IV. ИЗПОЛЗВАНА ЛИТЕРАТУРА (Heading 1)

1. **УВОД**

Този дипломен проект е свързан с разработката на приложение, което съхранява и предоставя информация за бивши възпитаници на гимназия. Целта на приложението е, да улесни контакта с вече завършили ученици и персонала на гимназията. Често, години след като са завършили, учениците прекъсват връзка с бившите си учители, а също и съученици. Това е пропуск, тъй като връзката с училището може да бъде много полезна и за двете страни, както за обмяна на опит, така и за професионални възможности.

Точно този проблем трябва да реши приложението – да даде информация за местоживеенето и професионалната реализация на бившите ученици на гимназията, като по този начин даде възможност за връзка с тях.

Приложението има форма за регистрация на нов потребител, с име и имейл, който може да бъде администратор или ученик. Има също форма за въвеждане на данни за бивш ученик – име, адрес, имейл, телефон, месторабота, професия, класен ръководител. Приложението поддържа форма за вписване на потребител с имейл и парола. Има търсачка, чрез която могат да се търсят записи на ученици по зададени критерии.

Използваните технологии за реализацията на проекта са език за програмиране на програмната логика C#, база данни Microsoft SQL Server 2022, Visual Studio 2022 с проект на база Win Forms (Net Core). Използван е подход MVC (Model-View-Controller) и ORM (Object-Related-Model) при изграждането на базата данни.

Очакваните резултати са приложението успешно да записва и предоставя информация за ученици в база данни, като по този начин улесни контакта между текущите учители и ученици, с бившите възпитаници на гимназията. Очаква се приложението да работи без възможност за грешки като дублирани или повредени записи.

1. **ОСНОВНА ЧАСТ**
2. **ТЕОРЕТИЧНА ЧАСТ**

**Част 1 – технически изисквания към проекта**

За разработката на проекта, използваме Visual Studio версия 2019, WinForms .NET Core тип приложение. Техническите изисквания, на които трябва да отговаря компютърната система за разработка на проекта са следните :

**Операционна система**

* Windows 2000
* Windows XP
* Windows Vista
* Windows 7
* Windows 8
* Windows 10
* Windows Server (от 2003 до последна версия)

**Хардуерни изисквания**

* Процесор: x86 или x64
* RAM : 512 MB (минимум), 1 GB (препоръчителни)
* Хард диск: до 4 GB може да се изискват.

**Среда за разработка**

* Microsoft Visual Studio 2008/2010/2012/2013/2015/2017/2019/2022
* .NET Framework 2.0/3.5/4.0/4.5/4.5.1/4.6
* .NET Core се поддържа от версия [2019 Volume 1 release](https://help.syncfusion.com/windowsforms/release-notes) (17.1.0.32).

**Системни изисквания към среда за разработка Visual Studio 2019**:

**Операционна система** от следните:

* Windows 11 version 21H2 or higher: Home, Pro, Pro Education, Pro for Workstations, Enterprise, and Education
* Windows 10 version 1703 or higher: Home, Professional, Education, and Enterprise (LTSC and S are not supported)
* Windows Server 2019: Standard and Datacenter
* Windows Server 2016: Standard and Datacenter
* Windows 8.1 (with Update 2919355): Core, Professional, and Enterprise
* Windows Server 2012 R2 (with Update 2919355): Essentials, Standard, Datacenter
* Windows 7 SP1 (with latest Windows Updates): Home Premium, Professional, Enterprise, Ultimate

**Хардуерни изисквания**:

* 1.8 GHz или по-бърз процесор. Препоръчва се поне четириядрен
* 2 GB RAM; 8 GB RAM препоръчително
* Място на хард диска: Минимум of 800MB до 210 GB в зависимост от избраните пакети за инсталация. За нашите нужди между 20 и 50 GB свободно място.
* Скорост на записващото устройство: за да се подобри скоростта на работа, се препоръча инсталацията на SSD (solid state drive).

Видеокарта, която поддържа минимум разделителна способност 720p (1280 х 720); Visual Studio ще работи най-добре с разделителна способност на екрана WXGA (1366 х 768) или по висока.

* **Дейности по събиране и съхранение на информация**

За нуждите на проекта, е необходимо да се събере и съхрани информация за завършили ученици в нашата гимназия. Проведената справка показва, че част от тази информация може да бъде намерена в архива на училището. Имената, професията и класният ръководител, с който се е обучавал ученика, могат да бъдат взети от архива на училището. От друга страна, актуалните данни, като телефонен номер, имейл, месторабота и текущо образование, трябва да бъдат набавени чрез контакт с ученика или негови близки и познати. Трябва да имаме предвид, че личните данни са чувствителна информация, която подлежи на регулация от Българските и Европейски власти. Личните данни, които можем да съхраняваме, трябва да бъдат внимателно преценени, за да се избегнат нарушения, особено след като учениците вече са завършили и не са част от системата на средното образование, а и пълнолетни граждани. Вероятно ще е необходимо личното съгласие на бившите възпитаници на гимназията, дали техните данни могат да бъдат съхранявани и споделени с екипа на училището.

Веднъж набавени, данните могат да бъдат въвеждани в приложението от оператор, с администраторски профил. Приложението също ще позволява редакция на данните, ако възникне промяна или се установи грешка в предоставените данни.

**Част 2 – използвани технологии в проекта**

**Какво представляват WinForms приложенията?**

Windows Forms е графична софтуерна рамка за създаване на Windows desktop приложения, част е от .NET Framework на Microsoft. Тя предоставя един от най-продуктивните начини да се създават desktop приложения, базирани на графичният дизайнер на Visual Studio. Функциите, като например изтегли-и-пусни на графичните контроли, правят изграждането на приложения лесно. С Windows Forms, могат да се разработват богати на функции и добре изглеждащи приложения, които са лесни за пускане в експлоатация, обновяване. Могат да работят както офлайн, така и свързани към интернет.

