**ДИПЛОМЕН ПРОЕКТ**

ЗА ПРИДОБИВАНЕ НА ТРЕТА СТЕПЕН НА ПРОФЕСИОНАЛНА КВАЛИФИКАЦИЯ

специалност „Приложно програмиране“

професия „Приложен програмист“

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

ДИПЛОМАНТ: Ивайло Георгиев

РЪКОВОДИТЕЛ-КОНСУЛТАНТ: Йорданка Ценова

Клас: 12в

ВИДИН, 2024 година

Съдържание

[УВОД 2](#_Toc156397888)

[ОСНОВНА ЧАСТ 3](#_Toc156397889)

[ТЕХНОЛОГИЯ ЗА РАЗРАБОТКА НА ПРИЛОЖЕНИЕ С ГРАФИЧЕН ПОТРЕБИТЕЛСКИ ИНТЕРФЕЙС С DАTABASE FIRST ПОДХОДА 3](#_Toc156397890)

[1. Дейности по съхраняване на документация и информация за завършили ученици в училище 3](#_Toc156397891)

[2. Същност на GUI 5](#_Toc156397892)

[3. Минимални хардуерни изисквания към компютъра за инсталация на Visual Studio 2022 5](#_Toc156397893)

[4. Модел на приложение, MVC модел и неговата същност 5](#_Toc156397894)

[5. Същност на DataBase first подхода при разработка на приложение 5](#_Toc156397895)

[6. Възможности за създаване на база данни с MySQL Workbench 6](#_Toc156397896)

[РАЗРАБОТКА НА ПРИЛОЖЕНИЕ С ГРАФИЧЕН ПОТРЕБИТЕЛСКИ ИНТЕРФЕЙС ЗА ЗАВЪРШИЛИ УЧЕНИЦИ С DАTABASE FIRST ПОДХОДА 6](#_Toc156397897)

[7. Анализ на дейностите по съхранение на ддокументация и информация за завършили ученици 6](#_Toc156397898)

# УВОД

Този дипломен проект е свързан с разработката на приложение, което съхранява и предоставя информация за бивши възпитаници на училище. 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. Липсва информация за реализацията и развитието на вече завършили ученици. 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 училищата, като по този начин даде възможност за връзка с тях.

Използваните технологии за реализацията на проекта са език за програмиране C#, база данни Microsoft ,SQL ,Visual Studio 2019 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.

Една от трудностите при използването на приложението е актуализиране на данните на бившите възпитаници. Това би изисквало периодичен контакт с бившите класни ръководители, което представлява трудност при промяна на информацията, предоставена за контакт от бившите ученици. Друга трудност е, че е необходима инсталация на приложението на персонален компютър, а не е достъпно в мрежата. Тъй като база данни за бивши ученици не е задължителен нормативен елемент за всички училища, то няма необходимост от това, базата данни да се поддържа онлайн.

# ОСНОВНА ЧАСТ

**Глава първа**

# ТЕХНОЛОГИЯ ЗА РАЗРАБОТКА НА ПРИЛОЖЕНИЕ С ГРАФИЧЕН ПОТРЕБИТЕЛСКИ ИНТЕРФЕЙС С DАTABASE FIRST ПОДХОДА

## Дейности по съхраняване на документация и информация за завършили ученици в училище

**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. Актуализация на данните е важен процес във всяка област, който се извършва с цел поддържане на точност и актуалност на информацията. Този процес се извършва често в различни контексти като бизнес, наука, образование и много други. Актуализацията на данните е важна защото информацията има склонност да се променя и остарява с времето. Например, в бизнеса актуализацията на данните може да включва промени в контактна информация на клиенти, актуализации на продуктови каталози или промени в законодателството, които могат да засегнат дейността на компанията.

Процесът на актуализация на данните включва събиране, проверка и обновяване на информацията. Това може да бъде постигнато по различни начини, включително чрез автоматизирани системи за управление на данни, използване на софтуерни приложения за обработка на информация или дори чрез ръчно въвеждане на данни от човешки оператори.

Една от основните ползи от актуализация на данните е подобрената точност и надеждност на информацията. Когато данните са актуални, решенията, базирани на тях, са по-точни и ефективни. В бизнес среда това може да доведе до по-добро вземане на решения, по-добър клиентски сервиз и по-голяма конкурентоспособност.

