

МИНИСТЕРСТВО НА ОБРАЗОВАНИЕТО И НАУКАТА

**ПРОФЕСИОНАЛНА ГИМНАЗИЯ “ГЕН. ВЛАДИМИР ЗАИМОВ” гр. СОПОТ**

4330 гр. Сопот, ул. ”Иван Вазов” №1, тел./факс: /03134/ 83-31, 83-32, e-mail: [pgzaimov@yahoo.com](mailto:pgzaimov@yahoo.com)

# 

ДИПЛОМЕН ПРОЕКТ

**Тема:**

**РАЗРАБОТКА НА DESKTOP ПРИЛОЖЕНИЕ ОРГАНИЗИРАЩО ДЕЙНОСТТА НA ФИРМА ЗА КЛИМАТИЧНИ ИНСТАЛАНЦИИ.**

**Дипломант: Консултант:**

Кристиан Богдански Милена Христозова

№ 13, 12А клас

**Специалност: код 4810201**  „Системно Програмиране“

Сопот – 2023

# Съдържание

**1. Увод**

1. Цел на дипломната работа ..........................................................................................4

2. Структура на дипломната работа ...............................................................................4

**2. Проучване**

2.1. Използвани технологии и инструменти ……………………………………….. 5

2.2. Създаване на менюта във форма на приложение ………………………………. 6

**3 . Проектиране**

3.1. Модел на случаи на употреба (Use Case) .................................................................7

3.2. Модел на случаите на употреба на приложението.................................................12

**4. Реализация**

4. 1.Реализация на offline режима .................................................................................12

4.2. Реализация на online режима …………………………………………………….13

4.3 User Kontrol ………………………………………………………………………...17

4.4. Динамична реализация ............................................................................................19

4.5. Структура на приложението ...................................................................................21

4.6. Имплементация …………………………………………………………………...22

4. 7. Реализация на базата данни …………………………………………………….27

**5. Заключение** ……………………….……………………………………………..32

**6. Използвана литература** ………………………………………………………..….33

# 1. УВОД

С развитието на бизнеса и нарастването на конкуренцията се увеличава необходимостта от софтуерни решения, които максимално да автоматизират и улеснят бизнес процесите във всяка една фирма. Подобни решения могат да се състоят от много компоненти, които са интегрирани помежду си, обменят информация и работят като едно цяло. От особена важност за такива софтуерни решения е възможността за достъп до централизирана информация навсякъде, по всяко време и от различни устройства. Платформата .NET, създадена от Microsoft, обединява различни технологии, чрез които може да се реализира всеки един компонент от информационни системи, улесняващи бизнес процесите във фирмата. В настоящата дипломна работа ще бъде представено как са приложени няколко .NET технологии при създаването на информационна система, отговаряща за бизнес нуждите на фирма за планински туризъм.

## 1.1 Цел на дипломната работа

Настоящата дипломна работа има за цел приложение организиращо дейността на фирма за климатични инсталации

Фирма за климатични инсталации ,, И-клима“ е една от водещите в браншa вече няколко години. Фирма за климатични инсталации ,, И-клима“ отговаря на най-високите стандарти в своята сфера и се очертава като символ на престиж, висок професионализъм и качество.

Целта на дипломният проект е да се проектира и реализира трислойно MVP DESКTOP приложение организищо дейността на фирма за климатични системи, което трябва да поддържа CRUD операции за клиентите

Целите ще са да се постигне моментално и прецизно регистриране и отчитане на дейностите, невъзможни за постигане с досегашната система на ръчна обработка на информацията на хартия и таблици на Microsoft Excel.За съхранение на данните трябва да се проектира и реализира релационна база данни(Microsoft SQL Server).

Информационната система трябва да има две клиентски приложения за двата типа потребители, които я използват – *Администратори* и *Потребители*. Достъпът от клиентските приложения до базата данни трябва да се осъществява през XML Web Service.

## 1.2 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*** – Представлява 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  коментарии, 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.

# ПРОУЧВАНЕ

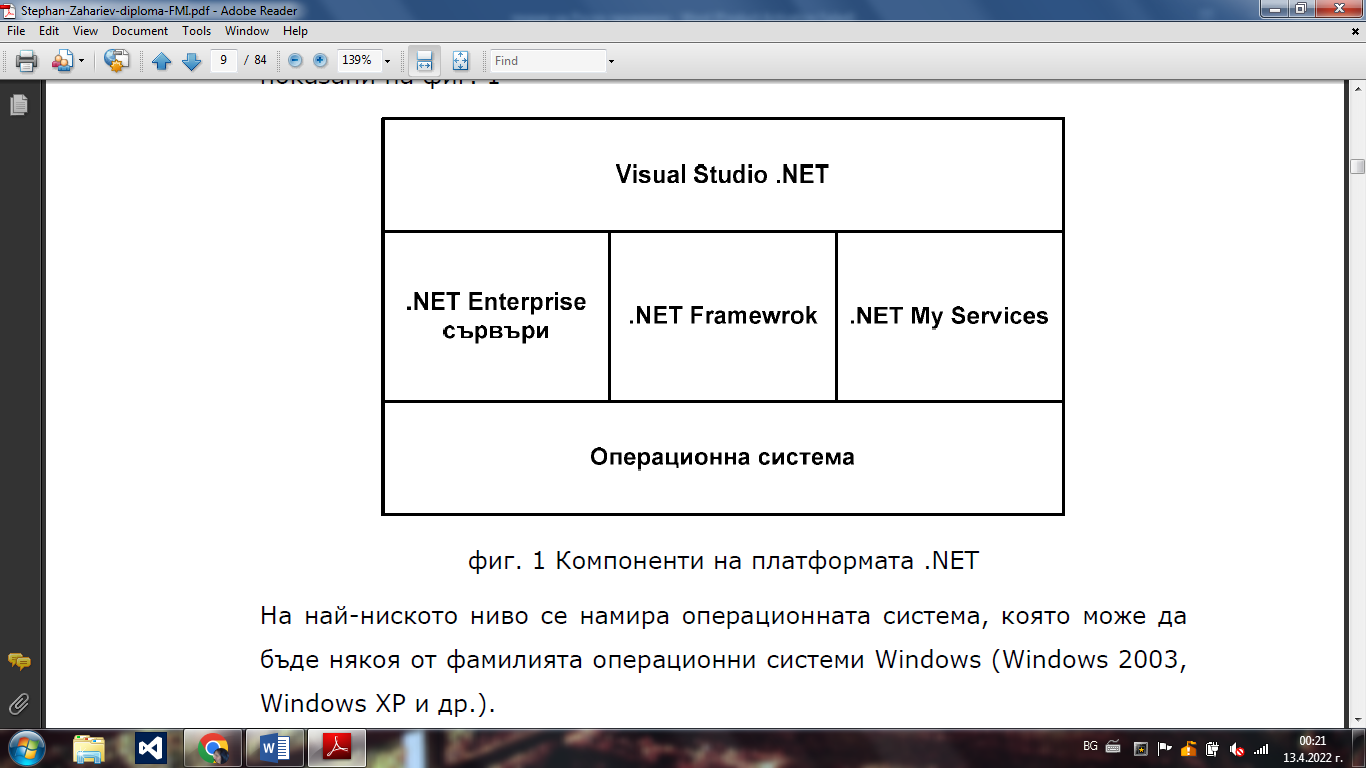
* 1. **Използвани технологии и инструменти**

Настоящата система е разработена на база на едни от последните технологии, предлагани от Microsoft. Изключение прави само NHibernate, който е проект с отворен код и се разработва от независими разработчици.

***Microsoft .NET Framework***

През юли месец 2000 година, Microsoft обявяват инициативата си за създаване на .NET платформата. Тя представлява нова среда за разработка на приложения, която идва с концепции, непознати дотогава в средствата за разработка на Microsoft. Същевременно се обединяват и технологиите на компанията създадени през 90-те години.