С приложенията, направени на Windows Forms, можем да изискваме въвеждане на информация от потребител, да възпроизвеждаме информация в дадена форма, както и да комуникираме с отдалечени компютри в мрежата. В Windows Form, ***форма***е визуален елемент, на който извеждаме информация на потребителя. Традиционно, построяването на такова приложение е свързано с добавянето на контроли от инструментите на Visual Studio, които откликват на действия от страна на потребителя. Като например – клик на мишката, движение на курсора, натискане на бутон, въвеждане на текст, отмятане на радио бутон и др. Когато дадено действие бъде извършено, това създава **събитие** в програмата. Събитията могат да бъдат обработени в програмна среда, така че да се получи очакван резултат.

Windows Forms приложенията могат да осъществяват връзка с база данни за четене или запис на информация. Това става, чрез т.н. Data Sources и BindingSource методи, т.е. източник на данни и свързващ компонент. В зависимост от използваният подход, може първо да се дефинират класовете (обектите) в приложението, а на база тях да бъдат създадени таблиците и съответните колони в базата данни (code first подход) или обратно – на база изградена база данни, да се опишат необходимите класове в приложението (database first).

Windows Forms класовете са имплементирани в библиотеката System.Windows.Forms и са достъпни през едноименното пространство от имена.

**Форми**

Формите (прозорците) в Windows Forms наследяват класа Form. За създаване на нова форма Visual Studio предоставя Windows Form файлов шаблон. Някои от свойствата на формите са:

* **Size** – това е размерът на формата;
* **Text** – това е текстът, който се изписва в заглавната лента;
* **StartPosition** – определя къде да бъде разположена инстанция на формата при създаването си (на определеното по подразбиране от Windows място, в центъра на екрана, на фиксирана от разработчика позиция);
* **FormBorderStyle** – определя стила на рамката на формата (дебела, тънка, разрешаваща промяна на размерите или не и др.);
* **MinimizeBox**, **MaximizeBox** – указват дали да бъдат видими съответно бутона за минимизиране и максимизиране на формата;
* **ShowInTaskbar** – указва дали формата да има асоцииран с нея бутон в лентата със задачи на Windows;
* **ShowIcon** – указва дали формата да има иконка в лявата част на заглавната си лента;
* **AcceptButton** – указва бутон от формата, който да бъде автоматично натиснат при натискане на клавиша Enter;
* **CancelButton** – указва бутон от формата, който да бъде автоматично натиснат при натискане на клавиша Escape.

По подразбиране Windows Forms приложенията се стартират с инстанция на главна форма, така че изпълнението на приложението автоматично да приключи при нейното затваряне.

За показване на екрана на допълнителни форми (които не са модални) се използва методът **Show**().

**Контроли**

Windows Forms предоставя стандартния набор от Windows контроли – Label, TextBox, ListBox, ComboBox, CheckBox, RadioButton, Button, ListView, TreeView. MonthCalendar и DateTimePicker се използват за избор на дата и/или час. RichTextBox се използва за реадктиране на текст с богато форматиране (цвят, шрифт и т.н.). WebBrowser служи за визуализиране на HTML документи.

Повечето контроли имат следните свойства:

* **Parent** – това е родителската контрола (контейнер) на контролата.
* **Size** – това е размерът на контролата;
* **Location** – това са координатите на горния ляв ъгъл на контролата спрямо горния ляв ъгъл на контейнера ѝ;
* **Margin** – това е дебелината на рамката на контролата (от четирите страни);
* **Dock** – указва дали контролата да бъде „прилепена” към някой от ръбовете на контейнера (заемайки цялата дължина на ръба); възможно е и да запълва цялото свободно пространство в контейнера (DockStyle.Fill);
* **Anchor** – позволява „закачване” на контролата за един или повече от ръбовете на контейнера, така че при промяна размерите на контейнера, разстоянието между закачените ръбове на контролата и контейнера да остава едно и също;
* **Text** – това е текста на контролата;
* **Enabled** – указва дали взаимодействието на потребителя с контролата е разрешено или не;
* **Visible** – указва дали контролата е видима или не;
* **ForeColor** – това е цвета на текста;
* **BackColor** – това е цвета на фона;
* **Font** – това е шрифта на текста;
* **TabIndex** – това е поредния номер на контролата в рамките на контейнера ѝ; определя реда на обхождане на контролите в клавиша Tab;
* **Tag** – това е произволен обект, закачен към контролата.

Всяка контрола предоставя и специфични за нея свойства, както и множество събития, свързани с функционалността ѝ.

* Контейнери

Контейнерите са специални контроли, които могат да съдържат други контроли.

* **Panel**

Това е най-простият контейнер – служи за невизуално групиране на контроли.

* **GroupBox**

Този контейнер служи за визуално групиране на контроли.

* **FlowLayoutPanel**

Този контейнер автоматично подрежда съдържащите се в него контроли една след друга (или една под друга).

* **SplitContainer**

Този контейнер разделя видимата си област на две части (хоризонтално или вертикално), границата между които може да се премества от потребителя.

* **TabControl**

Този контейнер предоставя възможност за странициране на видимата си област.

* **TableLayoutPanel**

Това е контейнер, който позволява дефинирането на редове и колони с абсолютни или относителни височини/ширини. За всяка от съдържащите се в него контроли се указва клетка, в която контролата да бъде разположена. В допълнение може да се посочи дали контролата да заема повече от един ред или колона.

* **Събития**

Обработчици на събития се създават и закачат лесно през визуалния редактор на формата. Всяко събитие предоставя информация за контролата или компонентата, която е изпратила събитието, както и специфични за събитието параметри.

* **Модални диалози**

Модалните диалози са просто форми, който са показани модално с метода ShowDialog() вместо Show(). Всяка форма има свойство DialogResult, което по подразбиране има стойност DialogResult.None. Когато тази стойност бъде променена, диалогът автоматично се затваря и стойността се предава като резултат на ShowDialog() метода.

Бутоните също имат свойство DialogResult. Ако стойността му е различна от DialogResult.None, след натискане на бутона свойството на формата автоматично се приравнява на свойството на бутона, затваряйки модалния диалог.

* **Data Binding**

При попълване на списъчна контрола (ListBox и ComboBox) може да се използва нейното свойство DataSource, което приема стойности от тип IList или IListSource. Контролата автоматично добавя всички елементи на източника в своя списък от елементи. В допълнение могат да се използват свойствата DisplayMember и ValueMember, за да се посочи съответно кое свойство на елемент от източника да се използва за визуализиране на елемента в текстова форма и кое – за генериране на стойност на свойството SelectedValue (по подразбиране се използва целият елемент).

