

Съдържание

I. Увод

- 1. Цел на дипломната работа6
- 2. Структура на дипломната работа7

II. Проучване

- 2.1. Използвани технологии и инструменти8
- 2.2. Създаване на менюта във форма на приложение 10

III. Проектиране

- 3.1. Модел на случаи на употреба (Use Case)10
- 3.2. Случай на употреба - Валидиране на парола (Validate password).....11
- 3.3. Случай на употреба - Обновяване на данни (Update data).....12
- 3.4. Случай на употреба - Преглед на записи съхранени в апарата
(Review offline saved data).....12
- 3.5. Случай на употреба - Изтриване на запис(Delete data).....13
- 3.6. Случай на употреба - Offline работа с приложението (Offline).....13
- 3.7. Случай на употреба - Online работа с приложението (Online).....13
- 3.8. Случай на употреба - Въвеждане на данни (Enter data).....13
- 3.9. Модел на случаите на употреба на приложението14

IV. Реализация

- 4. 1.Реализация на offline режима15
- 4.2. Реализация на online режима16
- 4.3 User Kontrol21
- 4.4. Динамична реализация23
- 4.5. Структура на приложението25
- 4.6. Имплементация26
- 4.7. Реализация на базата данни33

V. Заключение36

VI. Използвана литература37

I Увод

С разрастването на бизнеса и увеличаването на конкуренцията настъпва повишена нужда от софтуерни решения, които да автоматизират и оптимизират бизнес процесите във всяка компания. Тези решения често включват различни интегрирани компоненти, които обменят информация и функционират като единен механизъм. Ключова характеристика за подобни софтуерни платформи е възможността за централизиран достъп до информация по всяко време и от различни устройства. Платформата .NET, разработена от Microsoft, предоставя среда, в която различни технологии могат да се интегрират, създавайки компоненти за информационни системи, които значително улесняват бизнес процесите във фирмата. В представения дипломен проект ще бъде разгледано приложението на няколко технологии от .NET платформата при създаването на информационна система, предназначена за удовлетворяване на бизнес нуждите на фирма, занимаваща се с планински туризъм.

1. Цел на дипломната работа

Целта на настоящата дипломна работа е да създаде информационна система, която да координира, регистрира и облекчи избора на различни дейности, свързани с възможностите на потребителите във фирма "NightClubGK". Фирмата "NightClubGK" е водеща сред нощните клубове, отговаряща на високи стандарти в своята област и известна с престиж, професионализъм и качество.

Основният мотив за изграждането на информационната система е да се подобри регистрирането и отчитането на дейностите на потребителите, които досега са били обработвани ръчно, използвайки хартиени документи, таблици на Microsoft Excel. Този процес е бил несъвършен и е необходимо да се преодолее със създаването на релационна база данни, като се използва Microsoft Access за съхранение на информацията. Информационната система трябва да предостави две клиентски приложения, насочени към двата основни типа потребители - Администратори и Техници. Достъпът до базата данни от тези приложения трябва да се осъществява чрез XML Web Service.

2. Структура на дипломната работа

В дипломната работа ще бъдат разгледани базата данни на системата и двете клиентски приложения, тъй като са реализирани основно от дипломанта. Дипломната работа се състои от увод, три части, заключение, списък с използвана литература и приложения.

Увод – запознаване с темата и целите на дипломната работа.

Основна част

Първа глава - Проучване – Представява проучвателната част на дипломен проект. Прави се преглед на съществуващи подобни програмни системи и продукти и преглед на известните развойни средства и среди

Втора глава - Проектиране – Описание на изискванията към програмния продукт (SRS, use cases), описание на избраната технология и софтуерните средства, потребителски интерфейс (менюта, екрани, прозорци)

Трета глава - Реализация – Същинската част на дипломния проект, която е с най-голям обем. Да включва описание на начина на реализация на алгоритмите, фрагменти от сорс кода със съответни коментarii, структура на базата данни (E/R diagram)

Заключение - включва: обобщение на постиженията в дипломната работа; тенденции за усъвършенстване и обогатяване на разработката; възможностите за неговото приложение

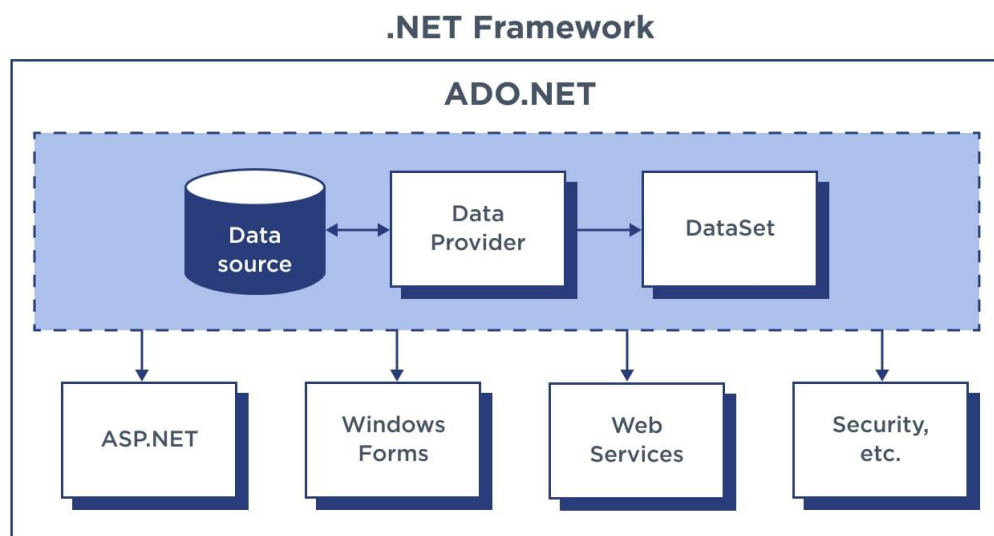
Използвана литература - представя списък на използваните в процеса на разработка на системата литературни и електронни източници.

II. Проучване

2.1. Използвани технологии и инструменти

Microsoft .NET Framework

Microsoft .NET Framework в десктоп приложенията, разработвани с C#, предоставя обширна среда и инструменти, които значително облекчават процеса на създаване на софтуер с богат функционален и графичен потребителски интерфейс. С използването на технологии като Windows Presentation Foundation (WPF) и Windows Forms, разработчиците могат лесно да създават интерактивни и стилкови приложения, които отговарят на нуждите на потребителите.



Jelvix

jelvix.com

фиг. 1 Архитектура на .NET Framework

.NET Framework също така предоставя средства за лесно взаимодействие с бази от данни чрез ADO.NET и Entity Framework. Този инструментариум улеснява работата със заявки и манипулирането на данни, обогатявайки функционалността на десктоп приложенията с обширна обработка на информация.

Възможността за асинхронно програмиране в .NET Framework дава възможност за изпълнение на задачи, които не блокират потребителския интерфейс по време на изпълнение, подобрявайки общото потребителско изживяване.

Освен това, .NET Framework предлага вградени механизми за управление на грешки и изключения, което улеснява откриването и обработката на проблеми по време на изпълнение.

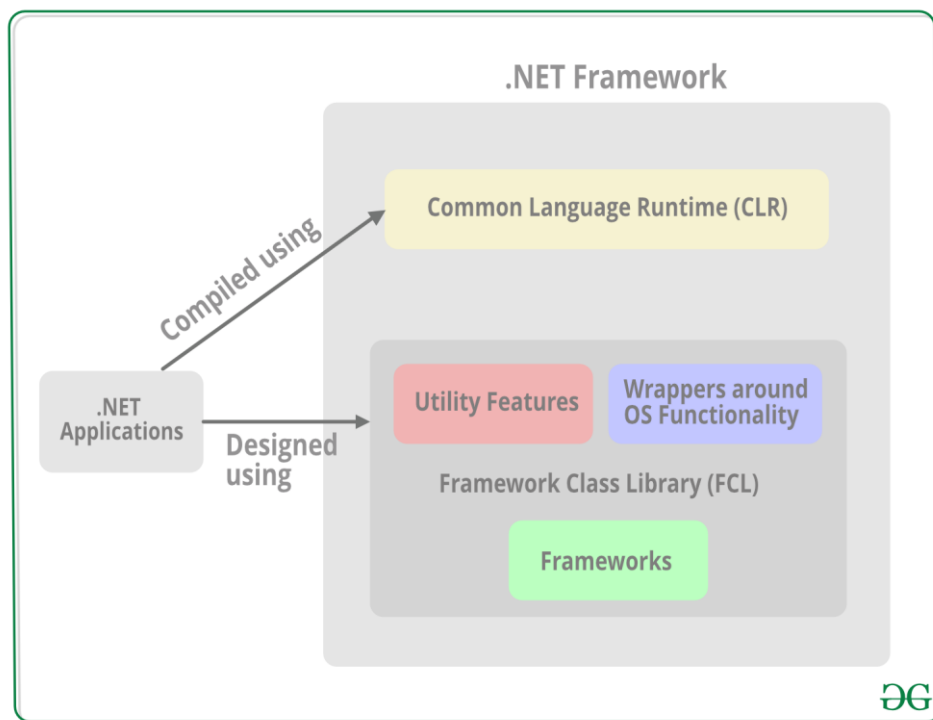
Богатите библиотеки на .NET Framework предоставят средства за обработка на различни формати на данни и файлове, както и за съхранение и защита на данни, включително механизми за шифроване и безопасност.