Освен това, актуализацията на данните може да помогне за спазване на регулаторните изисквания. Много индустрии са подложени на строги правила за съхранение и управление на данни, и актуализацията на информацията е често изискване за спазване на тези правила.

Накрая, актуализацията на данните е важна за защита от измами и злоупотреби. Актуалната информация помага да се открият и предотвратят нежелани дейности като измами, кражби на идентичност и други видове злоупотреби.

В заключение, актуализацията на данните е важен процес, който осигурява актуалност, точност и надеждност на информацията. Тя е от съществено значение за успешното функциониране на бизнеса, за спазване на регулаторните изисквания и за защита от различни видове измами и злоупотреби.

## Същност на GUI-Графичен потребителски интерфейс е разновидност в който елементите, предоставени на потребителя за управление, са изпълнени във вид на графични изображения (менюта, бутони, списъци и др.).

Разликата от интерфейса с команден ред, , в ГПИ потребителят има достъп по свой избор до всички видими обекти на екрана на монитора  (те се наричат елементи на интерфейса) с помощта на периферно устройство (клавиатура,мишка и т.н.), и осъществява непосредственото им манипулиране. Най-често елементите на графичния интерфейс се реализират като икони,които подсказват тяхното предназначение и свойства, което помага на потребителите да разберат и усвоят програмите.

…….

## Минимални хардуерни изисквания към компютъра за инсталация на Visual Studio 2019

## -Windows операционна система

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 се0 поддържа от версия [2019 Volume 1 release](https://help.syncfusion.com/windowsforms/release-notes) (17.1.0.32).

**Системни изисквания за инсталиране на 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.

## Модел на приложение, MVC и неговата същност

„Model-view-controller“ (за по-кратко „MVC”) е архитектурен шаблон, който най- често се използва при създаването на потребителски интерфейс. Той „разделя“ приложението на три взаимосвързани части. Това е направено с цел да се раздели вътрешното представяне на информация от начините по които информацията се представя на и приема от потребителя. MVC шаблонът разделя тези главни компоненти, което позволява на разработчиците да използват отново вече написан код по-ефективно, а също така позволява и паралелна разработка. Традиционно MVC шаблонът се използва за графични потребителски интерфейси (*Graphical User Interface – GUI*), но вече той се използва както и за уеб приложения, така и за мобилни такива, а дори и за други типове клиенти. Тъй като MVC разделя основните компоненти на приложението, това позволява на разработчиците да работят паралелно по различни компоненти, без да оказват влияние или да си пречат един на друг. Например един дев екип може се раздели на две части, като едната поема изграждането на back-end логиката, а другата – на front-end частта. Back-end разработчиците проектират структурата на данните и как потребителят взаимодейства с тях, без да се нуждаят от потребителски интерфейс (front-end), за да тестват и усъвършенстват тези аспекти на приложението. Обратно – front-end разработчиците могат да проектират и тестват изгледа на приложението, преди да имат готова структура от данни, с която той да борави.

## Същност на DataBase first подхода при разработка на приложение

Представлява колекция от логически свързани данни в конкретна предметна област, които са структурирани по определен начин. В първоначалния смисъл на понятието, използван в компютърната индустрия, базата от данни се състои от записи, подредени систематично, така че компютърна програма да може да извлича информация по зададени критерии. Например БД може да се използват в моделирането на хотелските системи, за да се проверява дали има налични свободни стаи в даден хотел.

Подходът Database First предоставя алтернатива на подходите Code First и Model First към Entity Data Model и създава моделни кодове (класове, свойства, DbContext.) от базата данни в проекта и тези класове стават връзка между базата данни и контролер.

Кога да използваме подхода Database-First на Entity Framework?

1. Подходът Entity Framework Database First се използва, когато вече имате база данни; тогава Entity Framework ще изпълни своето задължение и ще създаде необходимите обекти вместо вас.
2. Ако вече сте проектирали базата данни и ако не искате да полагате допълнителни усилия, тогава можете да използвате Database First Approach на Entity Framework.
3. Ако искате да модифицирате базата данни ръчно и искате да актуализирате модела от база данни, тогава можете да използвате този подход.