Подходът **ORM –** object relational mapping е техника в компютърното програмиране, която се ползва за преобразуване на данни между системи от несъвместим типове, чрез обектно-ориентирани езици за програмиране.

Защо **ORM** е полезна техника?

Основно предимство на тази техника е спестяването на време, чрез предотвратяване на повтарянето на **SQL** заявки. SQL (structured query language) заявките са специален синтаксис за обръщение към различни системи от бази данни. ORM техниката осъществява връзка между таблиците и колонките в базата данни и моделът от класове в обектно-ориентиран език за програмиране. Обекти, дефинирани в приложението, могат да бъдат преобразувани в таблица в базата данни. Освен това, методи за запис, редакция и изтриване на обекти в приложението, могат да генерират SQL заявки за същите операции в базата данни.

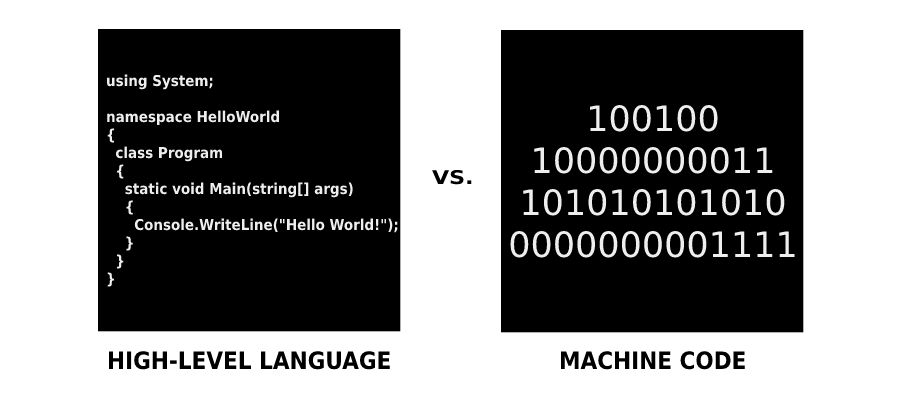
Друго важно предимство е възможността за проследяване на промените, направени по базата данни. Съществуват библиотеки, които специализират в ORM техниката, които записват данните от създаването им, до последните промени. В контекста на **Microsoft** .**NET** Платформата, техниката ORM се нарича **Entity Framework Core**.

Езикът за програмиране, използван в този Windows Forms проект, е **C#**.

**Какво представлява езикът C#?**

C# се произнася като „Си Шарп“. Той е създаден от Microsoft през 2001 г., а последното обновление за него излезе в края на 2021, т.е. това е един актуален език. Може да се каже, че C# e Microsoft-ският отговор на програмния език Java, продукт на компанията Sun Microsystems (понастоящем собственост на Oracle Corporation).

C# е обектно-ориентиран език, т.е. той изгражда системи посредством обекти, които взаимодействат помежду си. Обектно-ориентираното програмиране се отличава от традиционното програмиране, при което програмата е просто списък от инструкции, които компютърът изпълнява.

C# е програмен език от високо ниво. Това го прави изключително достъпен за начинаещи програмисти. Като език от високо ниво, той е по-лесно четим и разбираем, тъй като е по-близо до хората, отколкото до машините (които, както вероятно знаеш, говорят само с нули и единици). Това дава възможност на програмиста да се фокусира върху разработката, правейки процеса по-лесен, бърз и разбираем.

Освен това, C# разчита и на т.нар. garbage collector, който се грижи автоматично за правилното управление на паметта като ресурс, който използваш по време на разработване. Това дава изключително много свобода на програмистите, освобождава ти време и премахва излишни затруднения в процеса на работа.

**Къде и за какво се използва C#?**

Днес С# е един от най-използваните езици за програмиране. Той е език за back-end разработка, т.е. се грижи за сървърната логика, a през технологията Blazer вече опитва да се конкурира с JavaScript за целите на front-end разработката. Ето какво може да се създава със C#:

**1. Десктоп приложения**

От Microsoft използват C# във всички свои продукти в това число всичките си десктоп приложения за Windows 8 и 10, т.е. езикът става все по-актуален и широкоизползван, което дава повече шансове за реализация.

**2. Уеб приложения**

ASP.NET е технологична рамка с отворен код, която също е част от .NET. Бидейки част от едно и също семейство с програмния език C#, това улеснява тяхното взаимодействие и дори го предполага. Това означава, че от Microsoft се стремят да предразположат разработчиците да използват колкото се може повече MS продукти в разработването на своите приложения и платформи. Доброто взаимодействие между C# и уеб фреймуърка ASP.NET правят програмния език подходящ и за уеб сайтове, платформи и уеб приложения.

3**. Мобилни и cross-platform мобилни разработки**

C# е подходящ също и за създаване на мобилни и междуплатформени приложения. Visual Studio – IDE-то на Microsoft e съвместимо с Xamarin – платформа за разработка на native Android, iOS и Windows приложения.

**4. Игри**

Разработката на игри със C# също е една от възможностите за реализация с езика. Един от най-популярните гейм енджини, подходящ както за PC, така и за конзолни и мобилни приложения, е Unity, който е микс от няколко езика от семейството на „С“ – С, С++ и разбира се С#. С това далеч не се изчерпват игровите енджини, писани частично или изцяло на C# и това също е една добра насока за развитие.

**Какво е MVC**

**MVC** е съкратено от “**Model-view-controller**”. Представлява архитектурен шаблон при програмния дизайн. Той отговаря за разделянето на бизнес логиката на три взаимосвързани части: “Model”, “View” и “Controler”.  Разделяне на бизнес логиката от графичния интерфейс и данните в дадено приложение.  Какво представляват те и какви са техните функции?