Накрая, възможността за интеграция с други технологии и услуги допълнително обогатява възможностите на .NET Framework, предоставяйки гъвкавост и съвместимост между различни приложения и системи. Това прави .NET Framework подходящ избор за разработка на разнообразни и сложни десктоп приложения.

Common Language Runtime (CLR) – Това е среда, която изпълнява приложенията написани на .NET.

Framework Class Library (FCL) – Представява стандартна библиотека от класове спомагаща разработването на .NET приложения. Предоставя основната функционалност за разработка: ADO.NET, XML, ASP.NET, Web Services, Windows Forms.

Архитектурата на .NET Framework е показана на фиг. 2



фиг. 2 Архитектура на Framework class library

Microsoft Access

Microsoft Access е система за управление на бази от данни (СУБД), разработена от Microsoft. Тя предоставя графичен интерфейс за създаване и управление на бази от данни, форми, отчети и заявки. Access е насочен към малки и средни предприятия, като осигурява лесен начин за работа с данни и генериране на отчети, но не е предназначен

за по-големи и сложни приложения, за които се използват по-мощни релационни бази от данни.

2.2. Създаване на менюта във форма на приложение

Менютата са важен елемент на формите в едно приложение с графичен потребителски интерфейс. Чрез тях потребителят има възможност за бърз достъп при изпълнение на избрана от него операция чрез директен избор на съответната команда от менюто на приложението. Менютата са организирани в йерархична структура. Даден елемент от меню, сам по себе си може да представлява меню, когато се използва за визуализиране на елементи от подменю.

III. Проектиране

При създаването на повечето приложения, един от първите стъпки е определянето на графичния потребителски интерфейс, който да осигури на потребителите удобство при работата с програмата. Развитието на софтуерни системи с висока сложност, което често се извършва от различни екипи специалисти, изисква прилагането на стандарти и ефективни методологии за разработка.

В случая създаването на клиентския интерфейс на системата се извърши с използването на .NET Framework библиотеката, като инструментът Windows Forms предостави възможност за бързо и лесно създаване на графичен потребителски интерфейс. Тази платформа е базирана на концепцията за Rapid Application Development (RAD), която позволява визуално създаване на приложения чрез комбиниране на готови компоненти и автоматично генериране на голяма част от програмния код. Windows Forms предлага разнообразие от класове и типове, които бяха използвани при създаването на клиентските приложения в системата.

3.1. Модел на случаи на употреба- (Use Case) модел

Моделът на случаите на употреба съдържа актьорите, случаите на употреба, както и връзките между тях. Моделът на случаите на употреба се представя с помощта на UML диаграми, които показват актьорите и случаите на употреба от различни гледни точки и с различни цели.

3.1 Случай на употреба - Идентифициране на потребител (Log in)

Потребителят се идентифицира в системата чрез парола и име след направена предварителна регистрация от негова страна в определена форма.

Предусловие: Потребителят е въвел парола в първата форма от приложението- *WelcomeForm*, но потребителят може да разглежда по два начина- свободно или чрез регистрация.

Основен поток от действия:

1. Потребителят натиска бутон "*Вход потребител*" от първата форма на приложението - „Вход в системата”;
2. Приложението валидира въведената парола.

Алтернативен поток 1:

A1 1. Потребителят е успешно идентифициран.

A1 2. При наличие на интернет връзка приложението обновява локалния XML файл. В противен случай преминава в offline режим на работа.

A1 3. Приложението показва на потребителя форма с име Потребителски профил с меню за възможни заявки за определена дестинация или екипировка”;

A1 4. Изход.

Алтернативен поток 2:

A2 1. Потребителят не е успешно идентифициран.

A2 2. Приложението показва съобщение, че е въвел грешно парола или име и дава възможност отново във форма „Вход потребител ” да се идентифицира повторно.

A2 3. Край.

3.2. Случай на употреба - Валидиране на парола (*Validate password*) Валидиране на паролата, въведена от администратора.

Предусловие: Администратора е въвел парола и е натиснал бутона "*Напред*" във формата „Вход в системата”.

Основен поток от действия:

1. Изчислява се MD5 хеш стойността на въведената парола;
2. Към получения хеш се добавя *garbage*.
3. Приложението извиква web метод на web услугата за идентификация на потребител, като подава като параметри получения хеш, DeviceID-то.

Алтернативен поток 1:

A1 1. При наличие на връзка със сървъра приложението обновява локалния XML файл, със получените данни от сървъра.

A1 2. Приложението запазва валидната хеш стойност във файла с настройки.

A1 3. Приложението показва на администратора форма „Главно меню”;

A1 4. Край.

Алтернативен поток 2:

A2 1. При липса на връзка със сървъра приложението валидира хеш стойността спрямо тази запазена във файла с настройки (т.е. Последната валидна парола) и преминава в *offline* режим на работа.

A2 2. Приложението показва на администратора форма „Главно меню”;

A2 3. Край.

3.3. Случай на употреба - Обновяване на данни (*Update data*)

Данните в локалния XML файл се обновяват със данни от централната база данни, при първоначално идентифициране на администратора.

Предусловия: Потребителя се идентифицира в системата или се е задействал *timer*-а за обновяване на данни.

Основен поток от действия:

1. Приложението взема от локалните данни минималните и максимални индекси за оператори, складове, типове машини, типове аварии и задачи.

2. Приложението извиква web метод за обновяване на данни, като подава като параметри индексите.

Алтернативен поток 1:

A1 1. При наличие на връзка със сървъра приложението получава новите данни и данните, които трябва да се изтрият(ако има такива).

A1 2. Приложението обновява базата данни в паметта и я записва в локалния XML файл.

A1 3. Край.

Алтернативен поток 2:

A2 1. При липса на връзка със сървъра приложението преминава в *offline* режим.

A2 2. Край.

3.4. Случай на употреба - Преглед на записи съхранени в апарата (*Review offline saved data*)

Потребителят може да преглежда записи за дейността на фирма Етър, съхранени в приложението, по време на липса на връзка със сървъра

Предусловия: Потребителя не е нужно да се идентифицирал в системата за да разгледа и избере дадена дестинация или екипировка придружена с артикул.

Основен поток от действия:

1. Приложението взема от локалните данни записа за първата дейност.
2. Потребителят преминава отново последователно по формите, в които е въвел информацията за дейността. Като във формите му се показват въведените или търсените от него данни.

3.5. Случай на употреба - Изтриване на запис(Delete data)

Администраторът може да изтрива записите, които са били съхранени в приложението или такъв за който в момента е въвел данни.

Предусловия: Администраторът е на форма „Главно меню” на приложението му и след изчерпване на даден артикул записва изчерпан или добавя дадена нова дестинация или изтрива дадена дестинация с бутон.

Основен поток от действия:

1. Администраторът натиска бутона "Изтрий".
2. Приложението изтрива записа от локалната базата данни в паметта.
3. Край.

3.6. Случай на употреба - Offline работа с приложението (Offline)

Потребителят има възможност да въвежда данни и запазва информация при липса на интернет връзка.

Предусловия: Приложението е в *offline* режим на работа.

Основен поток от действия:

1. Потребителят преминава последователно по формите според типа на дейността, за която въвежда данни.
2. Приложението запазва въведените данни в локален XML файл.
3. Приложението показва формата „Вход в системата”, за да може потребителя да се идентифицира online в системата.
4. Край.

3.7. Случай на употреба - Online работа с приложението (Online)

Администратора има възможност да въвежда данни и запазва информация в централна база данни при наличие на интернет връзка.

Предусловия: Администратора е успешно идентифициран и приложението е в *online* режим на работа

Основен поток от действия:

1. Администратора преминава последователно по формите според типа на дейността, за която въвежда данни.
2. Приложението валидира дали е избрана задача, ако типа дейност изисква такава.
3. Приложението валидира елемента на който се е спрял, при въвеждане на данни зажелани дейности и дейности за оставяне или вземане на определен елемент или посещение на определена дестинация.
4. Приложението извиква съответен Web метод, според типа на дейността, за запазване на данните в централната база данни.
5. Приложението показва форма („Главно меню“) за да може администратора да избере типа на следващата дейност.
6. Край.