A screenshot of a computer

Description automatically generatedТаблиците и колоните изглеждат по този начин:

За да се получи този резултат, трябва да бъде изградена SQL заявка, която може да се изпълни:

CREATE Database Movies

GO

USE Movies2

CREATE TABLE [dbo].[Movies]

(

[Id] INT NOT NULL PRIMARY KEY IDENTITY,

[Title] VARCHAR(100) NULL,

[Rating] NCHAR(10) NULL

)

CREATE TABLE [dbo].[Actors]

(

[Id] INT NOT NULL PRIMARY KEY IDENTITY,

[Name] VARCHAR(500) NULL,

[Age] INT NULL,

[Gender] INT NULL

)

CREATE TABLE [dbo].[ActorsMovies]

(

[Id] INT NOT NULL PRIMARY KEY IDENTITY,

[ActorId] INT NULL,

[MovieId] INT NULL

)

ALTER TABLE ActorsMovies

ADD CONSTRAINT FK\_ActorId\_Actors\_Id FOREIGN KEY (ActorId)

REFERENCES Actors (Id)

ON DELETE CASCADE

ON UPDATE CASCADE;

ALTER TABLE ActorsMovies

ADD CONSTRAINT FK\_MovieId\_Movies\_Id FOREIGN KEY (MovieId)

REFERENCES Movies (Id)

ON DELETE CASCADE

ON UPDATE CASCADE;

DECLARE @actorId int

DECLARE @movieId int

INSERT INTO Movies (Title, Rating) VALUES ('Shrek', 5)

SET @movieId = @@IDENTITY

INSERT INTO Actors (Name, Age, Gender) VALUES ('Cameron Diaz', 50, 1)

SET @actorId = @@IDENTITY

INSERT INTO ActorsMovies (ActorId, MovieId) VALUES (@actorId, @movieId);

INSERT INTO Actors (Name, Age, Gender) VALUES ('Mike Myers', 59, 0)

SET @actorId = @@IDENTITY

INSERT INTO ActorsMovies (ActorId, MovieId) VALUES (@actorId, @movieId);

INSERT INTO Movies (Title, Rating) VALUES ('Inceptiopn', 1)

SET @movieId = @@IDENTITY

INSERT INTO Actors (Name, Age, Gender) VALUES ('Leonardo DiCaprio', 48, 0)

SET @actorId = @@IDENTITY

INSERT INTO ActorsMovies (ActorId, MovieId) VALUES (@actorId, @movieId);

INSERT INTO Movies (Title, Rating) VALUES ('The Matrix', 4)

SET @movieId = @@IDENTITY

INSERT INTO Actors (Name, Age, Gender) VALUES ('Keanu Reeves', 58, 0)

SET @actorId = @@IDENTITY

INSERT INTO ActorsMovies (ActorId, MovieId) VALUES (@actorId, @movieId);

INSERT INTO Movies (Title, Rating) VALUES ('The Muppets', 5)

SET @movieId = @@IDENTITY

INSERT INTO Actors (Name, Age, Gender) VALUES ('Kermit the Frog', 67, 0)

SET @actorId = @@IDENTITY

INSERT INTO ActorsMovies (ActorId, MovieId) VALUES (@actorId, @movieId);

Тази база данни ще действа, като е база данни, от която трябва да се внедри в проекта, чрез Entity Framework. Създавам конзолно приложение, но не е необходим код. Entity Framework ще генерира кода.

За да направим продължи проекта ще трябва да се инсталират няколко неща:

1. Package Manager конзола във Visual Studio 2022.
2. Текстов низ за свързване към базата данни
3. Празен проект.
4. Пакетът NuGet Microsoft.EntityFrameworkCore.Tools.

Когато проектът е готов, е време да се получи информацията от базата данни и да се създаде C# кода, с програмната логика за обработка на данните. Този процес наричаме скеле (scaffolding).

За да се създаде скеле на базата данни, ще ви трябва командата Scaffold-DbContext. Тя има няколко параметъра, като низ за свързване към базата данни, среда на базата данни (Mysql, SQL Server, PostgreSql…) и изходна директория, където Entity Framework да постави C# файловете.