* „**Model**“ – ядрото на приложението, предопределено от областта, за която се разработва; обикновено това са данните от реалния свят, които се моделират и над които се работи – въвеждане, промяна, показване и т.н. Трябва да се прави разлика между реалния обкръжаващ свят и въображаемият абстрактен моделен свят, който е продукт на разума, който се възприема като твърдения, формули, математическа символика, схеми и други помощни средства. Например в банково приложение това са класовете, описващи клиентите, техните сметки, транзакциите, които са осъществили и т.н., както и класовете за извършване на операции над тези обекти (engines).
* „**View**“ – най-общо казано това е изходна част от софтуера, която визуализира наличните, обработени данни. Чрез View компонентите, данните от приложението се извеждат или въвеждат в уеб страницата. Например, когато имаме поле за писане или таблица с данни, тези данни могат да бъдат изпратени или заредени във формата, с натискането на бутон.
* „**Controller**“ – тази част от сорс кода (клас или библиотека), която взима данните от модела или извиква допълнителни методи върху модела, предварително обработва данните, и чак след това ги дава на изгледа. Например може да бъде създаден един малък обект, в който да бъдат сложени данните за транзакцията – като в контролера бъдат взети данните за транзакцията от модела, бъдат преведени датите от UNIX формат в четим от потребителя формат, бъде преобразувана валутата от долари в евро например, бъде закръглено до втория знак вместо да се виждат данните както са в модела (и в базата) до 10-ия. Също така когато се прави уеб графичен интерфейс това би довело до много лесна модификация на HTML кода дори от човек, който не е програмист – той ще гледа на шаблона просто като на обикновена HTML страница.

**Предимства**:

* Моделът е независим от контролера и изгледа.
* Моделът може да бъде планиран и осъществен независимо от другите части на системата.
* За един и същи модел могат да бъдат осъществени различни изгледи (интерфейси) – например уеб интерфейс и нативен интерфейс към Facebook.
* Контролерът и изгледът могат да бъдат променени, без да се налага промяна в модела.

**Недостатъци**:

* Софтуерната система достига ново ниво на сложност, което я прави по-трудно разбираема за софтуерните разработчици, работещи по нея.

**Възможности за създаване на база данни с MySQL Workbench.**

**MySQL Workbench** е универсално графично приложение, подходящо за архитекти на база данни, разработчици и администратори. Подходящо е за администриране, разработка на база данни, изпълнение на SQL заявки, конфигурация на сървъри, администрация на потребители на базата данни, създаване на резервни копия, и други. Програмата е налична за Windows, Linux и MacOS.

**Дизайн на база данни**

MySQL Workbench позволява на администратор или разработчик на база данни да визуализира, моделира и генерира модел на база данни. Включва всичко, което е необходимо за създаването на сложни ER диаграми, както и ключови функции за създаването на документация и отбелязването на промени, които типично изискват много усилия.

Разработка

MySQL Workbench предоставя графични инструменти за създаване, изпълнение и оптимизиране на SQL заявки. SQL редактора има цветово подчертаване на синтактичните правила, автоматично коригиране и попълване, преизползване на блокове от код и история на използваните SQL заявки. Панелът за връзки помага на разработчиците бързо да се свързват и управляват по няколко бази данни.

**Създаване на база данни с MySQL Workbench**

Съществуват няколко възможности за съхранение на данни с MySQL Workbench. Единият подход е, чрез SQL команди, в прозорец за създаване на заявки. Дизайна на базата данни може да бъде планиран отделно, а в средата на MySQL Workbench да бъде само изпълнен. Втори вариант обаче е, да използваме графичната среда за дизайн и планиране на база данни. Да бъде създадена диаграма, включваща имената, колоните и връзките между необходимите таблици. Кодът за генериране на така описаната база данни, ще бъде автоматично генериран. Така създадената база данни може да бъде редактирана или управляване с графичните инструменти на MySQL Workbench. Също така, могат да бъдат добавяни и редактирани записи в таблиците. В нашият проект обаче, записите на данни ще се случват не в MySQL Workbench, а чрез WindowsForms приложението, чрез връзка с базата данни.

**Част 3 – Създаване на приложението и контрол на версиите**

При процесът на работа по проекта, наложително е да имаме някаква форма за контрол на версиите на файловете. По този начин се проследяват промените и могат да се отстраняват грешки и несъвършенства по приложението, с възможност да се виждат хронологично.

За такава система, без избрана **Github.**

**GitHub** e уеб базирана услуга за разполагане на софтуерни проекти и техни съвместни разработки върху отдалечен интернет сървър в т.нар. хранилище (software repository). Базира се на Git системите за контрол и управление на версиите. Услугата може да бъде както платена за частни проекти, така и безплатна за т.нар. проекти с общодостъпен код, като и в двата случая потребителите могат да ползват всички възможности на услугата. Сайтът предоставя функционалност за изграждане на социални мрежи за споделяне на софтуерни програми и за работа с тях, като например индивидуално участие на всеки програмист в групови разработки, както и възможност за наблюдение върху работата на всеки участник в екипа.

Според условията на услугата, ако обемът от данни, трансферирани от даден клиент, значително надхвърля обичайната големина на трафика на останалите клиенти на GitHub, хостинг услугата към този клиент може да бъде незабавно прекратена или ограничена, докато трансферът на данни не се нормализира. Освен това, макар да няма конкретно ограничение, препоръчително е максималният размер на хранилище да е един гигабайт.

В контекстът на нашият проект, основните ползи от тази система е проследяването на промените по кода, хронология на промените и удобство при трансфер на файлове.

**Част 4 – функционалност на проекта**

Приложението, което представям в този проект, съхранява, обработва и визуализира данни за бивши възпитаници на гимназия. С помощта на горе упоменатите технологии, приложението бе планирано спрямо нуждите на гимназията. Данните, които се съхраняват за бившите ученици са следните :

* Три имена на ученика
* Актуален адрес
* Телефон
* Специалност, в която ученика се е обучавал
* Бивш класен ръководител на ученика
* Година на завършване
* Дали работи и ако да – къде
* Дали учи и ако да – къде
* Имейл адрес

За целта, бяха дефинирани и съответните форми и елементи в приложението, за съхраняване, редактиране и изтриване на данни. Необходимо бе и избор на подходящ тип данни на променливите.