3.8. Случай на употреба - Въвеждане на данни (Enter data)

Потребителят може да въвежда данни за определена дейност, като преминава през форми, чиято последователност зависи от указания тип на дейността. При преглед на записи, които са били съхранени в апарата, потребителят няма възможност да редактира типа на дейността, адреса на дестинацията

Предусловия: Потребителя е успешно идентифицира.

Основен поток от действия:

1. Потребителят въвежда данни.

Алтернативен поток 1:

A1 1. Потребителя натиска бутона „Напред“.

A1 2. Въведените данни се валидират.

A1 3. Приложението показва следващата форма на потребителя *Алтернативен поток 2:*

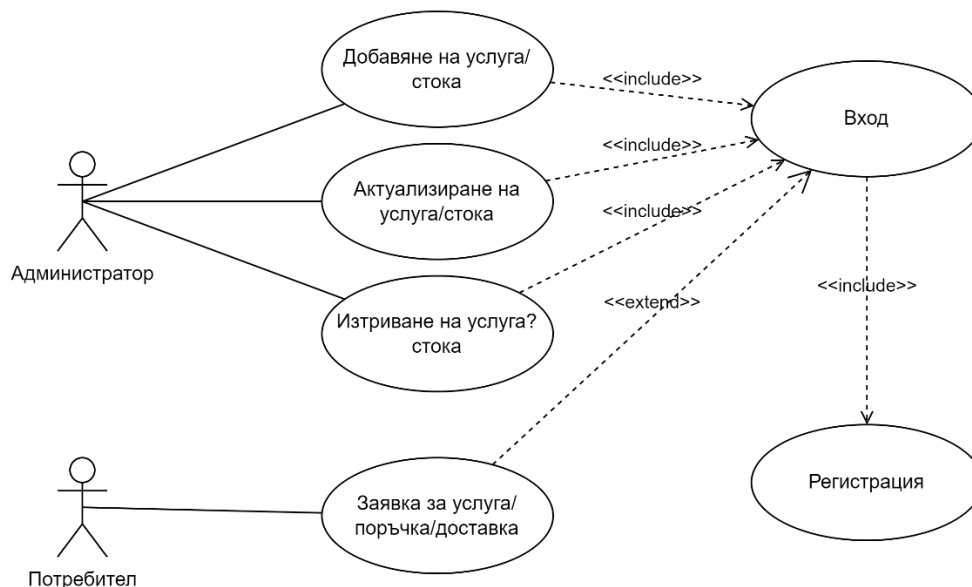
A2 1. Потребителя натиска бутона „Затвори“.

A2 2. Приложението показва предходната форма.

A2 3. Потребителя редактира вече въведените от него данни.

3.9. Модел на случаите на употреба на приложението

Определяме един актьор за приложението – Администратор, потребителят, които извършва всички взаимодействия с него.



фиг. 1 Модел на потребителските случаи – Use Case Model за приложението

IV Реализация

4.1. Реализация на offline режима

Приложението трябва да позволява на потребителя да работи и без интернет връзка. Въведените от потребителя данни трябва да се съхраняват локално в устройството и да се синхронизират със сървъра при възстановяване на интернет връзка. В раздел II са представени различни стратегии за съхранение на данни в приложения за PocketPC: релационна база данни (Microsoft SQL Server CE), локални файлове (XML файлове) и структура данни, базирана на сесии, в паметта.

В това приложение данните се съхраняват в локален XML файл, който се зарежда в паметта при стартиране на приложението. Решението за използване на XML файл за съхранение на данни се базира на няколко фактора:

- Обемът на данните, които трябва да бъдат съхранени на PocketPC, обикновено е между 50 и 100 kb. Зареждането им в паметта би било по-ефективно от достъпа до тях чрез SQL Server CE Query Engine.
- Обменът на данни между PocketPC и уеб услугата е малък (около 1Mb). Използването на XML и уеб услуга е предпочитано, а .NET Compact Framework осигурява обширна поддръжка на двете технологии.

Локалният XML файл има следната структура:

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <images>
3      <image image="ishiharal6.gif" question="text...">
4          <normal weigth="2" answer="26"/>
5          <colorblind weigth="2">
6              <protan weigth="2" answer="6"/>
7              <protan weigth="1" answer="(2)6"/>
8              <deutan weigth="2" answer="2"/>
9              <deutan weigth="1" answer="2(6)"/>
10         </colorblind>
11     </image>
12 </images>

```

Фиг. 2 XML файл

4.2. Реализация на online режима

Приложението осъществява обмен на данни със сървъра чрез асинхронни извиквания на веб услуга. При използването на асинхронни извиквания на веб услуга, потребителският интерфейс не се блокира, докато операцията не завърши. Класът WebProху представлява отдалечен пълномощник (remote proху), който е разновидност на шаблона за пълномощници и предоставя локално представяне на обект от друго адресно пространство.

Този тип пълномощник е известен като "посредник" в литературата. Класът WebProху съдържа референция към автоматично генериран клас за достъп до веб услугата.

В него са дефинирани методи, които съответстват на методите, предоставени от веб услугата. Обработката на общи изключения, включително тези, които възникват при обмен на данни със сървъра или при автентикация на потребителя, се извършва централизирано в метода GetWebCallResult.

Класът `WebProху` има вложен клас `Nested`, който съдържа статично поле `instance`, представляващо единствената инстанция на `WebProху`. Този вложен клас е създаден с цел ограничаване на достъпа до инстанцията на `WebProху` и гарантиране на съществуването ѝ само в един екземпляр (singleton pattern).

Вътрешният статичен клас `Nested` съдържа статично поле `instance`, което се инициализира при стартиране на програмата. Тъй като конструкторът на вложения клас е скрит (private), инстанцията на `WebProху` може да бъде достъпена само чрез статичното поле `instance`.

```

public class WebProxy
{
    private static readonly Lazy<WebProxy> lazyInstance = new
    Lazy<WebProxy>(() => new WebProxy());

    public static WebProxy Instance
    {
        get
        {
            return lazyInstance.Value;
        }
    }

    private WebProxy()
    {
        // Private constructor to prevent external instantiation
    }

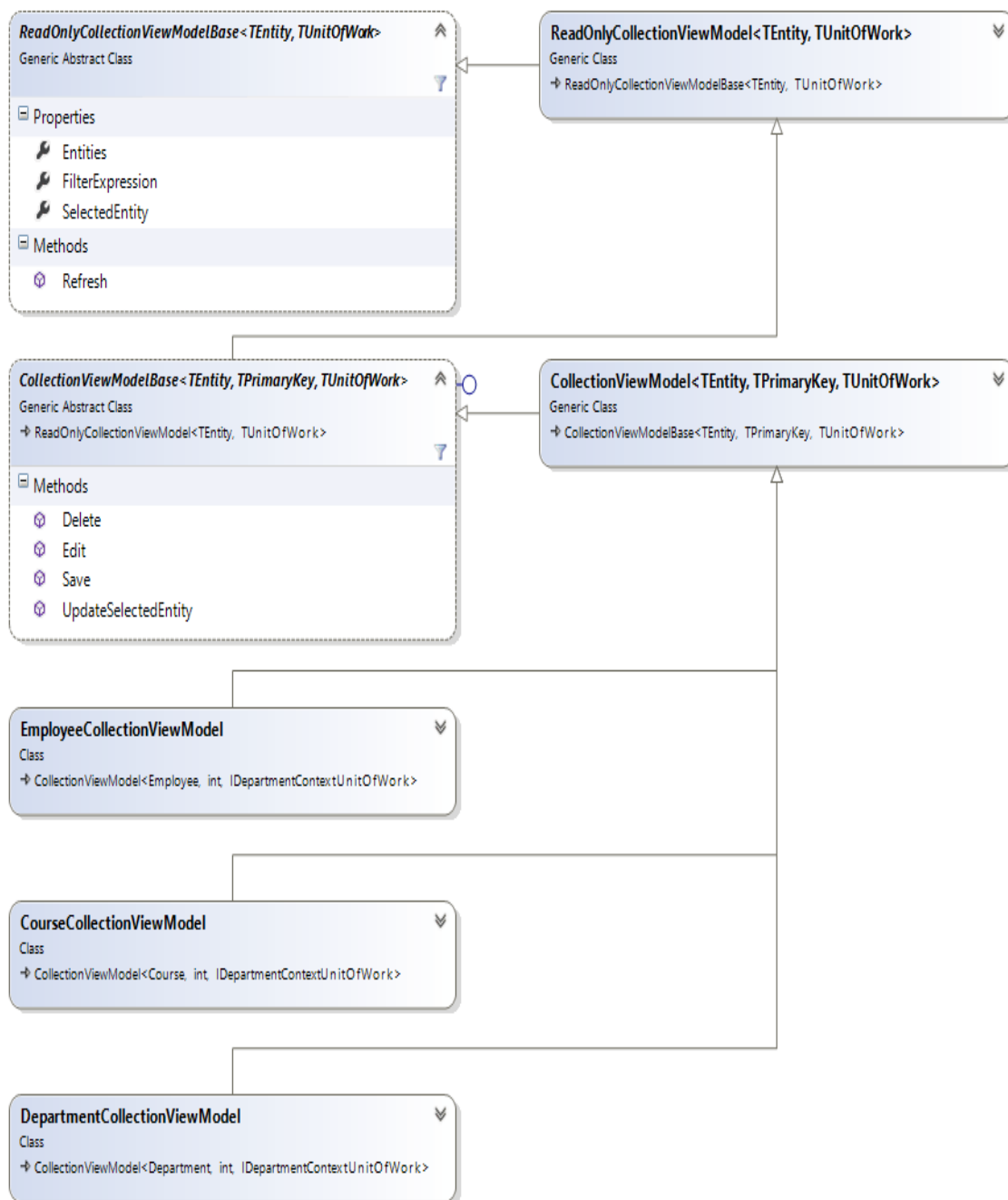
    ...
}

```

Фиг. 3 Реализация на класа WebProxy

В самия `WebProxy` клас има статично свойство `instance`, което позволява достъп до инстанцията на `WebProxy` чрез вложения клас `Nested`.

Този подход осигурява гаранция, че в рамките на програмата ще съществува само една инстанция на `WebProxy`, като същевременно се предотвратява ненужната инстанциация при създаването на множество обекти от класа.



фиг. 4 Клас диаграма на класовете от пространството на имена *CompactClient.WebActions*

За да се намали интернет трафика, данните, които се изпращат към сървъра, се компресират. За тази цел е използвана open-source библиотеката за компресиране SharpZipLib за .NET Framework.

За да може всеки уеб метод да бъде маркиран за компресия, се създава клас *CompressionSoapExtensionAttribute*, който е наследник на класа *SoapExtensionAttribute*:

```

using System;
using System.Web.Services.Protocols;

namespace WSCompress
{
    [AttributeUsage(AttributeTargets.Method)]
    public class CompressionSoapExtensionAttribute :
    SoapExtensionAttribute
    {
        private int priority;

        public override Type ExtensionType
        {
            get { return typeof(CompressionSoapExtension); }
        }

        public override int Priority
        {
            get { return priority; }
            set { priority = value; }
        }
    }
}

```

Фиг. 5 Дефиниране на клас

Свойството `ExtensionType` връща типа, който изпълнява допълнителната логика (`CompressionSoapExtension`), докато свойството `Priority` указва реда на изпълнение, когато има няколко допълнения едновременно.

Имплементацията на конкретната логика за добавяне на функционалността е клас, който е наследник на `SoapExtension` (от пространството от имена `System.Web.Services.Protocols`).

```

using System;
using System.IO;
using System.Web.Services.Protocols;

namespace WSCompress
{
    public class CompressionSoapExtension :
SoapExtension
    {
        Stream oldStream;
        Stream newStream;

        public override Stream ChainStream(Stream
stream)
        {
            oldStream = stream;
            newStream = new MemoryStream();
            return newStream;
        }

        public override object
GetInitializer(LogicalMethodInfo methodInfo,
SoapExtensionAttribute attribute)
        {
            return attribute;
        }

        public override object GetInitializer(Type
type)
        {
            return typeof(CompressionSoapExtension);
        }

        public override void Initialize(object
initializer)
        {
            CompressionSoapExtensionAttribute attribute
=
(CompressionSoapExtensionAttribute)initializer;
        }
    }
}

```

Фиг. 6 Дефиниране на методи с атрибути


```

10
11 namespace Diplomen_proekt
12 {
13     public partial class Form1 : Form
14     {
15         public static string username = "admin";
16         public static string password = "password";
17         public Form1()
18         {
19             InitializeComponent();
20         }
21
22
23
24         private void Form1_Load(object sender, EventArgs e)
25         {
26             // TODO: This line of code loads data into the 'Diplomen_proektDBDataSet3.Users' table. You can move, or
27             // remove it, as needed, by clicking the 'Load Data' button in the Data Designer.
28             mainPanel2.Hide();
29         }
30
31
32
33         private void button2_Click(object sender, EventArgs e)
34         {
35             mainPanel2.Show();
36         }
37     }
38 }