Ново обявената платформа се състои от 5 основни компонента, които са показани на фиг. 1



фиг.1: Компоненти на платформата .NET

На най-ниското ниво се намира операционната система, която може да бъде някоя от фамилията операционни системи Windows7,8,10,11 (Windows 2003, Windows XP и др.).

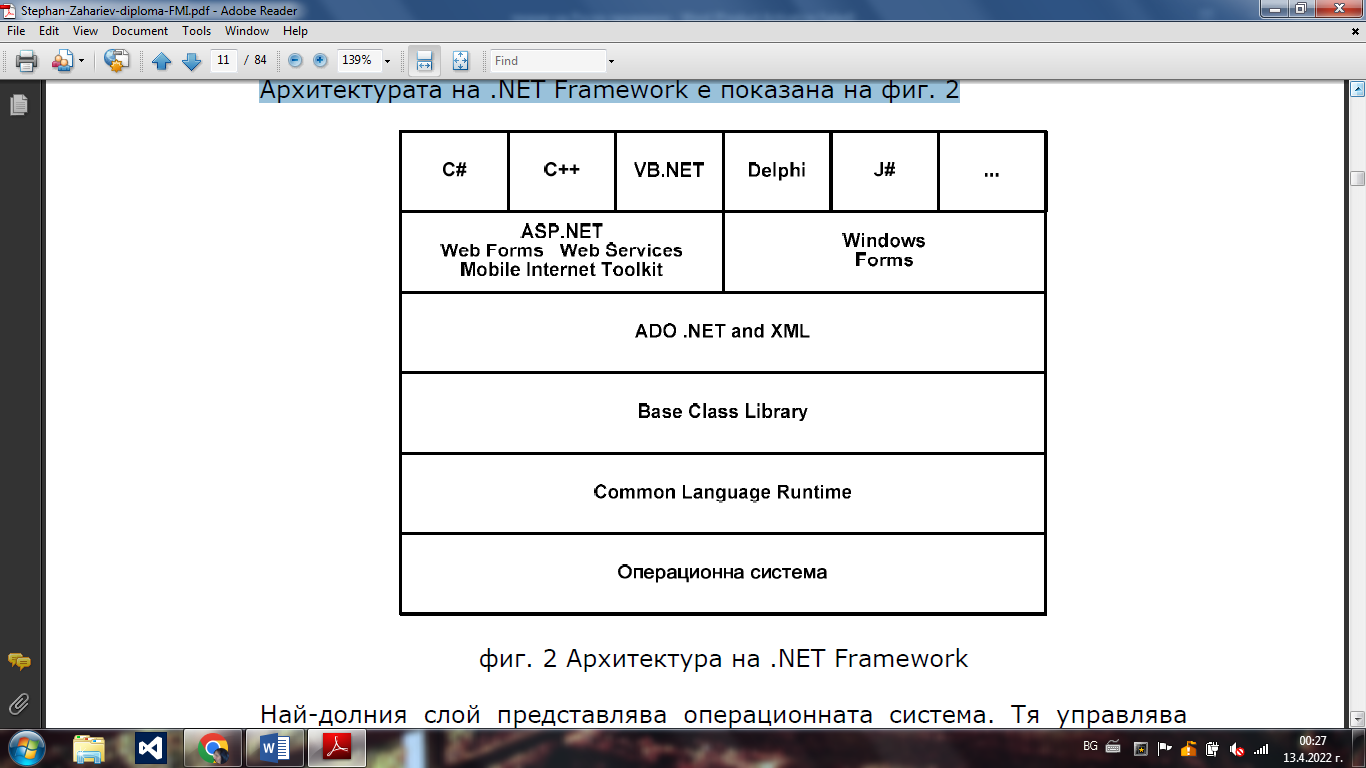
В центъра на .NET платформата се намира .NET Framework – нова среда за разработка и изпълнение на приложения. Тя предлага независимост от използване на езика за програмиране, те едно приложение може да бъда написано на няколко различни програмни езици. Освен независимост, се предлага и интеграция между различните езици. Това означава, че разработчикът може да наследява класове, обработва изключения и използва полиморфизъм във взаимодействащи си части на приложението писани на различни езици.

NET Framework се състои от следните части:

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

***Framework Class Library (FCL)*** – Представлява 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 .NET Framework

***Microsoft SQL Server или Access***

Microsoft SQL Server е система за управление на релационни бази от данни, разработвана от Microsoft. Характерно за него е лесната администрация, подобно на останалите сървъри на Microsoft, добрата производителност и увеличаващата се с всяка следваща версия надеждност. За съжаление Microsoft SQL Server работи само под операционната система Windows.

В момента актуалната версия на продукта е на Access, защото неговият обем на данни е до 2 GB, а в случаят моето приложение ще с малък обем от данни.

* 1. ***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.

# 3. ПРОЕКТИРАНЕ

В дизайна на почти всички приложения, едно от първите решения, които се взимат е как да се имплементира графичния потребителски интерфейс, така че да предостави на потребителите интуитивен подход за работа с програмата.

Реализацията на софтуерни системи с нарастваща сложност, паралелно от множество екипи от специалисти изисква прилагането на общи стандарти и ефективни методологии за разработка.

При изграждането на клиентския слой на системата е използвана библиотеката на .NET Framework за изграждане на прозоречно-базиран графичен потребителски интерфейс - Windows Forms. Windows Forms предоставя възможност за бързо изграждане на потребителски интерфейс тъй като е базирана на RAD(Rapid Application Development) концепцията. RAD е подход за разработка, при който приложенията се създават визуално, чрез сглобяване на готови компоненти посредством помощници и инструменти за автоматично генериране на голяма част от кода. Windows Forms дефинира набор от класове и типове, част от които са използвани при реализирането на двете клиентски приложения на 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.

**С*лучай 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*:

А1 1. Потребителят е успешно идентифициран.

А1 2. При наличие на интернет връзка приложението обновява локалния XML файл. В противен случай преминава в offline режим на работа.

А1 3. Приложението показва на потребителя форма с име Потребителски профил с меню за възможни заявки за определена дестинация или екипировка”;

А1 4. Изход.

*Алтернативен поток 2*:

А2 1. Потрбителят не е успешно идентифициран.

А2 2. Приложението показва съобщение, че е въвел грешно парола или име и дава възможност отново във форма „Вход потребител ” да се идентифицира повторно.

А2 3. Край.

**С*лучай 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*:

А1 1. При наличие на връзка със сървъра приложението обновява локалния XML файл, със получените данни от сървъра.

А1 2. Приложението запазва валидната хеш стойност във файла с настройки.

А1 3. Приложението показва на администратора форма „Главно меню”;

А1 4. Край.

*Алтернативен поток 2*:

А2 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.

А2 2. Приложението показва на администратора форма „Главно меню”;

А2 3. Край.

**С*лучай 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*:

А1 1. При наличие на връзка със сървъра приложението получава новите данни и данните, които трябва да се изтрият(ако има такива).

А1 2. Приложението обновява базата данни в паметта и я записва в локалния XML файл.

А1 3. Край.

*Алтернативен поток 2*:

А2 1. При липса на връзка със сървъра приложението преминава в offline режим.

А2 2. Край.

**С*лучай на употреба - Преглед на записи съхранени в апарата (Review offline saved data)***

Потребителят може да преглежда записи за дейността , 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 дадения климатик или дейност 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.

**С*лучай 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.

*Основен поток от действия*:

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

**С*лучай на употреба - Оffline 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.

**С*лучай на употреба - Оnline 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 за желани 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 адми-нистратора да избере типа на следващата дейност.
6. Край.

**С*лучай 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.

*Предусловия*: Потребителя е успешно идентифицира.

*Основен поток от действия*:

1. Потребителят въвежда данни.