**ПРАКТИЧЕСКА ЧАСТ**

**Аналитична част**

Съхраняването на данни изисква връзка с база данни. За целта на приложението, в конфигурационния файл, е необходимо да се запише т.н. „connection string“. Текстът, записан в променлива, за връзка с база данни, съдържа следното :

public static string ConnectionString = "Data Source=.\\SQLEXPRESS;Initial Catalog=studentsdb;Integrated Security=True";

SqlConnection sqlCon = new SqlConnection(ConnectionString);

Където:

**Public** **static** **string** – ниво на достъп и тип на данните на променливата

**ConnectionString** - Име на променливата

**Data** **Source**=.\\SQLEXPRESS - Източник на данни, път до сървъра на базата данни

**Initial** **Catalog**=studentsdb - Име на базата данни

Integrated Security=True - Ниво на автетинкация

След декларирането на данните, необходими за връзка с базата данни, се създава нова SQL връзка, записана в нова променлива от тип SqlConnection. От тук натам, за всяка нужда от връзка с базата данни, ще се ползва този тип връзка.

За съхраняването на записи за бивши ученици, трябва да се прецени внимателно какъв тип данни са най-подходящи, за отделните колони в таблиците в базата данни. Необходими са и определен набор от таблици, всяка от които да съхранява определен тип данни. Една за бивши ученици и една за регистрирани потребители. За регистрация на нов потребител, се ползва [форма, която може да се види в част „приложение“](#регистрация).

Съхраняването и редакцията на данните трябва да е право само на потребител тип администратор. За това, в таблицата „registration“ има и колонка „**isAdmin**“, която е от тип **bit** – т.е. може да има стойност 0 – потребителят не е администратор или стойност 1 – потребителят е администратор. В зависимост от ролята на потребителя, се зареждат различни форми в интерфейса на приложението. Администраторите виждат контролите за добавяне, редакция, изтриване и търсене на записи, а обикновените потребители, могат само да търсят и разглеждат [подробности](#подробности) за записите.

Проектна част

Заключителна част

**ИЗПОЛЗВАНА ЛИТЕРАТУРА:**

[*https://help.syncfusion.com/windowsforms/installation-and-upgrade/system-requirements*](https://help.syncfusion.com/windowsforms/installation-and-upgrade/system-requirements)

[*https://www.telerik.com/blogs/dotnet-basics-orm-object-relational-mapping*](https://www.telerik.com/blogs/dotnet-basics-orm-object-relational-mapping)

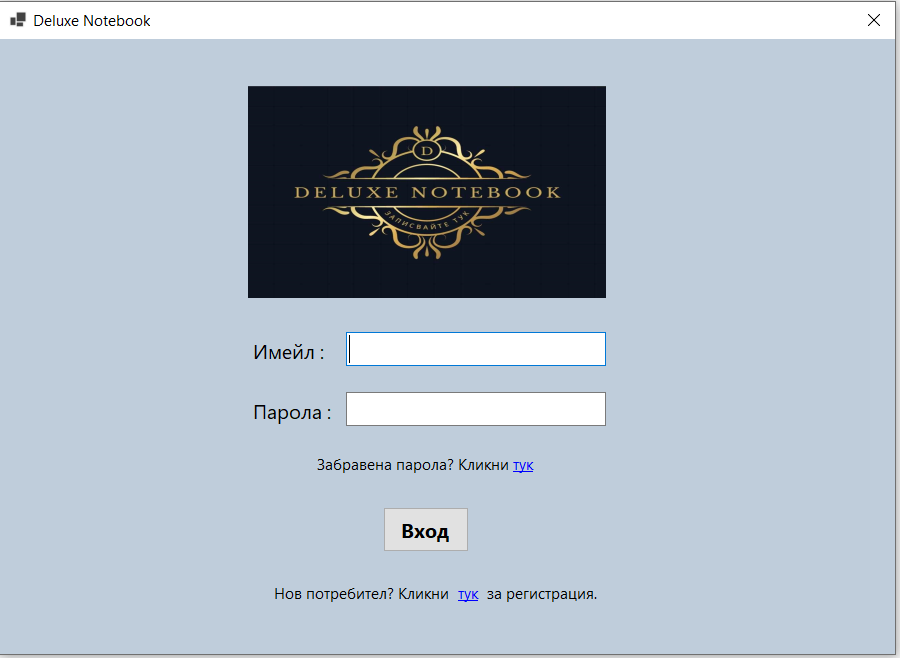
[*https://nakov.com/blog/2015/10/17/ruseconf-2015-orm-technologies-entity-framework/*](https://nakov.com/blog/2015/10/17/ruseconf-2015-orm-technologies-entity-framework/)

[*https://www.geeksforgeeks.org/introduction-to-c-sharp-windows-forms-applications/*](https://www.geeksforgeeks.org/introduction-to-c-sharp-windows-forms-applications/)

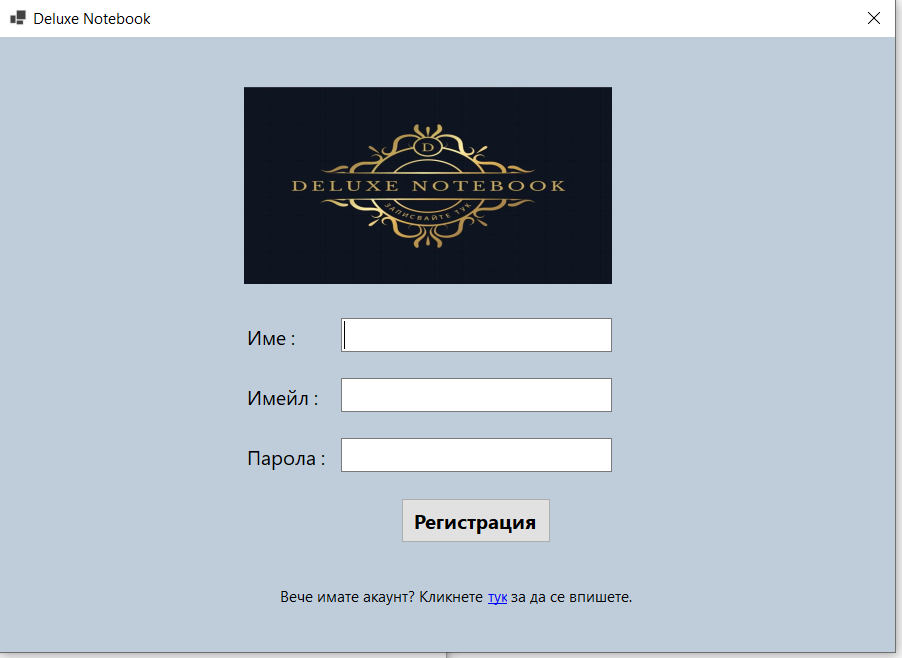
**ПРИЛОЖЕНИЯ**

В тази част на проекта, са представени екранни снимки от интерфейса на приложението.