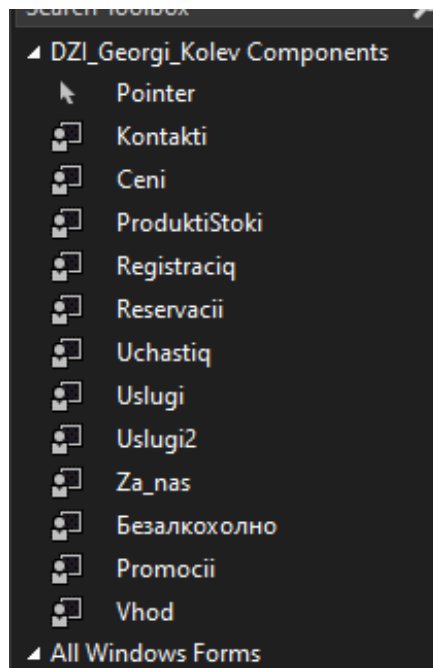
```

Фиг. 7 Деклариране на локални променливи

4.3. User контроли

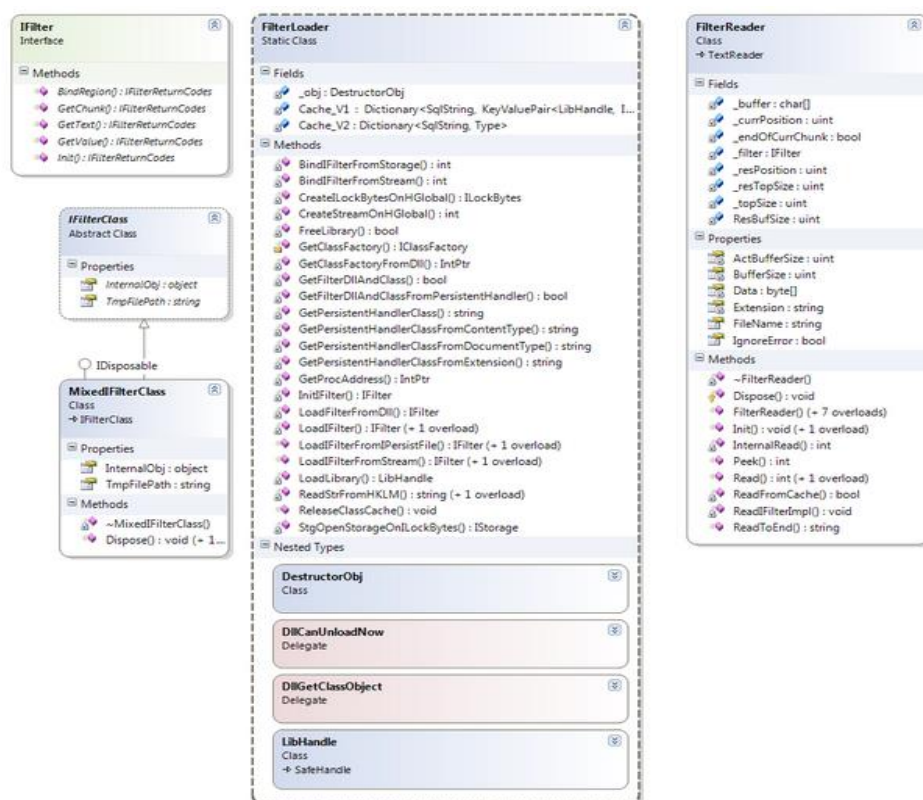
Създаден е клас, който наследява UserControl, за да се предостави функционалност за показване и избиране на задачи на няколко места.

Този контрол има възможност за зареждане на задачите от таблицата на типизиран DataSet DBMobile в TreeView контрол (дървовидна колекция от именувани обекти, всеки представен чрез TreeNode). Контролът предоставя методи за добавяне, обновяване и премахване на задача от дървото, като входът за тези методи е обект от тип TaskRow от таблицата Tasks. Чрез свойствата на контрола могат да се извлече информация за броя задачи, името на избраната задача и идентификационния номер на задачата. Освен това, TasksControl дефинира и събития, които известяват абонатите за тях за изтриване на избрана задача, промяна на броя на задачи, отказ от задача или избор на задача.



Фиг. 8 User Controls

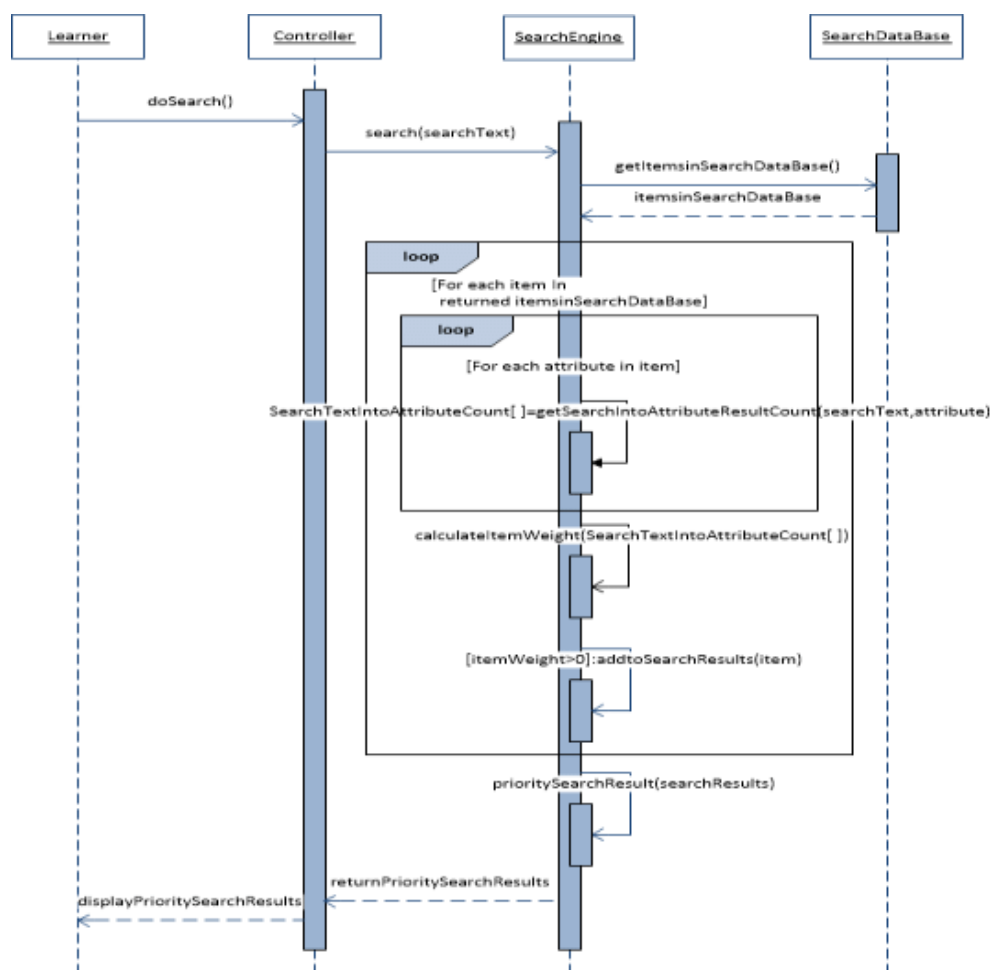
Имплементацията на класовете може да се види в кода предоставен към дипломната работа.



фиг. 9 Клас диаграма на класовете в пространството от имена
CompactClient.CompactFrameworkUtils

4.4. Динамична реализация

След като са обсъдени основните класове, в последващите диаграми на последователности ще бъдат представени обектите, които се създават, и техните взаимодействия в различни сценарии на потребителското използване.



фиг. 10 Диаграма на последователността за метода *ShowForm(IBaseForm sender)* на класа *MainForm*

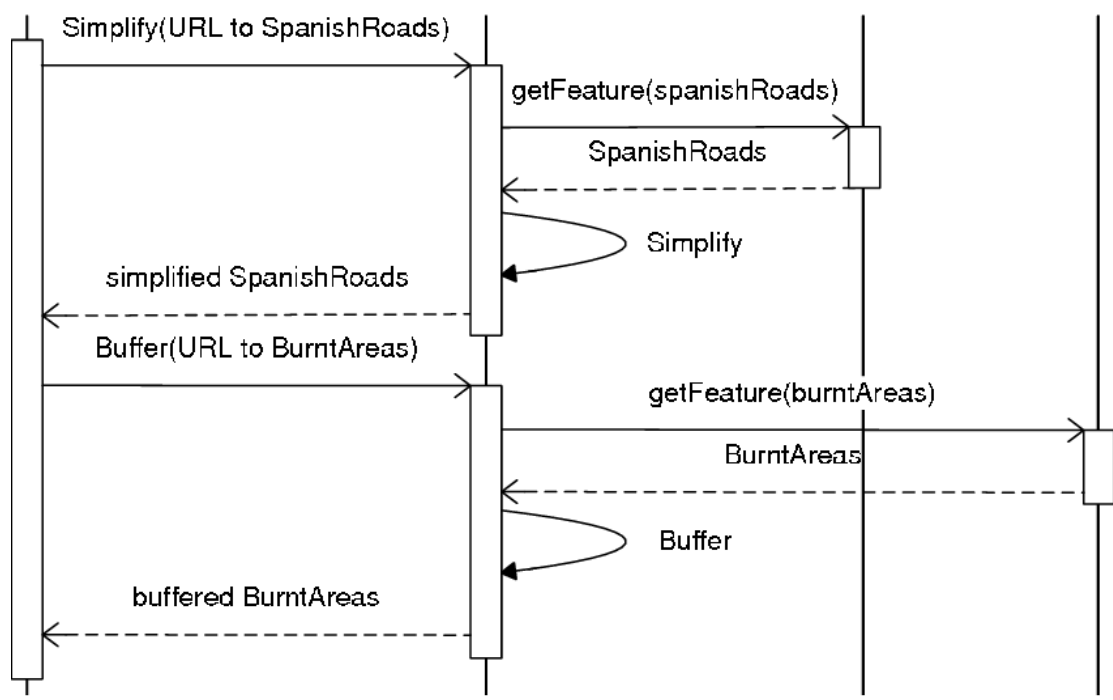
На диаграмата се показва базовата последователност от съобщения, които се изпращат между обектите при показването на формата "Главно меню". В зависимост от наличието на интернет връзка, записи съхранени в апарата и задачи, формата може да се покаже по различен начин:

- При наличие на интернет връзка и ако има записи, съхранени в апарата по време на offline режима на работа на програмата, се показва панел, предоставящ на потребителя информация за броя записи, съхранени в телефона, и обобщена информация за първия запис.

- В противен случай се проверява дали има получени задачи. Ако има такива, на потребителя се дава възможност да избере задача от контрола за задачи.

- Ако няма нито съхранени записи, които да се преглеждат, нито задачи на потребителя, се показва списък с дейности, в който трябва да укаже типа дейност, за която желае да въвежда данни.

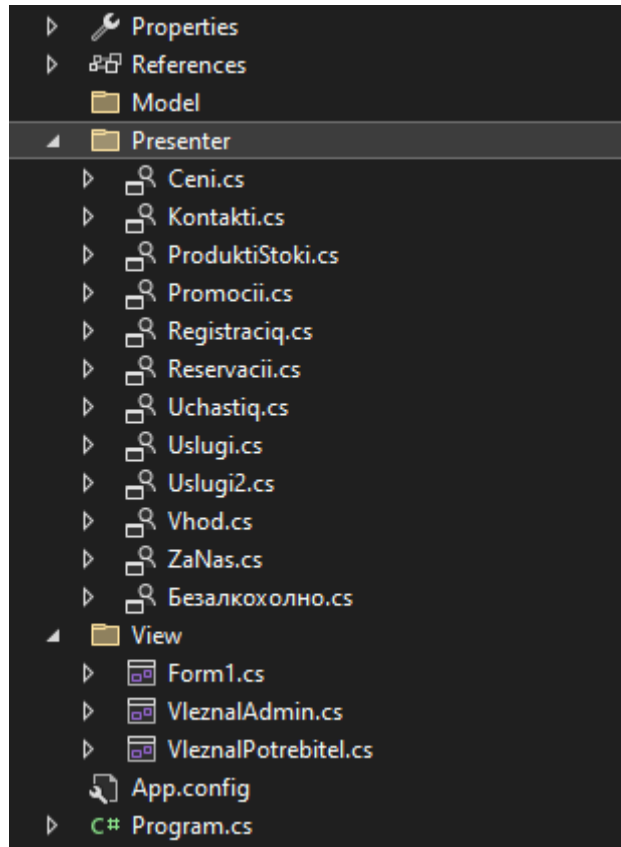
На следващата диаграма е представен процесът на стартиране на приложението. При стартиране на потребителя се показва форма, която се стартира на отделена нишка, докато се инициализира и зареди формата "Вход в системата". След приключване на инициализацията и зареждането на настройките и локалния XML файл в паметта, се извиква функция за скриване на формата.



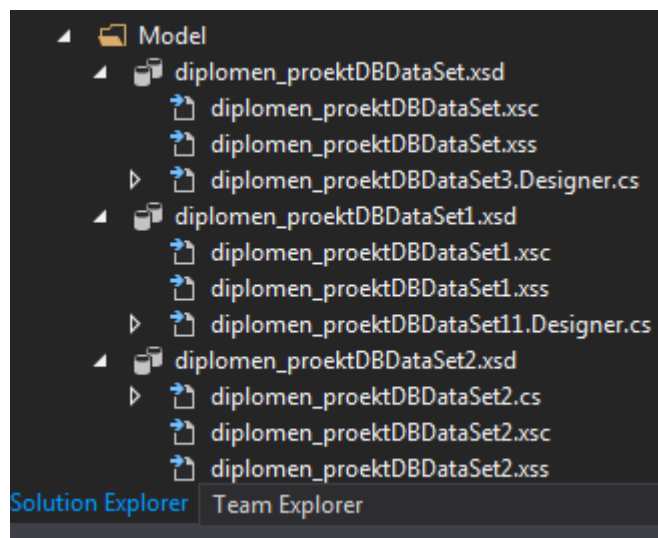
фиг. 11 Диаграма на последователностите при стартиране на приложението

4.5. Структура на приложението