*Алтернативен поток 1*:

А1 1. Потребителя натиска бутона *„Напред”*.

А1 2. Въведените данни се валидират.

А1 3.Приложението показва следващата форма на потребителя

*Алтернативен поток 2*:

А2 1. Потребителя натиска бутона *„Затвори”*.

А2 2. Приложението показва предходната форма.

А2 3. Потребителя редактира вече въведените от него данни.

### **3.2. Модел на случаите на употреба на приложението**

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

Administrator

Set Settings

Enter new data

Make reports

«uses»

«uses»

Log in

«uses»

Update data

*фиг.3: Модел на потребителските случаи – Use Case Model за приложението*

**4. РЕАЛИЗАЦИЯ**

#### 4.1. Реализация на offline режима

Приложението трябва да предоставя на потребителя възможност за работа и при липса на интернет връзка. Въведените от потребителят данни трябва да се съхраняват в апарата и при поява на интернет връзка да се съхранят на сървъра. В глава II са представени различните стратегии за съхранение на данни при PocketPC приложения: релационна база данни(Microsoft SQL Server CE), в локален файл (XML файл), и в базирана на сесии структура данни, в паметта.

В приложението данните се съхраняват в локален XML файл, който се зарежда в паметта при стартиране на приложението. Избора за използване на XML файл за съхранение на данни се ръководи от няколко факта:

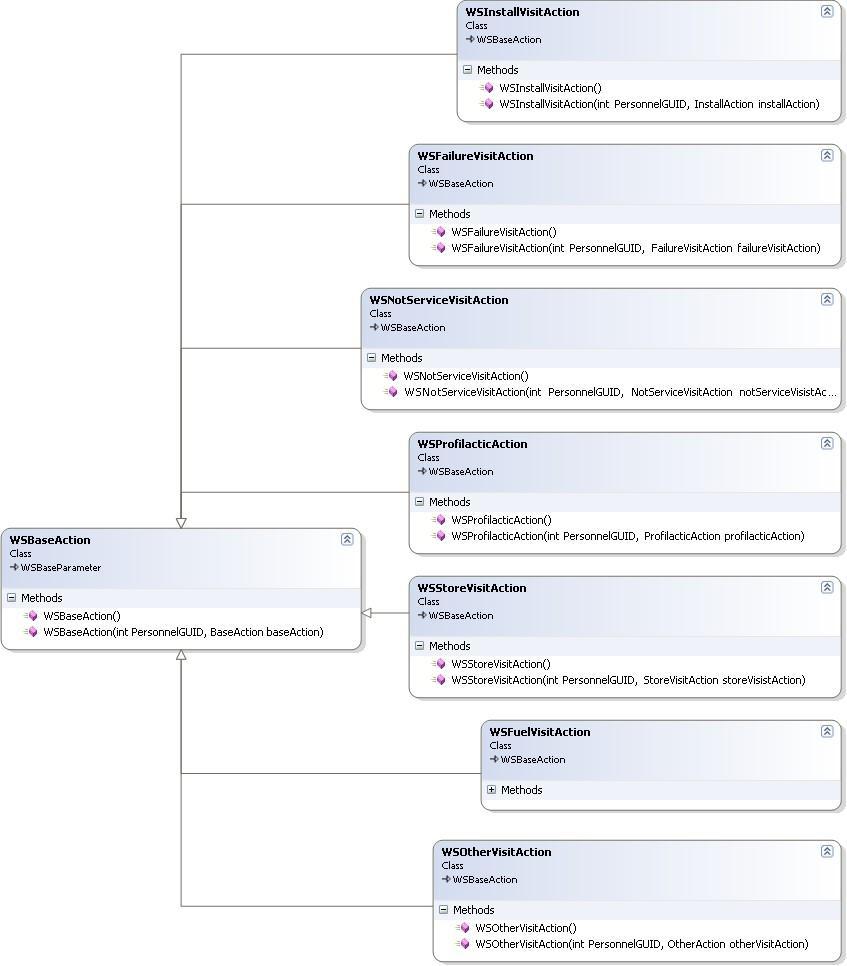
* Количеството данни, което трябва да се съхранява на PocketPC в повечето случаи ще бъде между 50 и 100 kb. Зареждането им в паметта ще има подобра производителност от колкото достъп до тях през SQL Server CE Query Engine.
* Обмена на данни между PocketPC и Web Servece ще е малък(около 1Mb). Употребата на XML и Web Service е предпочитана, и .NET Compact Framework предоставя широка поддръжка и на двете технологии. Локалния XML файл има следната схема:

**4.2.** **Реализация на online режима**

Приложението обменя данни със сървъра чрез асинхронно извикване на Web услуга. При асинхронното извикване на Web услуга потребителския интерфейс не замръзва докато приключи обръщението.

|  |
| --- |
| public class WebProxy  {  private class Nested  {    internal static readonly WebProxy instance = new WebProxy(); static Nested()  {  }  } ...    public static WebProxy instance  { get {  return Nested.instance;  } }    ...  } |

*WebProxy* класа представлява отдалечено пълномощно (*remote proxy*), разновидност на шаблона пълномощно, който предоставя локален представител на обект от друго адресно пространство. Този вид пълномощно се срещна в литературата под името „посредник”. *WebProxy* класа съдържа референция към, автоматично генерирания клас, за достъп до Web услугата. В *WebProxy* класа са дефинирани методи, съответстващи на методите предоставени от Web услугата. В класа се централизира обработката на общи изключения, като изключения възникнали при обмен на данни със сървъра или изключения при автентикация на потребителя. Това става в метода *GetWebCallResult*.



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

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

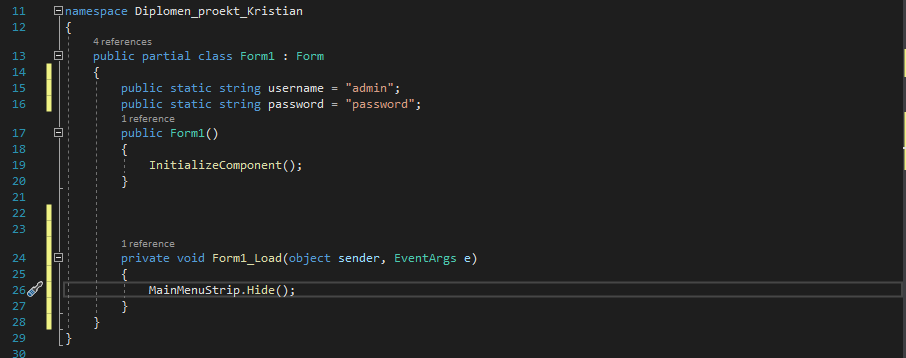
За да може да се маркира всеки Web метод с атрибут за компресия, се създава клас *CompressionSoapExtensionAttribute***,** наследник на класа *SoapExtensionAttribute***:**

|  |
| --- |
| using System; ... using ICSharpCode.SharpZipLib.GZip;    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; }  }  } |

Свойството *ExtensionType* връща типа имплементиращ допълнителната логика *(CompressionSoapExtension*), а свойството *Priority* указва реда на изпълнение, когато има няколко допълнения едновременно.