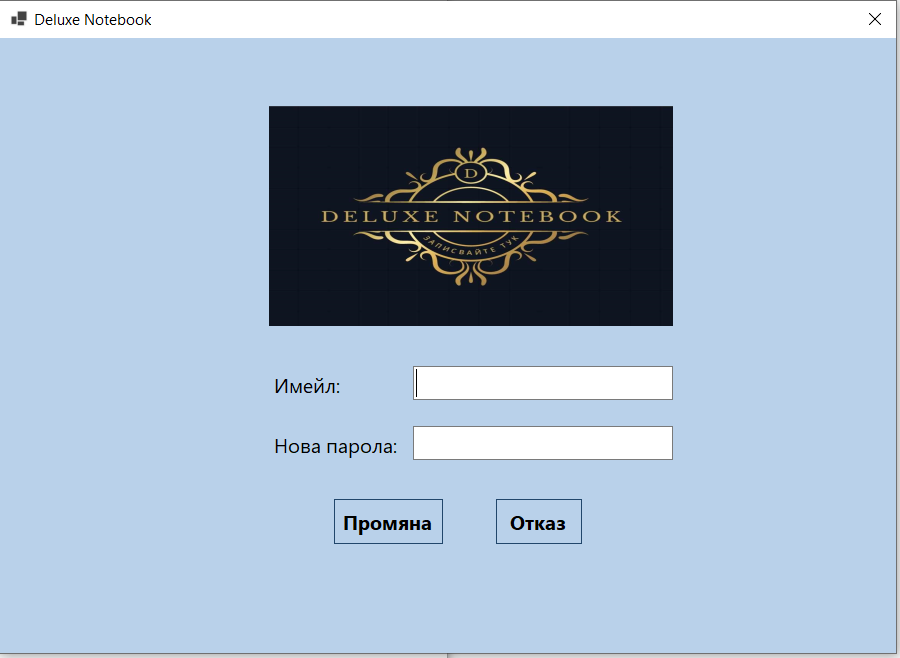
**Начален екран**



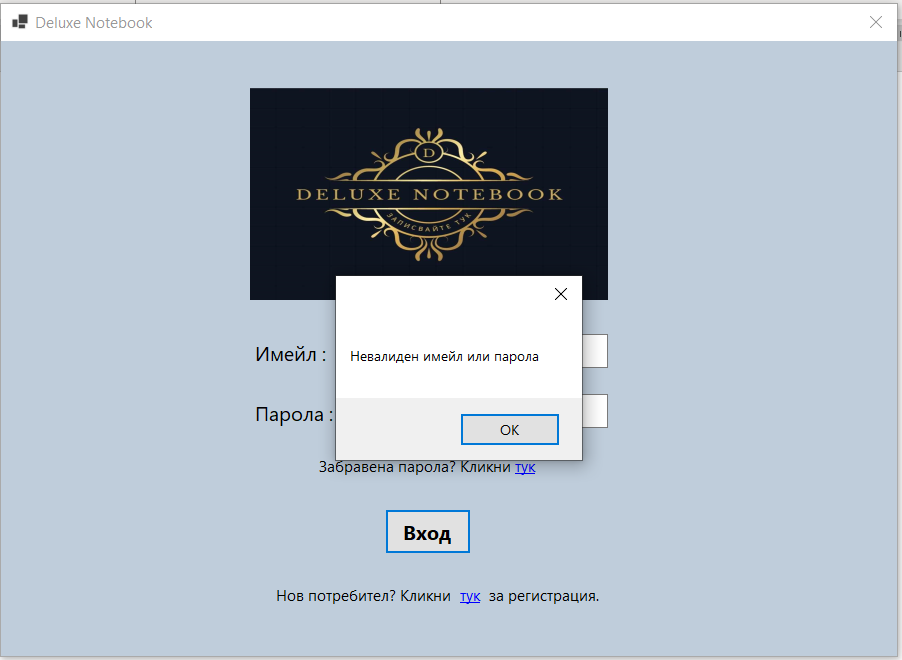
**Форма за регистрация**



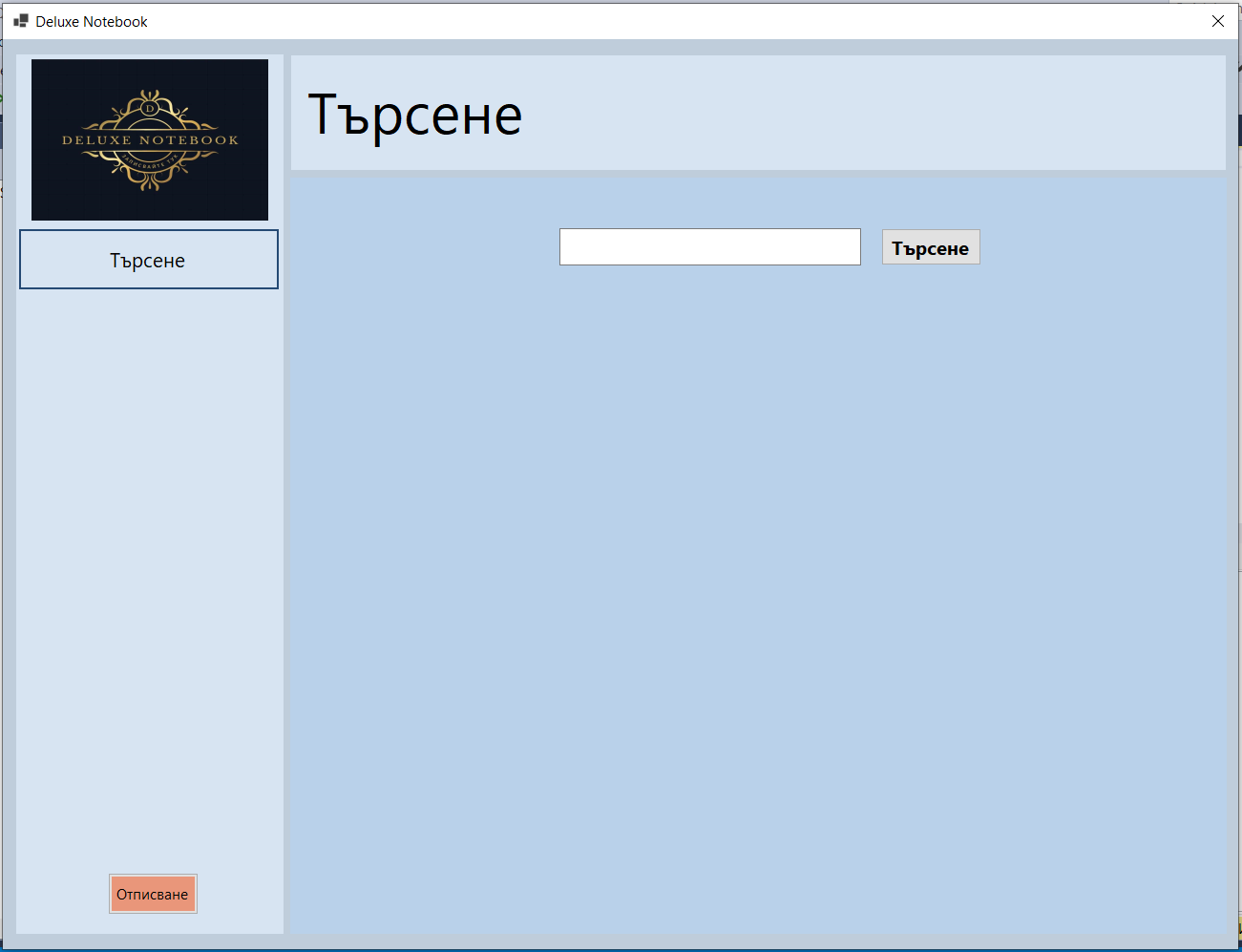