При създаването на приложението първоначално създадох три папки отговарящи на името на изискването ми проекта да е MVP. В папката Model както се вижда на фигура 13 поставих базата данни, а в папката View на фиг.12 поставих класовете и моделите, а в най-важната папка Presentation поставих изгледният слой или така нареченият презентационен слой. Наименувах всеки елемент подобаващо за да ми е по-лесно при търсене и попълване на данните в класовете.



Фиг. 12



фиг:13

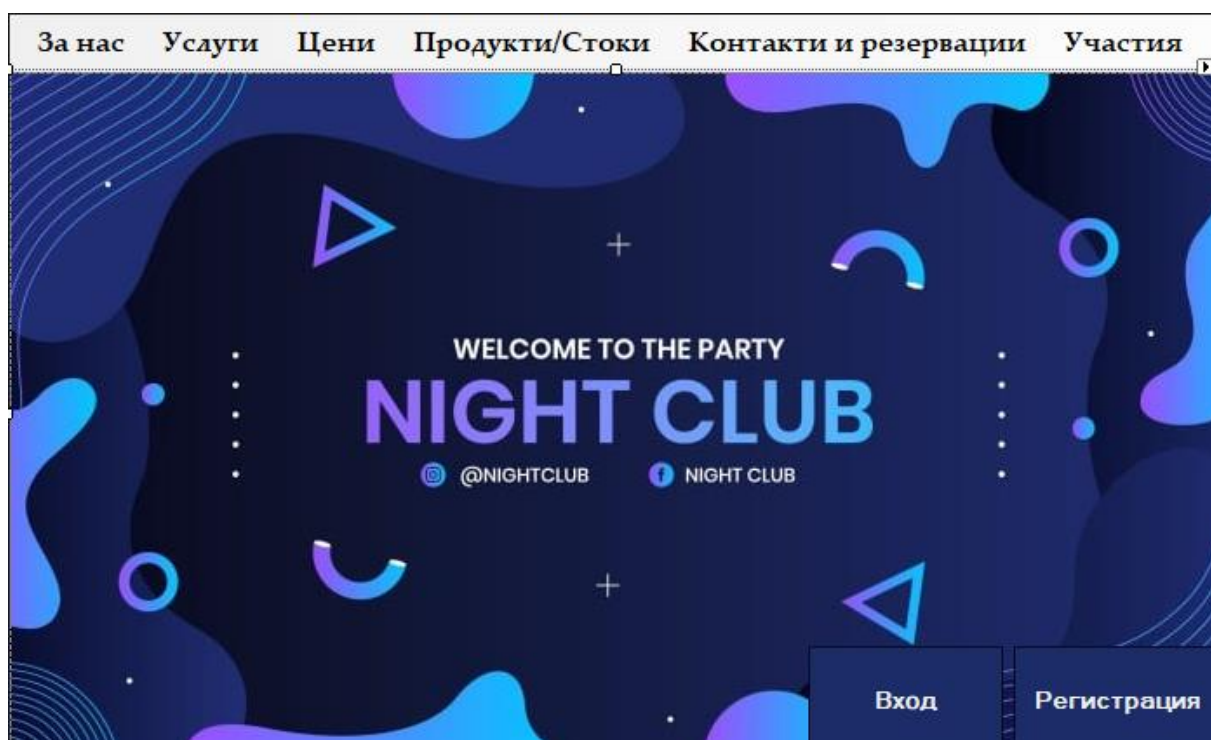
4.6 Имплементация

За реализирането на приложението е използвана платформата Microsoft Forms и езика за програмиране C#. В глава I са представени потребителските изисквания към приложението.

4.6.1. Потребителски интерфейс

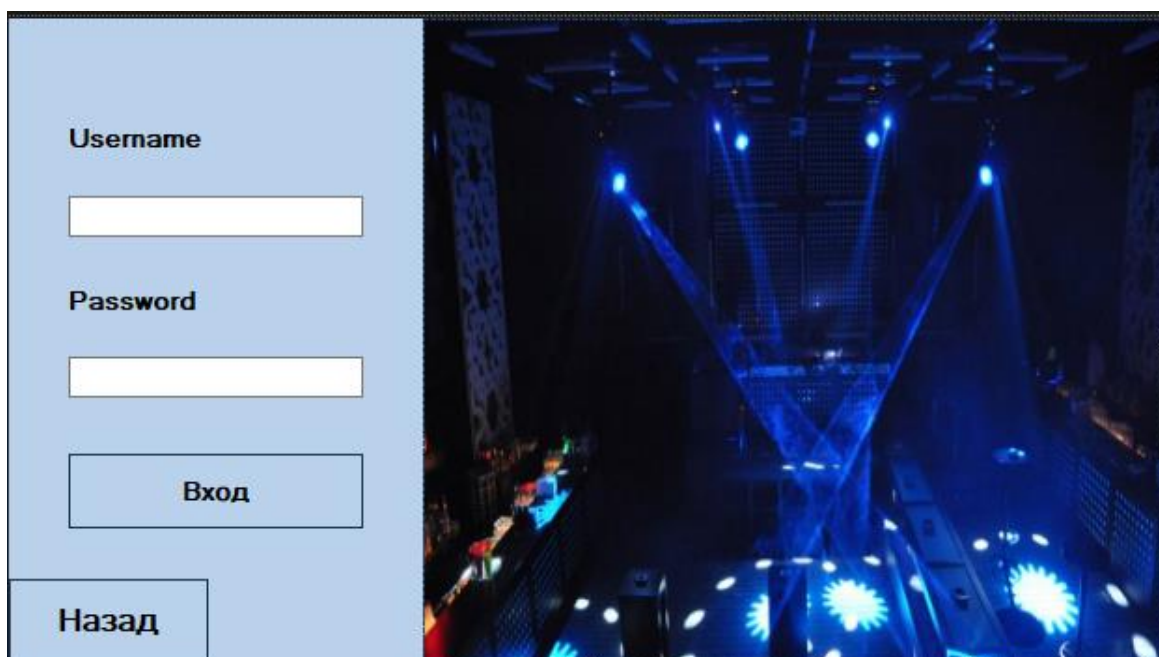
Потребителският интерфейс е изграден на базата на библиотеката Windows Forms, част от .NET framework. Интерфейса е проектиран като диалогово базирано Windows приложение, заради по-големите удобства които предлагат тези приложения. Друг фактор оказващ влияние е използването на приложението само в локалната мрежа.

Началната страница ми е така наречената Form1. Тя е изградена от три леара с цел по-лесна работа и връзка с класовете при повикване. На следващата фигура 14 е показана визуално как изглежда началната страница на приложението.



фиг:14 Начален екран на приложението

Администратора има възможност да влезне от началната страница чрез бутона вход Админ . Връзката със даденият контрол става чрез зададени точни данни които съм декларира в класът на кода на програмата със стойности за име – admin, а за парола – password. При правилно въвеждане автоматично без натискане на бутон той влезе в даденият контрол за данни, при грешно излиза съобщение – грешни данни. На фигура 15 съм показал съответната опция за данни име и парола как се въвеждат без натискане на бутон вход.



Фиг. 15 Контрола при влизане на потребител

На фигура 16 показвам как след активиране с правилно потребителско име и парола на администратора, който само един му се активира следният прозорец. Първоначално базата данни винаги е празна, дори и да има натрупани данни в нея. След въвеждане на данни в полетата Име, Фамилия, Описани и дата на вписване, администратора има възможност да добави веднага данните в базата данни.

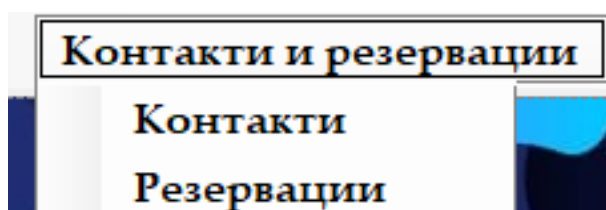
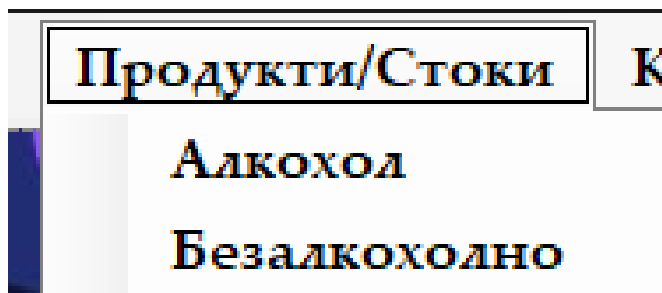
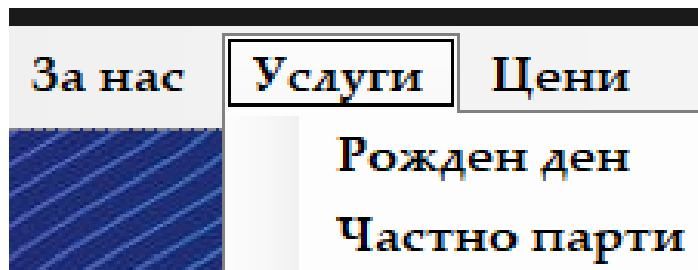
Те се генерират автоматично в онлайн режим и офлайнрежим и се виждат в таблицата. Всяко поле без значение на данните администратора има възможност да коригира, изтрива или да добавя нови данни. Когато приключи работа излиза от дадената форма с бутона затвори и се връща в първоначалният екран за да види влезнал ли е.

Фиг. 16 Влезнал админ форма

На фигура 17 е изобразено след правилно вписване в системата как изглежда потребителският профил и какви възможности могат да се правят с него, като след избиране на полетата в белия Layer и активиране на бутона Save автоматично в дясната част на сивото поле излиза неговото предпочитание като справка която ще се изпрати на админа и в базата данни на потребителя с името, с което се е вписал.