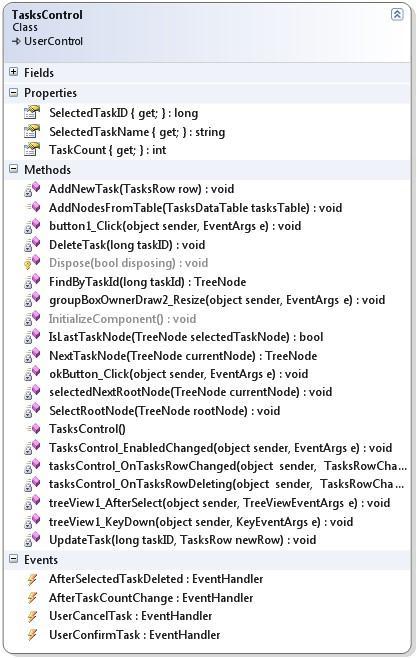
Имплементацията на конкретната логика за добавяне на функционалността е клас, който е наследник на *SoapExtension* (в *System.Web.Services.Protocols*пространството от имена) :

|  |
| --- |
| 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; } |

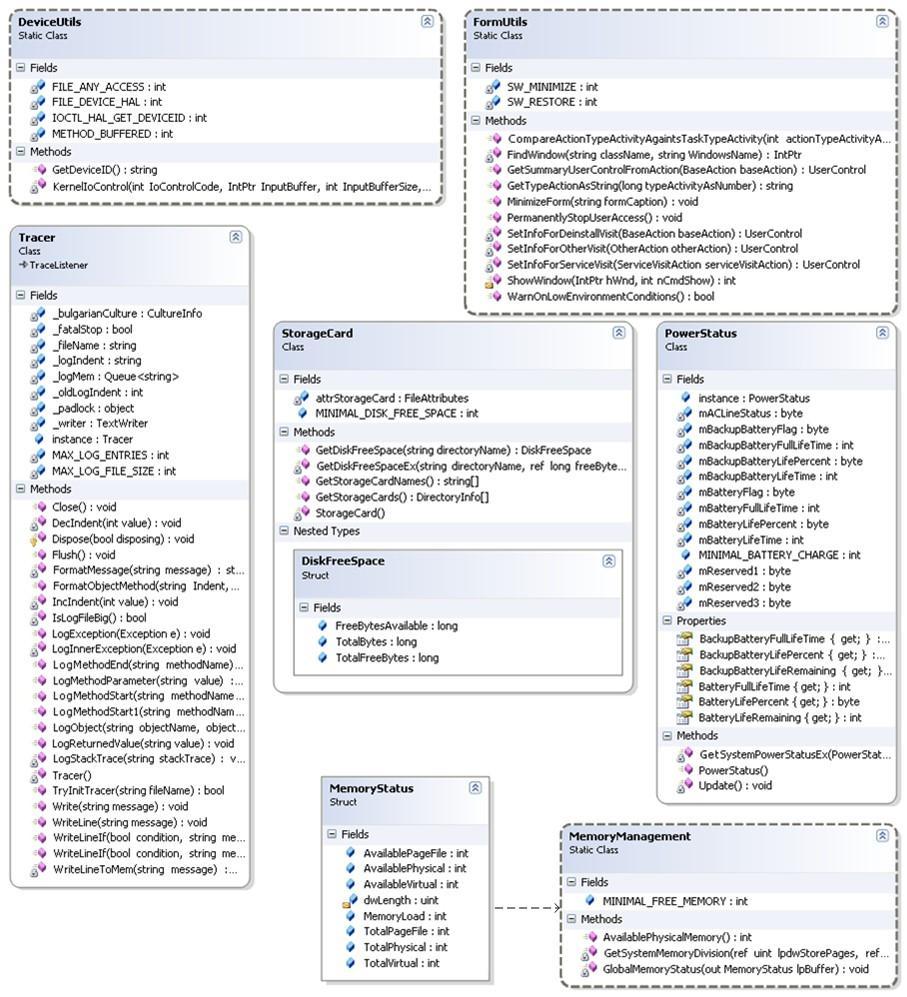


#### 4.3. User контроли

За да може да се преизползва на няколко места функционалността за показване и избиране на задачи, е имплементиран класа, наследник на *UserControl*. Контрола предоставя възможност за зареждане на задачите от таблицата на типизирания dataset *DBMobile*, във *TreeView* контрол(дървовидна колекция от именувани обекти, всеки представен чрез *TreeNode*). Контрола предоставя методи за добавяне на задача в дървото по подаден ред - *TasksRow* от таблицата *Tasks,* обновяване на задача, премахване на задача от дървото. Чрез свойствата на контрола могат да се извлече информация за броя задачи, името на избраната задача и ID-то на задачата. *TasksContorl*-a дефинира и събития, които известяват обектите абонирани за тях, за изтриване на избрана задача, смяна на броя на задачи, за отказ от задача или за избор на задача.



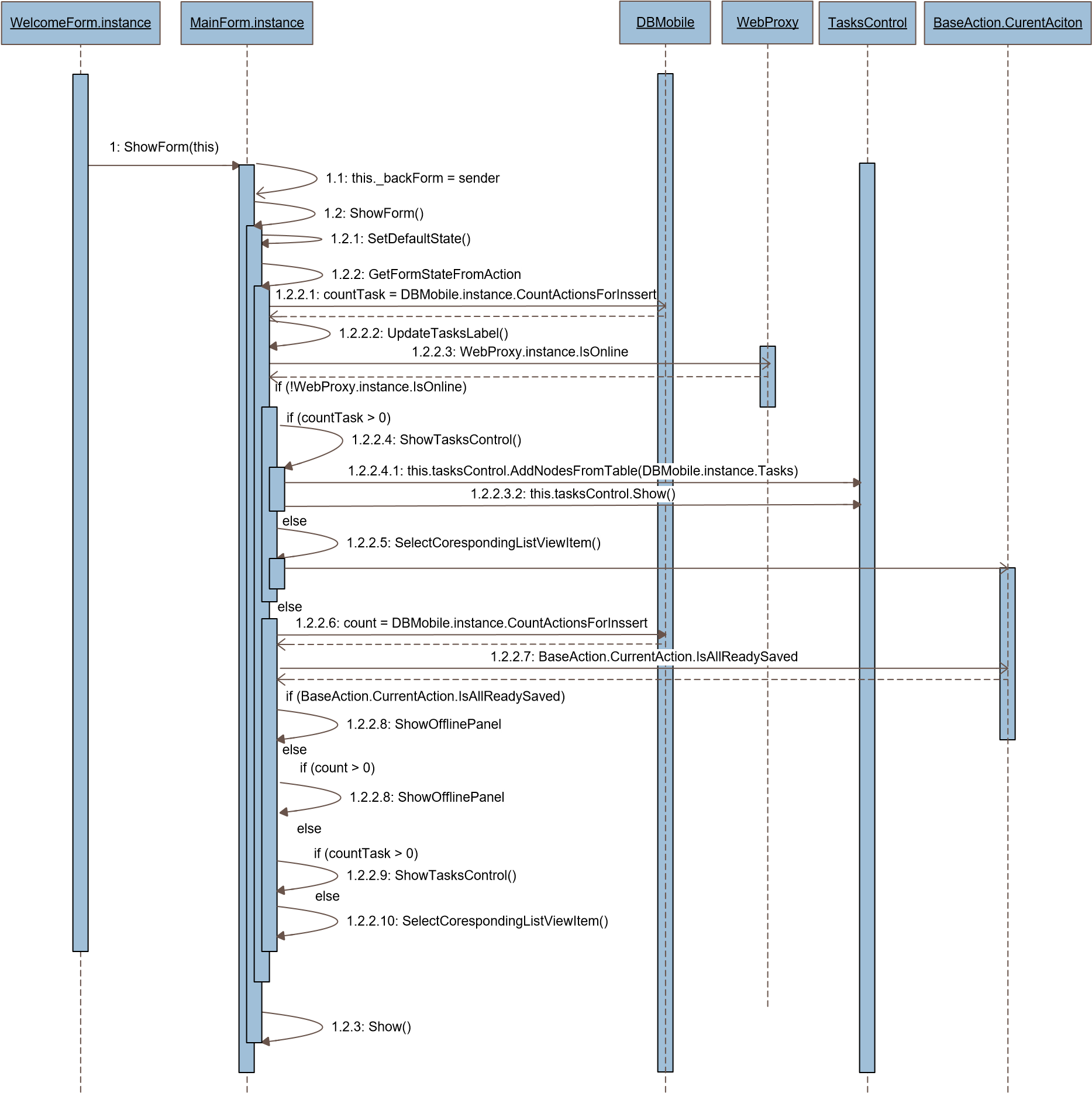
Имплементацията на класовете може да се види в кода предоставен към дипломната работа.