Командата ще изглежда по следния начин:

Scaffold-DbContext [-Connection] [-Provider] [-OutputDir]

## Възможности за създаване на база данни с MySQL Workbench

MySQL е система за управление на релационни бази данни (Relational Database Management System) с отворен код, разработена от Oracle Corporation който е глобална корпорация, която разработва и продава компютърни софтуерни приложения за бизнеса и Sun Microsystems който е бивш американски производител на компютърни работни станции, сървъри и софтуер, която използва език за структурирани заявки (Structured Query language) за взаимодействие с бази данни. MySQL се използва за съхраняване, извличане, манипулиране и обработка на данни, които са под формата на таблици.

Днес в света на технологиите съществуват различни системи за управление на релационни бази данни, като Microsoft SQL Server, Microsoft Access, Oracle, DB2.

Това са някои причини, поради които хората използват MySQL пред други системи за управление на бази данни:

* Множество машини за съхранение:

MySQL се придържа към множество машини за съхранение и всяка една от машините за съхранение притежава характеристики.

* Висока производителност:

MySQL отчита висока производителност в сравнение с други системи за управление на бази данни поради опростения си дизайн и придържането към множество машини за съхранение.

MySQL Workbench е междуплатформен инструмент за проектиране на релационни бази данни с отворен код, който добавя функционалност и улеснява работата по разработката на MySQL и SQL. MySQL Workbench предоставя моделиране на данни, разработка на SQL и различни административни инструменти за конфигуриране. Той също така предлага графичен интерфейс за работа с базите данни по структуриран начин.

Може да се създават графичен модел с помощта на MySQL Workbench

MySQL Workbench осигурява обратно инженерство за бази данни към модели

MySQL Workbench предлага преден инженерен модел към скрипт/жива база данни

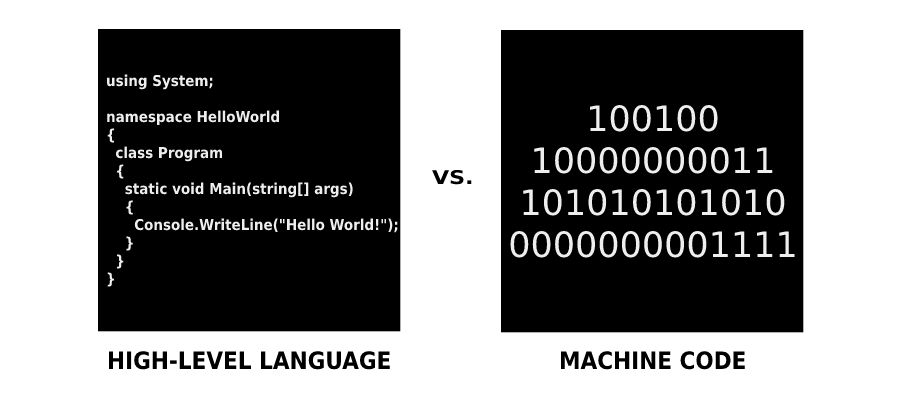
…….

## Глава втора

# РАЗРАБОТКА НА ПРИЛОЖЕНИЕ С ГРАФИЧЕН ПОТРЕБИТЕЛСКИ ИНТЕРФЕЙС ЗА ЗАВЪРШИЛИ УЧЕНИЦИ С DАTABASE FIRST ПОДХОДА

## Анализ на дейностите по съхранение на документация и информация за завършили ученици

**Какво представляват WinForms приложенията?**

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.

**ПРОЕКТНА ЧАСТ**

Ф**ункционалност 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.

Формата за създаване на нов потребител, включва регистрация на имейл и парола. Двете полета са от тип текстов низ (string). Ако потребителят е тип администратор, тоест има права за добавяне на нови записи и редактиране на текущи, може да попълва следните полета:

* + Три имена на бивш ученик – тип текстов низ (string)
  + Актуален адрес – – тип текстов низ (string)
  + Телефон - – тип текстов низ (string), с ограничение в дьлжината
  + Специалност, в която ученика се е обучавал– тип текстов низ (string)
  + Бивш класен ръководител на ученика– тип текстов низ (string)
  + Година на завършване – цяло число (int)
  + Дали работи и ако да – – тип текстов низ (string)
  + Дали учи и ако да – – тип текстов низ (string)
  + Имейл адрес - – тип текстов низ (string)

**При въвеждане на данните** и натиска на бутона за запис, контролера на приложението проверява дали данните са въведени правилно, спрямо типа данни и поставените ограничения. Ако има невалидни данни, се извежда съобщение за грешка. Ако няма невалидни данни, се извършва заявка към базата данни за запис на данните. В нея, също може да има проверка за валидността на данните, чрез поставени ограничения.