Фиг. 17 Влезнал потребител форма

На следващата фигура 18 изобразява началният екран с част от падащите подменюта. Всяка една дума от главното меню след позициониране на мишката върху нея се отварят допълнителните под менюта както се вижда на следните изображения във фигура 12.

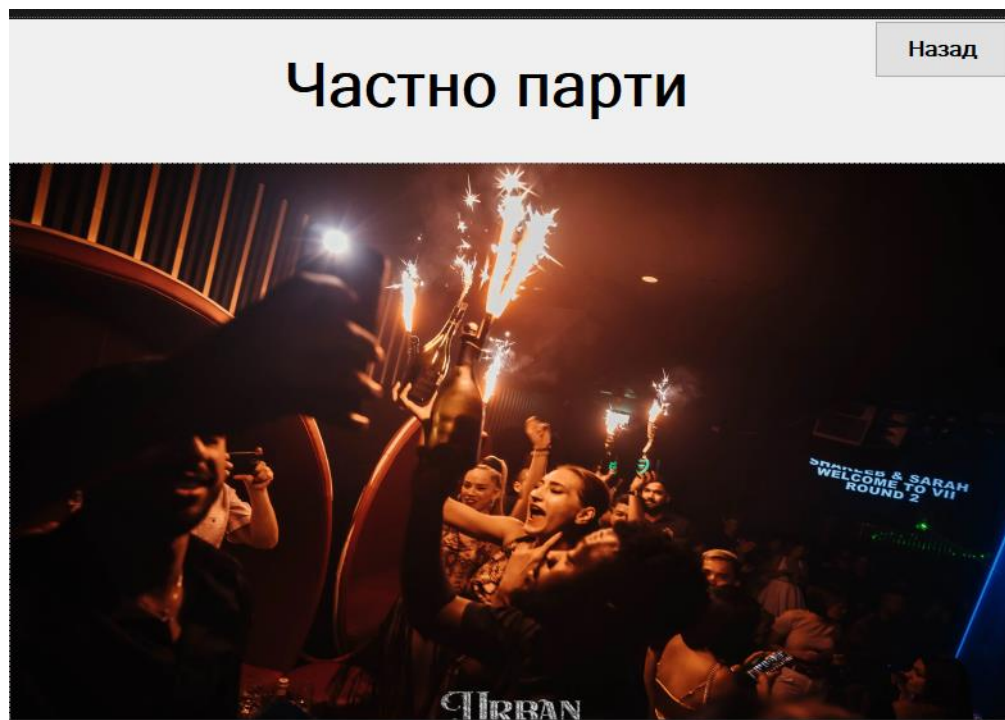


Фиг. 18 Подменюта

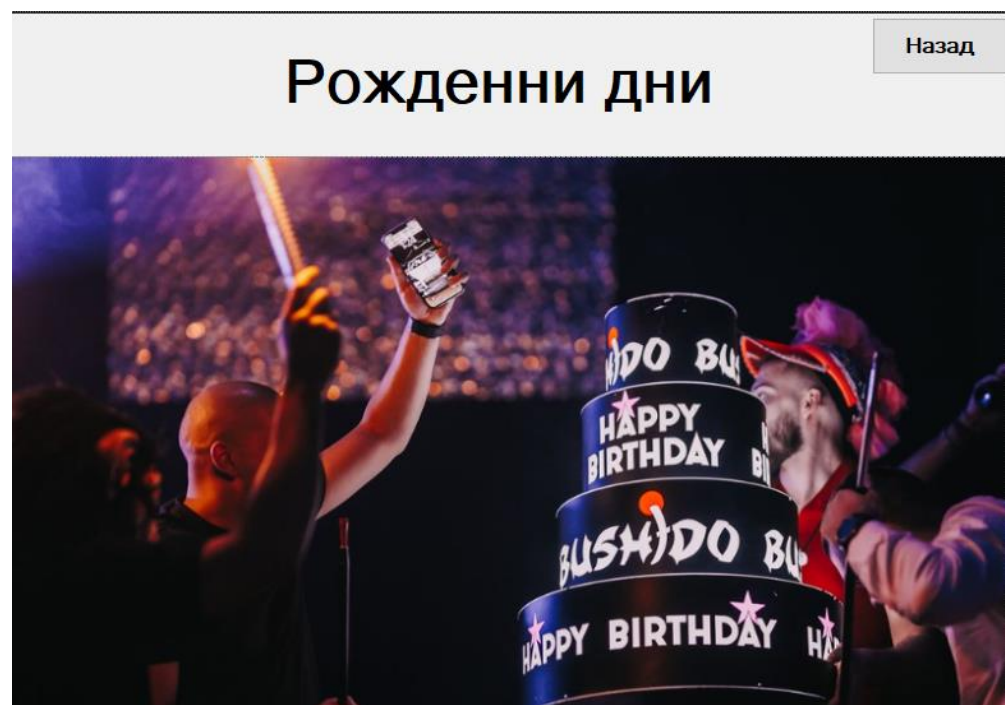
На следващите няколко фигури показващи останалите части от приложението показват другата част на самите контроли, а именно:

✚ Ако се избере дадено меню с услуги, продукти, резервации или участия се вижда пълното описание на участието или услугата както и нейната цена.





✚ В случай,че желаем да се върнем в началният екран избираме бутона назад и затваря даденият контрол и ни връща пак в началната форма на приложението.



фиг:19 Видове услуги



Фиг 20 Видове услуги

Участия			
Назад			
Сряда	Четвъртък	Петък	Уикенд
			
Simona AlphaMusic	Galin	Dj Damyan и Yavorcho	Без участие

фиг 21

Алкохол		
Назад		
 Уиски Dalmore King Alexander 3 700ml 40% 479.88лв.	 Водка Elit 1.0L 40% 82.50лв.	 Бяло вино с екстракт от билки 22 by Burdi Oaked 750ml 13.5% 25.20лв.
 Уиски Laphroaig Four Oak 1.0L 111.97лв.	 Водка Beluga 1.0L 40% 106.92лв.	 Червено вино Castra Rubra Butterfly's Rock 750ml 13.5% 39.84лв.

фиг 22 Контрола със част от стоките

Безалкохолни

 <p>КОКА КОЛА КЕН 330МЛ. 2,20 лв.</p>	 <p>МИН. ВОДА БАЛДАРАН 350МЛ. 1,50 лв.</p>	 <p>РЕД БУЛ 250МЛ. 4,80 лв.</p>
--	---	--

фиг 23


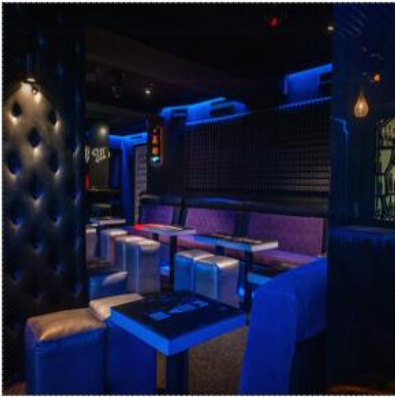

Цени за вход

Без участие/10лв/	DJ party /15лв/	Певец /20лв/
		

фиг 24 Ценова листа

Резервации и цени

[Назад](#)

		
Маса тип щъркел от 1-5 човека 80лв консумация <div style="border: 1px solid black; padding: 5px; width: fit-content; margin: 0 auto;">Резервирай</div>	Сепаре за 6-10 човека 500лв консумация <div style="border: 1px solid black; padding: 5px; width: fit-content; margin: 0 auto;">Резервирай</div>	ВИП сепаре 1000лв консумация <div style="border: 1px solid black; padding: 5px; width: fit-content; margin: 0 auto;">Резервирай</div>

25 Форма за резервации

4.7. Реализация на база данни

4.7.1. Генериране на кода на базата данни

За създаването и управлението на базата данни се използват СУБД (Системи за управление на бази данни). Всяка СУБД е програмна система, чието предназначение е да създава и управлява данните, организирани в базата от данни. Пълноценното функциониране на една СУБД налага тя да осигурява изискванията към БД:

- разделяне на описанието на данните от тяхната обработка; логическа и
- физическа независимост; минимално излишество на данни в БД; удобен
- потребителски интерфейс; осигуряване цялостност на данните, т.е.
- логическа непротиворечивост в БД.

В настоящата дипломна работа за физическата реализация на базата данни е използван SQL Server. Сървърът поддържа всички важни характеристики на съвременните RDBMS системи (съхранени процедури, транзакции и т.н.). Характерно за него е лесната администрация, подобно на останалите сървъри на Microsoft, добра производителност, висока скалируемост и висока надеждност.