### фиг.5: Клас диаграма на класовете в пространството от имена CompactClient.CompactFrameworkUtils

#### 4.4. Динамична реализация

След като са разгледани основните класове, в следващите диаграми на последователности ще бъдат показани какви обекти се създават и как си взаимодействат в различни потребителски случаи.



фиг.6: Диаграма на последователността за метода ShowForm(IBaseForm sender) на класа MainForm

На диаграмата е представена базовата последователност от съобщения, които се изпращат между обектите, при показването на форма „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 процеса 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.

Program

SplashForm

1

:

splash

=

new SplashForm

()

;

Application

.

Run

(

splash

)

WelcomeForm

2

:

Application

.

Run

(

WelcomeForm

.

instance

)

3

:

LoadSettings

()

3

.

1

:

LoadSetings

()

3

.

2

:

ApplySettings

()

5

:

StopSplash

()

6

:

Refreh

()

DBMobile

4

:

LoadDataBase

()

7

:

Invoke

(

new EventHandler

(

splash

.

KillMe

))

;

Dispose

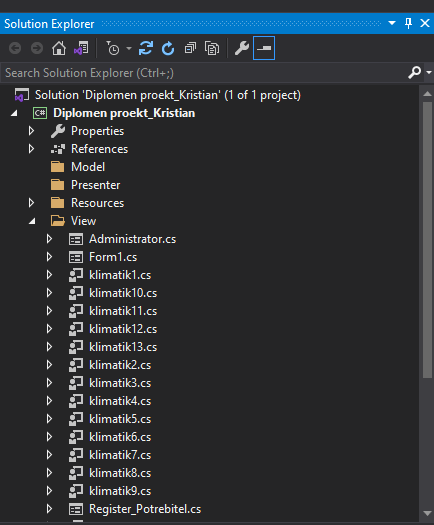
()

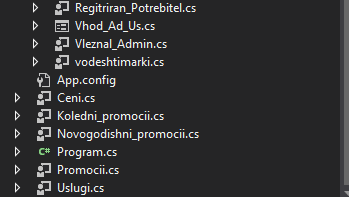
;

*фиг.7: Диаграма на последователностите при стартиране на приложението*

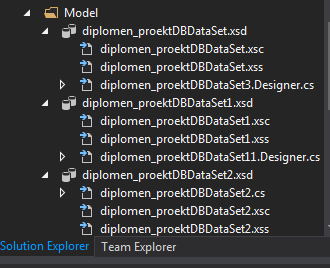
#### 4.5. Структура на приложението

#### 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 8 поставих базата данни, в папката View 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.8: Структура на приложението

**

фиг.9: База данни

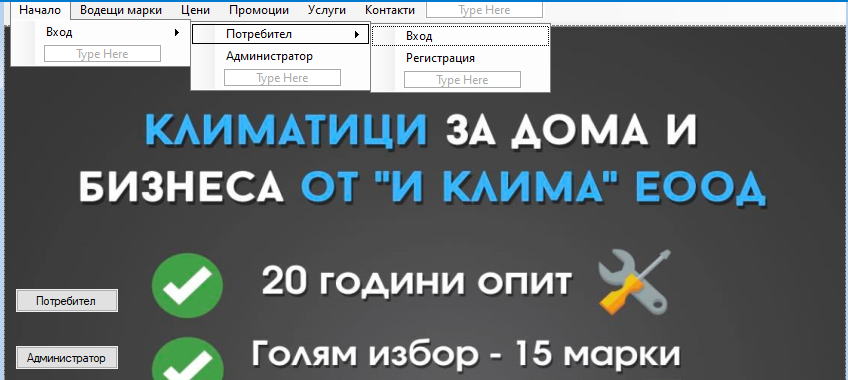
## 4.6 Имплементация

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**

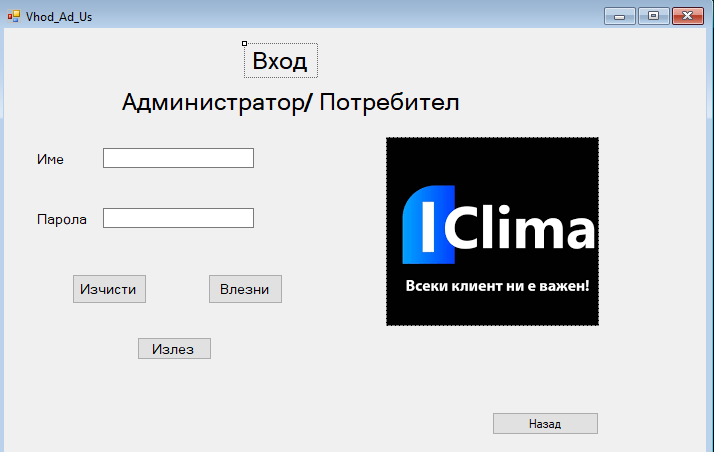
#### Потребителският интерфейс е изграден на базата на библиотеката Windows Forms, част от .NET framework. Интерфейса е проектиран като диалогово базирано Windows приложение, заради по-големите удобства които предлагат тези приложения. Друг фактор оказващ влияние е използването на приложението само в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

Началната страница ми е така наречената ,,И-клима“. Тя е изградена от три леара с цел по лесна работа и връзка с класовете при повикване. На следващата фигура 10 е показана визуално как изглежда началната страница на приложението.



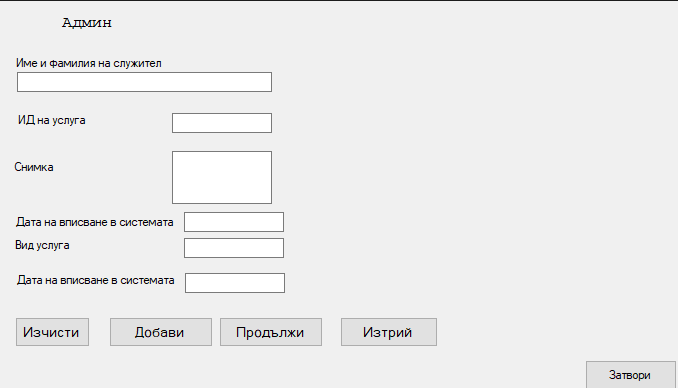
фиг.10: Представлява началната страница

Администратора има възможност да влезне от началната страница чрез бутона вход Админ . Връзката с даденият контрол става чрез зададени точни данни които съм декларирал в класът на кода на програмата със стойности за име – admin, а за парола – password. При правилно въвеждане автоматично без натискане на бутон той влиза в даденият контрол за данни, при грешно излиза съобщение – грешни данни. На фигура 11 съм показал съответната опция за данни име и парола как се въвеждат без натискане на бутон вход.