**Забравена парола**



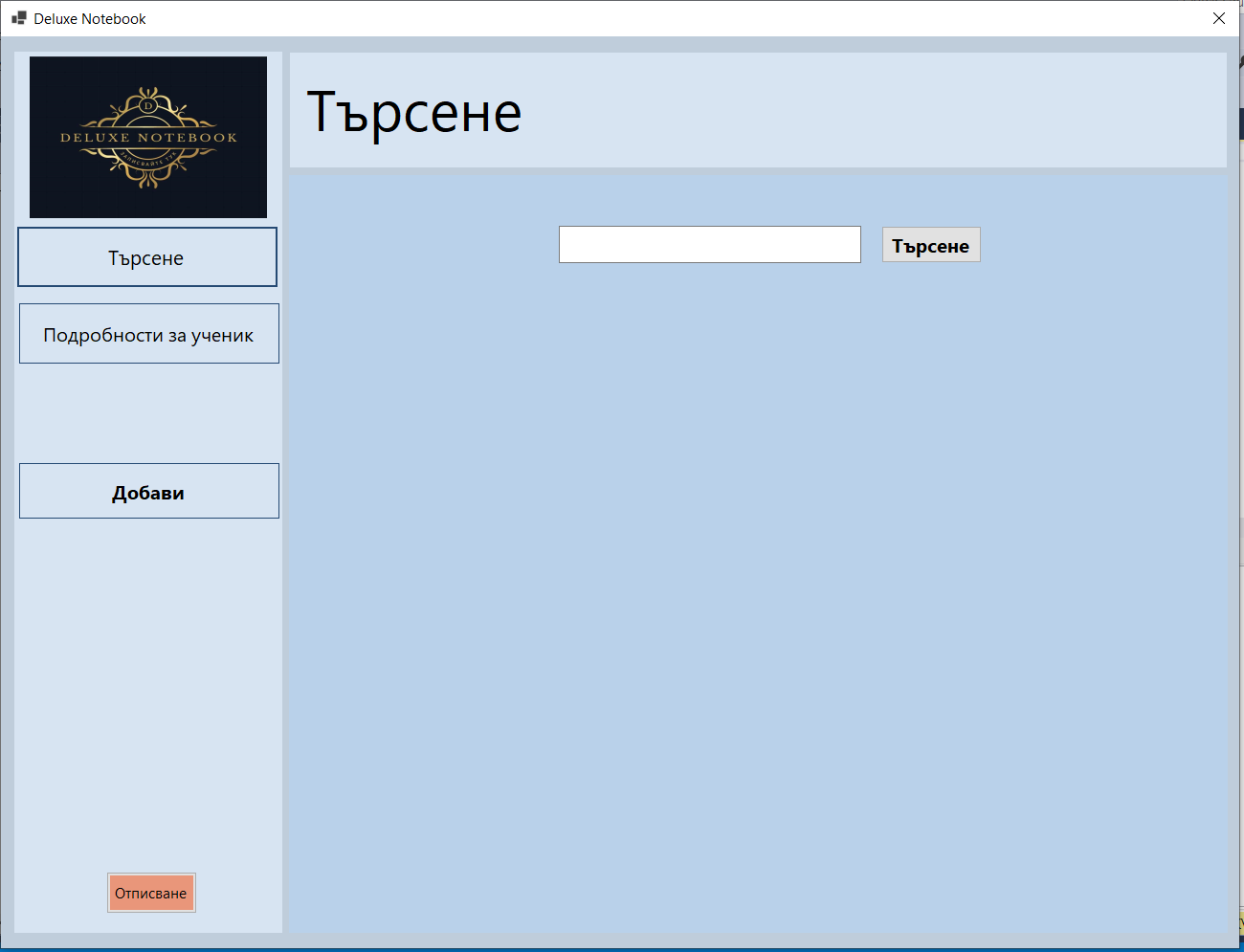
**Невалиден вход (име/парола)**



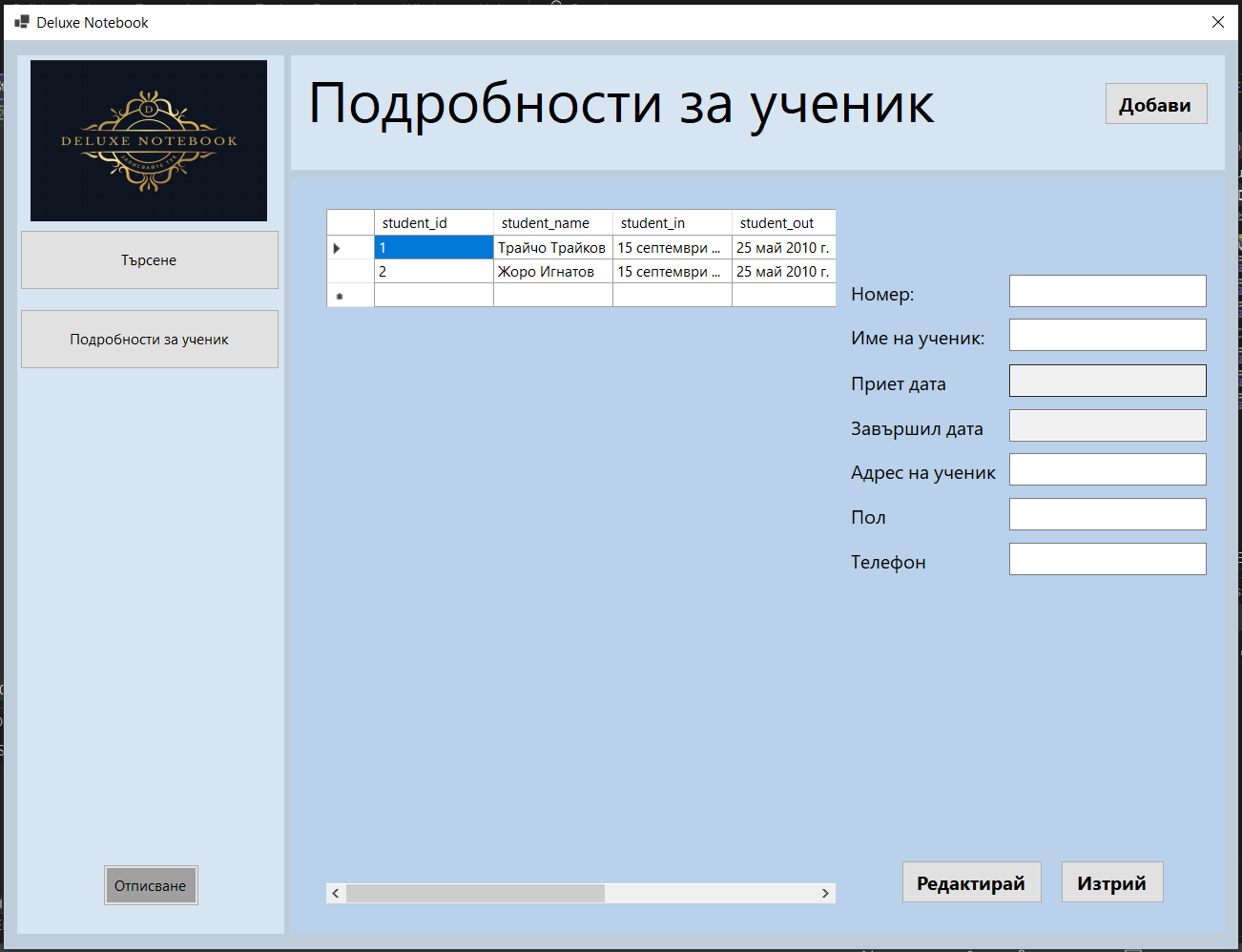
**Търсене на запис, без администраторски