```

68     }
69
70
71     private void adminMenu_Load(object sender, EventArgs e)
72     {
73     }
74
75
76     private void loadbtn_Click(object sender, EventArgs e)
77     {
78         dataBaseTableAdapter.GetData();
79         dataBaseTableAdapter.Fill(diplomen_proektDBDataSet.dataBase);
80     }
81
82     private void textBox3_TextChanged(object sender, EventArgs e)
83     {
84     }
85
86
87     private void textBox5_TextChanged(object sender, EventArgs e)
88     {
89     }
90
91
92     private void label1_Click(object sender, EventArgs e)

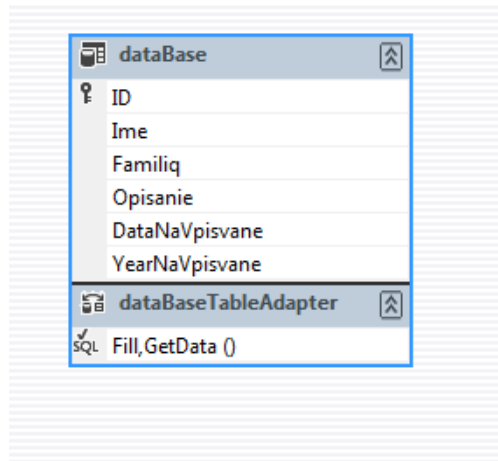
```

фиг.26 Генериране на база данни

4.7.2. Логическа структура на базата данни

База данни е структурирано множество от постоянна информация, съхранявана от компютърна програма. Терминът постоянен означава, че данните продължават да съществуват и след прекратяването на програмата или на потребителската сесия, която ги е създавала. Релационна база данни е съвкупност от такава информация, съхранена в двумерни таблици.

Базата данни на приложението съдържа общо 3 таблици. При проектирането на базата данни са взети под внимание правилата за нормализация. Въпреки това някои от таблиците са денормализирани с цел по-бързото генериране на нужния набор от данни. За да се изпълни изискването за минимален трафик между базата данни и клиентското приложение, при обновяване на данните му трябва да се изпращат само новите данни, които са от значение за потребителите на приложението: за операторите – името им, за адрес на доставка – град и адрес, за избор на дестинация – име, за екипировка – име. За тази цел в таблиците, в които ще се съхраняват данни за оператори име, фамилия, описание, дата на вписване и година на вписване.



фиг. 27 Структура на база данни

```

409
410
411 [global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
412 [global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "15.0.0.0")]
413 public RequestedSongsRow AddRequestedSongsRow(string Ime, string Familij, string Opisanie, string _Data_na_vp
414     RequestedSongsRow rowRequestedSongsRow = ((RequestedSongsRow)(this.NewRow()));
415     object[] columnValuesArray = new object[] {
416         null,
417         Ime,
418         Familij,
419         Opisanie,
420         _Data_na_vp,
421         YearOfCreations);
422     rowRequestedSongsRow.ItemArray = columnValuesArray;
423     this.Rows.Add(rowRequestedSongsRow);
424     return rowRequestedSongsRow;
425 }
426
427 [global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
428 [global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Data.Design.TypedDataSetGenerator", "15.0.0.0")]
429 public RequestedSongsRow FindByID(int ID) {
430     return ((RequestedSongsRow)(this.Rows.Find(new object[] {
431         ID})));
432 }
433 [global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
  
```

фиг. 28 Код на администраторския панел на база данни

V. Заключение

Приложимостта на програмата е разнообразна – както може да се ползва за малки фирми, в които има към 1000 потребителя, които ползват Exchange услугите, така и в корпоративни фирми, които имат към няколко сайта с по 5 000 потребителя. Скрипта на програмат може да се адаптира и редактира с изискванията на клиента.

Съществува графичен интерфейс за възтановяване.

Технологията е все още млада, но има потенциал за развитие и автоматизиране на живота на крайният потребител.

Десктоп приложенията се предпочитат пред уеб приложенията, когато трябва да се предостави офлайн функционалност или интеграция със специализиран софтуер и хардуер. Те не са добро решение, когато целевите потребители са разнородна публика вън от организацията.

Централен аспект на приложението се явяват данните и кеширането. За да бъде успешна имплементацията, проектирането трябва да им обърне сериозно внимание. Трябва да се реализира централизиран подход, чрез изграждането на кешираща инфраструктура. Тя трябва да поддържа техники за съхранение на данните, стратегии за загуба на давност и стратегии за изчистване на кеша.

Данните са сравнително постоянни във времето или краткотрайни. Подходящо за краткотрайните данни е оптимистично противодействащо кеширане за кратък период, а за данните само за четене, предварително дълготрайно кеширане.

Основните подходи при разработване на офлайн десктоп приложения са два: базиран на данни и ориентиран към услуги. Подходът базиран на данни използва релационна база данни за поддържането на офлайн функционалността, синхронизирането, заключването и изглаждането на. Разработчикът е отговорен за дефиниране на бизнес правилата. При подхода ориентиран към услуги обаче, разработчикът трябва да се погрижи за много повече аспекти – реализиране на асинхронната комуникация и минимизиране на сложните мрежови взаимодействия

VI Използвана литература

1. Наков, С. и колектив, Програмиране за .Net Framework, том 2, Барс, 2007.
2. Гъров К., Ст. Анева, За задачите в модула “Събитийно програмиране с Visual Basic 6.0”. Задачи, реализиращи приложения за връзки с бази от данни. Задачи за приложения – тип меню, сп. “Математика и информатика”, приложение на кн. 3, 2004 г., стр. 1-20 (от приложението)
3. Гъров, К., Задачите в обучението по информатика и информационни технологии, Сборник доклади на Национална конференция „Образованието в информационното общество”, Пловдив, 27-28.05.2010, 95-101.
4. Iliev, A., N. Kyurkchiev, Nontrivial Methods in Numerical Analysis: Selected Topics in Numerical Analysis, LAP LAMBERT Academic Publishing GmbH & Co. KG, Saarbrücken (2010).