****

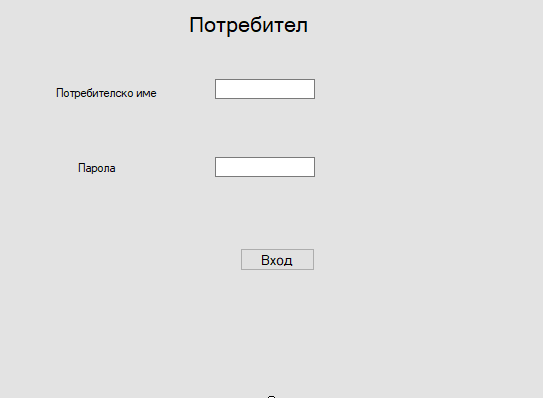
фиг.11: Вход администратор/потребител

На фигура 12 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 офлайн режим и се виждат в дясно 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.12: Добавя на данните в базата данни

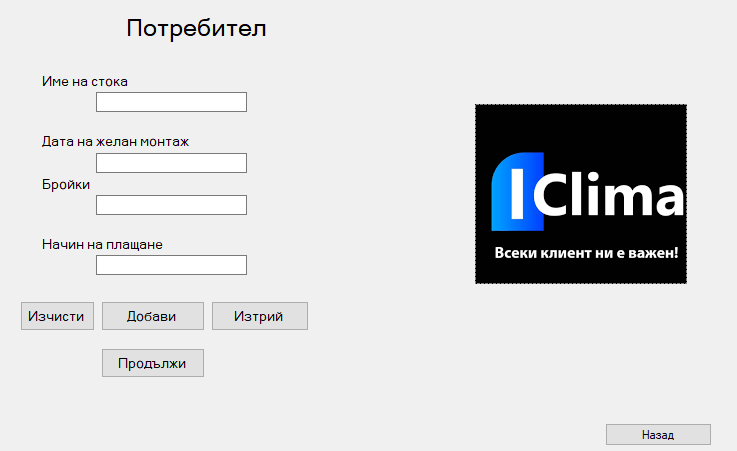
На фигура 13 се изобразява началният екран на потребителя който е регистриран. При него има вече бутон вход за разлика от админа. За негово улеснение и в случай на грешни данни в полетата излиза надпис грешни данни опитай пак.

фиг.13: Началния екран на потребителя който е регистриран

На фигура 14 е изобразено след правилно вписване в системата как изгледажда потребителският профил и какви възможности могат да се правят с него, като след избиране на полетата в жълтият 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. Останалите бутони запиши и изтрити са в случай че се откаже .

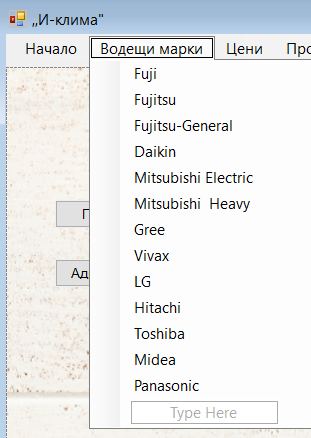
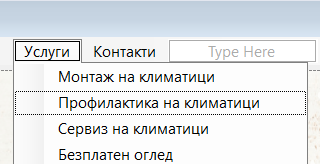
В случай, че потребителя желае да направи поръчка на друг човек с друго име или фамилия те автоматично влизат в базата данни независимо дали е онлайн или офлайн системата, достатъчно е да активира бутона запиши.

.

фиг. 14: Потребителският профил

На следващата фигура 15 изобразява началният кран с част от падащите подменюта и отделни услуги.

Всяка една дума от главното меню след позициониране на мишката върху нея се отварят допълнителният под менюта както се вижда на следният изображения в фигура 13.



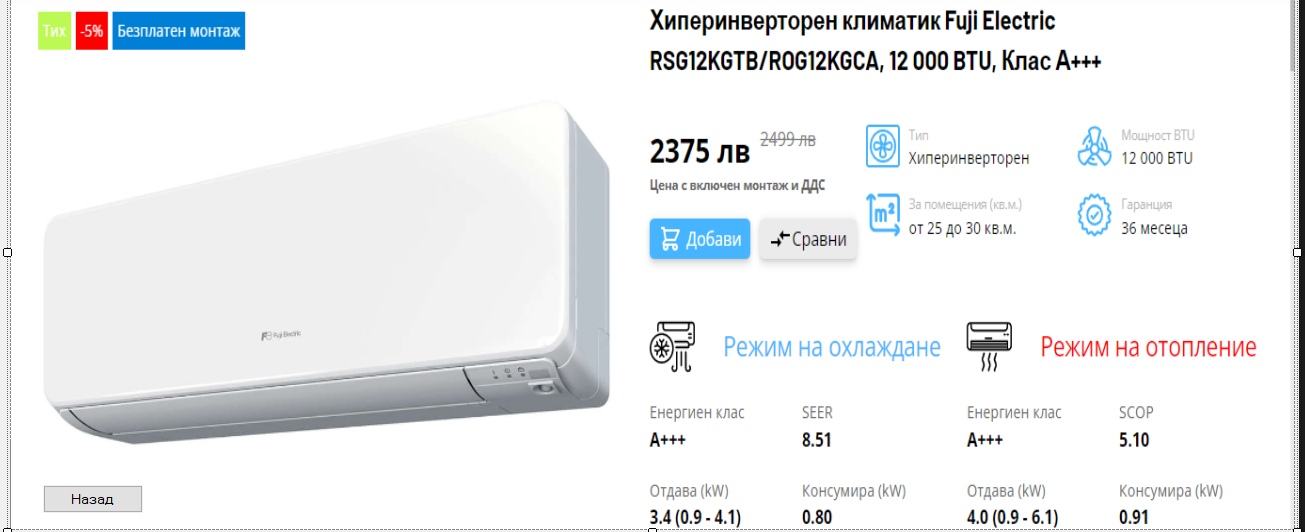
фиг.15: Част от падащите подменюта и отделни услуги.

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

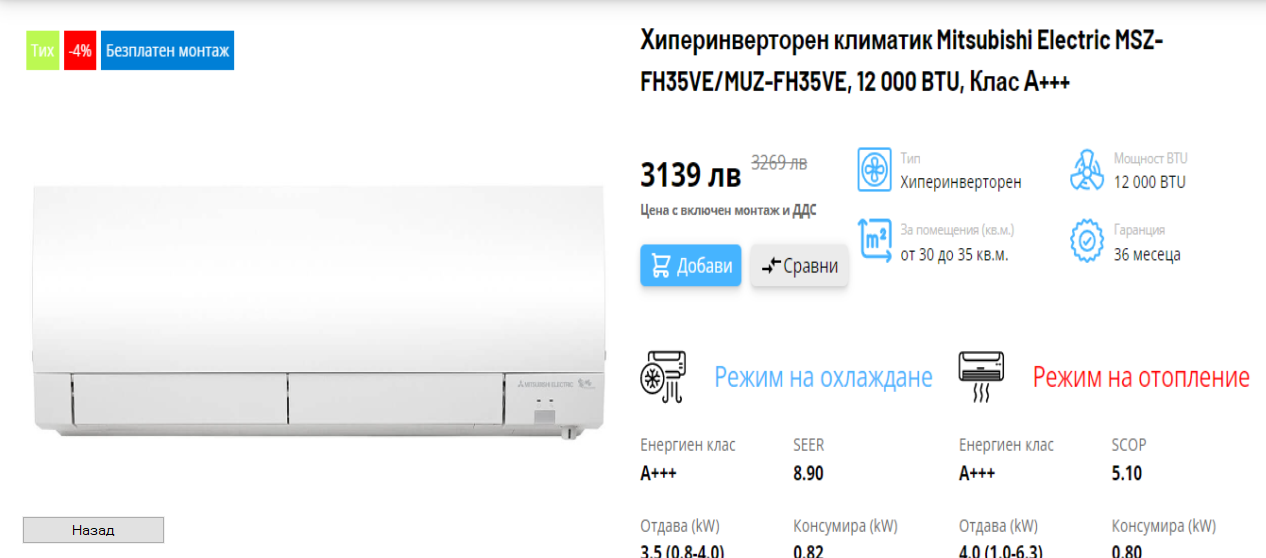
Ако се избере дадено меню с водещи марки се вижда пълното описание на обекта или избраната марка както и нейната цена.

Ако се избере услуги излизат изображения като заставайки върху дадено изображение имаме връзка да отворим и разгледаме пълоното описание за него.