права**



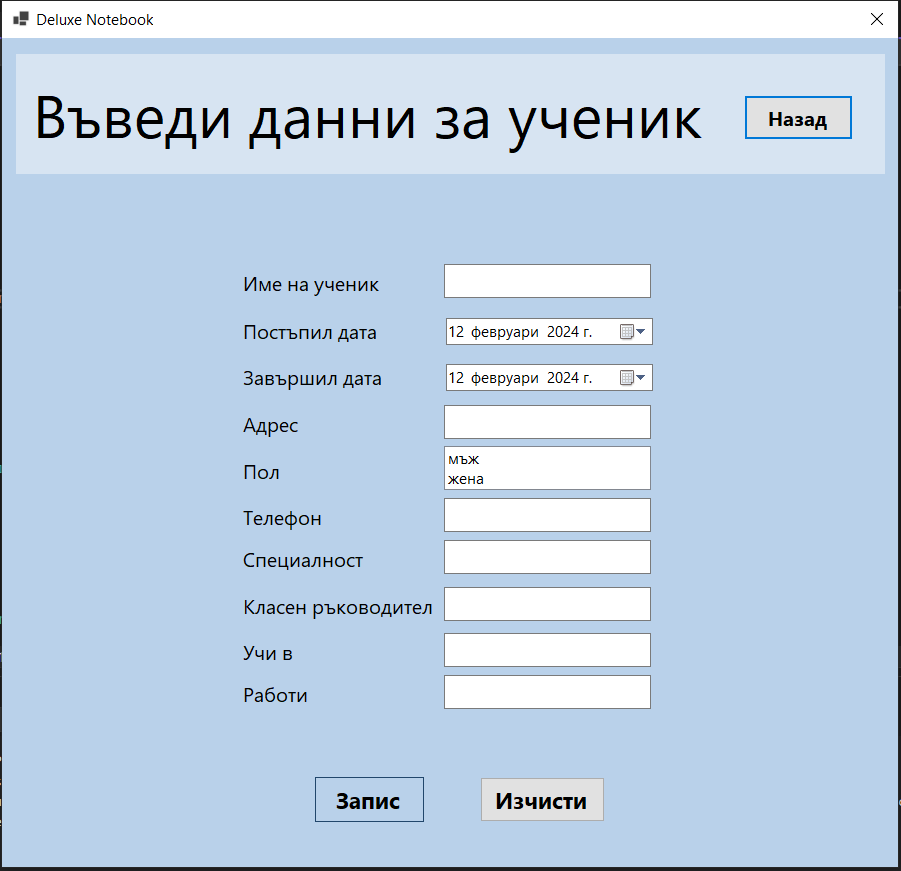
**Търсене, с администраторски права**



**Подробности за ученик**



**Добави запис**



**Диаграма на базата данни**