**При редакция**, логиката е аналогична – ако данните, които се въвеждат като редакция са коректни, операцията ще бъде осъществена, иначе ще бъде изведена грешка.

**Зареждането на данни** за визуализация на потребителя, се случва еднакво, без значение дали той е от тип администратор или обикновен потребител, но само администраторите имат бутон за редакция.

**Регистрация на нов потребител** – изисква въвеждане на имейл адрес и парола. При натискане на бутон за регистрация, се извършва проверка на въведените данни и дали вече е регистриран вече такъв потребител. При наличие на такъв запис, се извежда съобщение за грешка. В противен случай, се извършва запис.

Смяна на забравена парола

**ЗАКЛЮЧЕНИЕ**

Разработката на текущият проект изискваше проучване на нуждите за съхранение на информация за бивши ученици, завършили в дадена гимназия. След процеса по проучване на данни за съхранение, следваше и подбор на подходящи технологии, за реализирането на поставените задачи. Полученото Winforms настолно приложение, изпълнява функциите по съхранение и обработка на нужните данни. Постигната е целта за ефективно поддържане на база данни, с информация за бивши възпитаници на гимназията. Става ясно, че някои от процесите по разработка на подобен род приложение, могат да бъдат оптимизирани. Предизвикателствата, които се появяваха в различните етапи на разработка на приложението, показаха, че методът „проба-грешка“, не е времево ефективен и по-доброто предварително планиране на необходимите инструменти е по-добро решение.

Силните страни на проекта са, че сорс кода е сравнително лесен за разбиране, с възможност за разширения и подобрения при нужда. Освен това приложението е леко, не натоварва компютъра твърде много и практически може да бъде използвано на всеки Windows компютър от последното десетилетие и повече. Интерфейсът е прост и разбираем.

Слаби страни са фактът, че приложението работи само на Windows базирани компютри – не е подходящо за Linux, Mac OS или мобилни устройства с Android и iOS. Също така, има какво да се желае откъм външният вид по отношение на стилистиката.

Някои части от проекта, остават с възможности за подобрение – например по-добра система за идентификация на потребителя, като двуфакторна аутентикация. Подобрения могат да бъдат направени в бъдеще и по отношение на типа данни и тяхната валидация, така да се избегнат двусмислици и невалидни данни. Графичният интерфейс, може да се възползва от някоя от готовите разширения, за да изглежда по-съвременен и привлекателен за потребителя, като например **Bunifu Framework**.

# ИЗПОЛЗВАНА ЛИТЕРАТУРА

Britanica Money . (н.д.). *Oracle Corporation*. Извлечено от https://www.britannica.com/topic/Oracle-Corporation

Britanica Money. (н.д.). *Sun Microsystems*. Извлечено от https://www.britannica.com/topic/Sun-Microsystems-Inc

CODE PROJECT . (н.д.). *Първо база данни с Entity Framework*. Извлечено от https://www.codeproject.com/Articles/5354627/Database-First-with-Entity-Framework

GeeksForGeeks. (н.д.). *Въведение в C# Windows Forms приложения*. Извлечено от https://www.geeksforgeeks.org/introduction-to-c-sharp-windows-forms-applications/

Hjalager, A.-M. (2015 ). 100 Innovations That Transformed Tourism . *Journal of Travel Research*, 3-21.

NAKOU.com. (н.д.). *ORM технологии за лесен достъп до бази данни с много малко писане на код*. Извлечено от https://nakov.com/blog/2015/10/17/ruseconf-2015-orm-technologies-entity-framework/

Progress Telerik. (н.д.). *.NET Basics: ORM (Object Relational Mapping)*. Извлечено от https://www.telerik.com/blogs/dotnet-basics-orm-object-relational-mapping

simpli learn. (н.д.). *Пълно ръководство за MySQL Workbench*. Извлечено от https://www.simplilearn.com/tutorials/mysql-tutorial/mysql-workbench

Syncfusion. (н.д.). *Системни изисквания за Windows Forms*. Извлечено от https://help.syncfusion.com/windowsforms/installation-and-upgrade/system-requirements