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

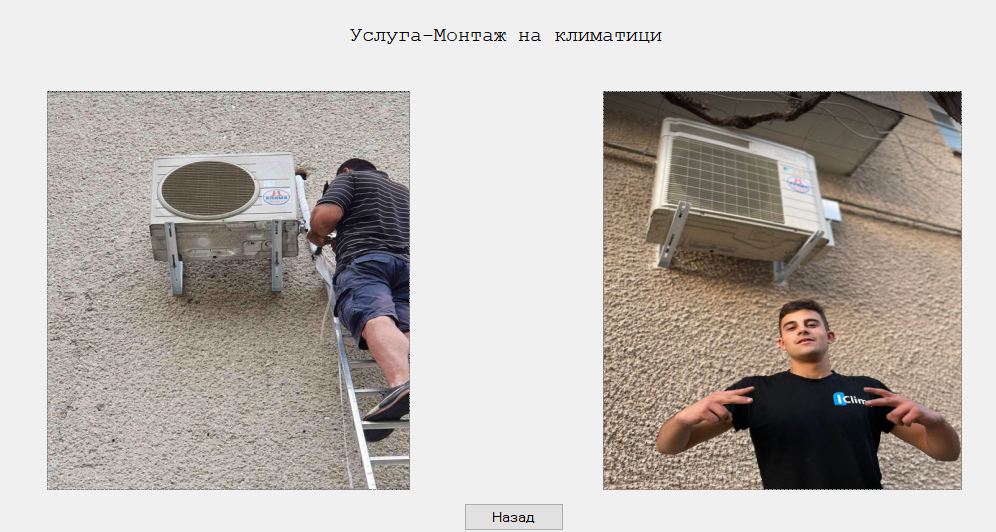
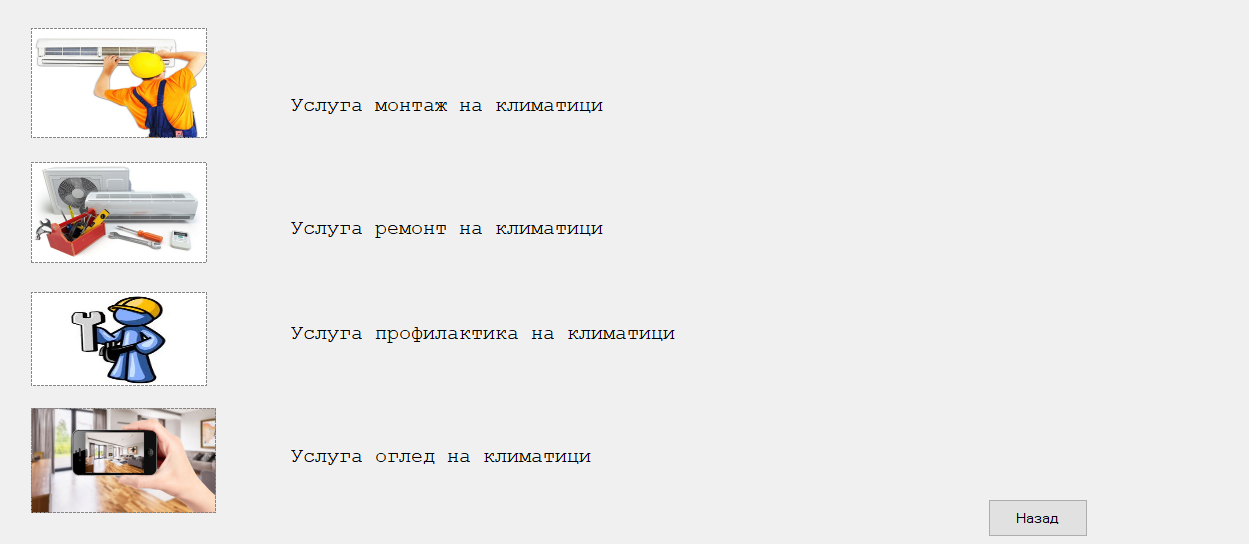


фиг.16: Останалите части от приложението(контроли)

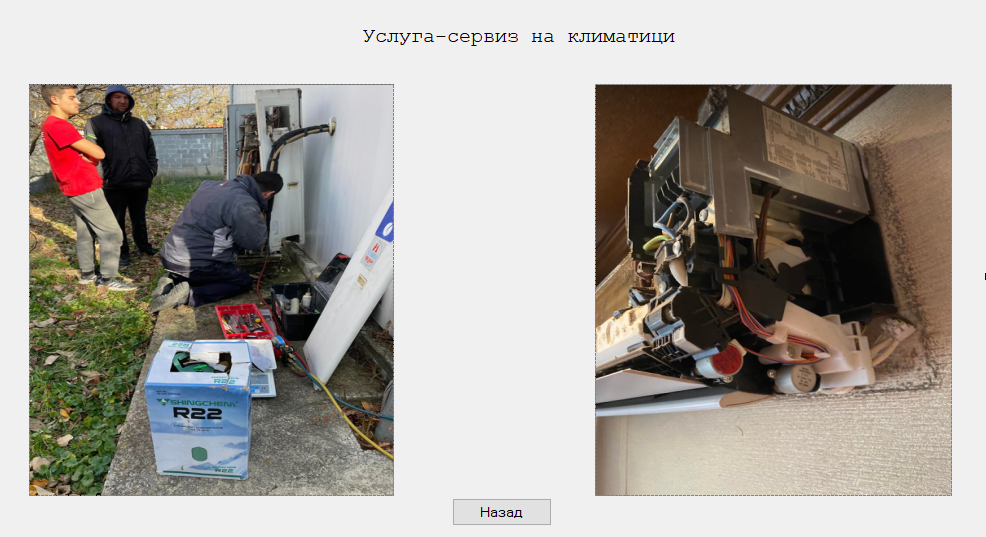


фиг.17: Останалите части от приложението(контроли)

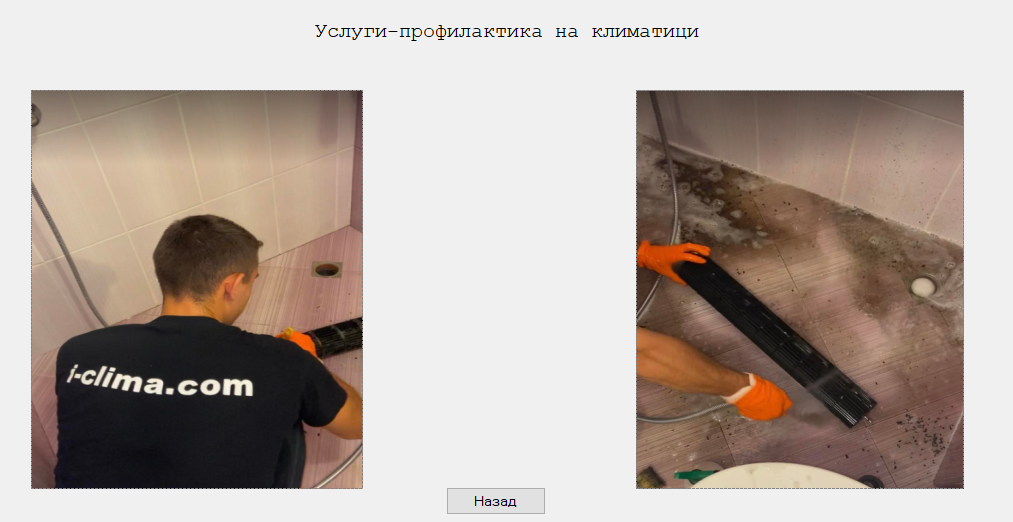
фиг.18: Видове услуги



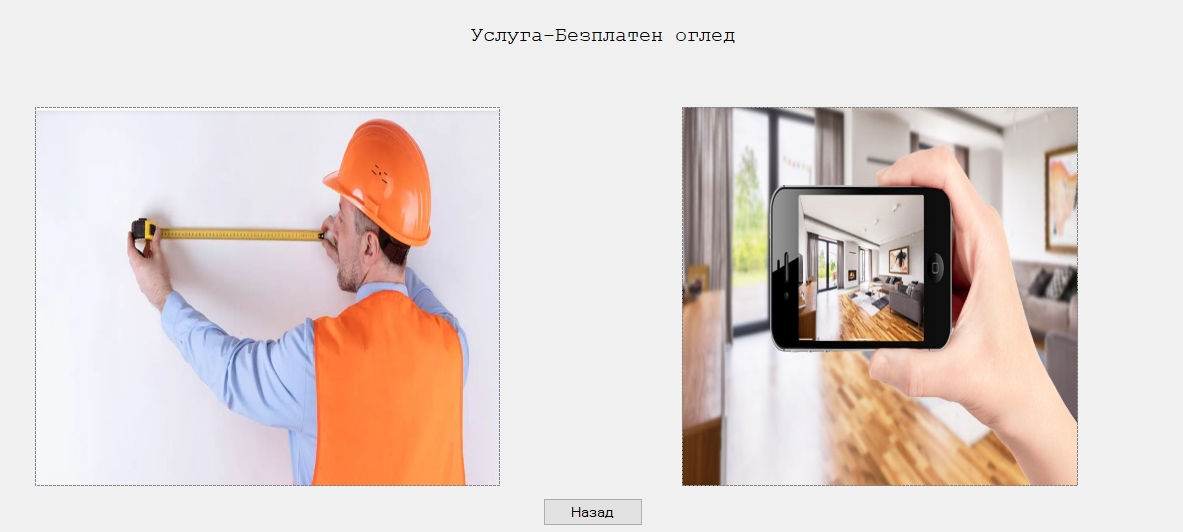
фиг.19: Услуга-Монтаж на климатици



фиг.20: Услуга-сервиз на климатици



фиг.21: Услуги-профилактики на климатици



#### фиг.22: Услуга-Безплатен оглед

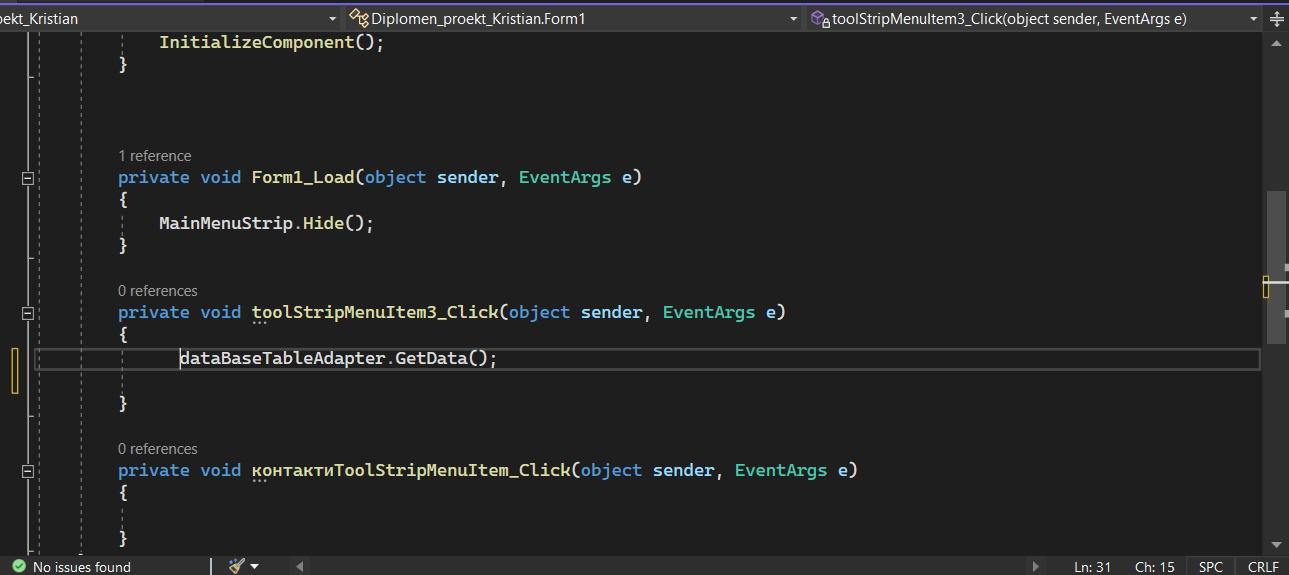
# 4.7 Реализация на база данни

**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 не противоречивост 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.

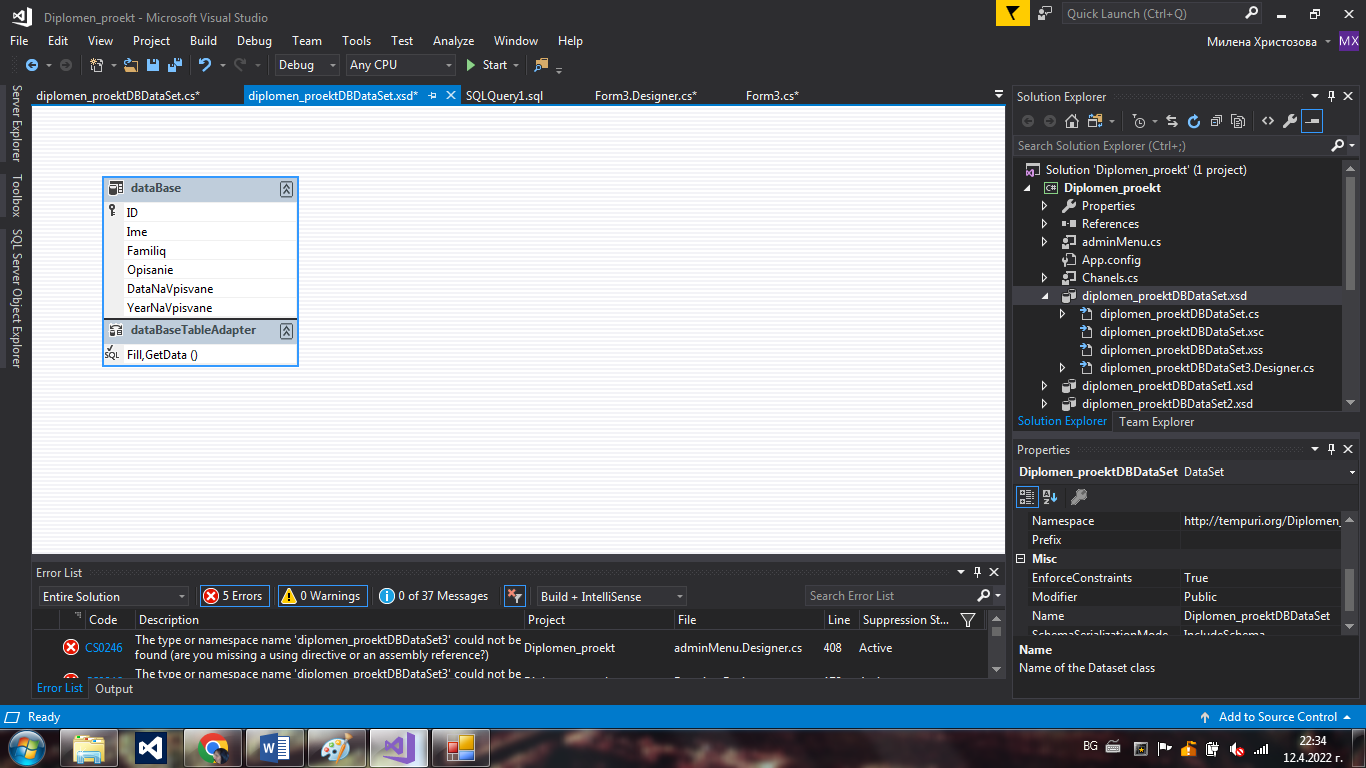


*фиг.*23*: Генериране на база данни*

**Логическа структура на базата данни**

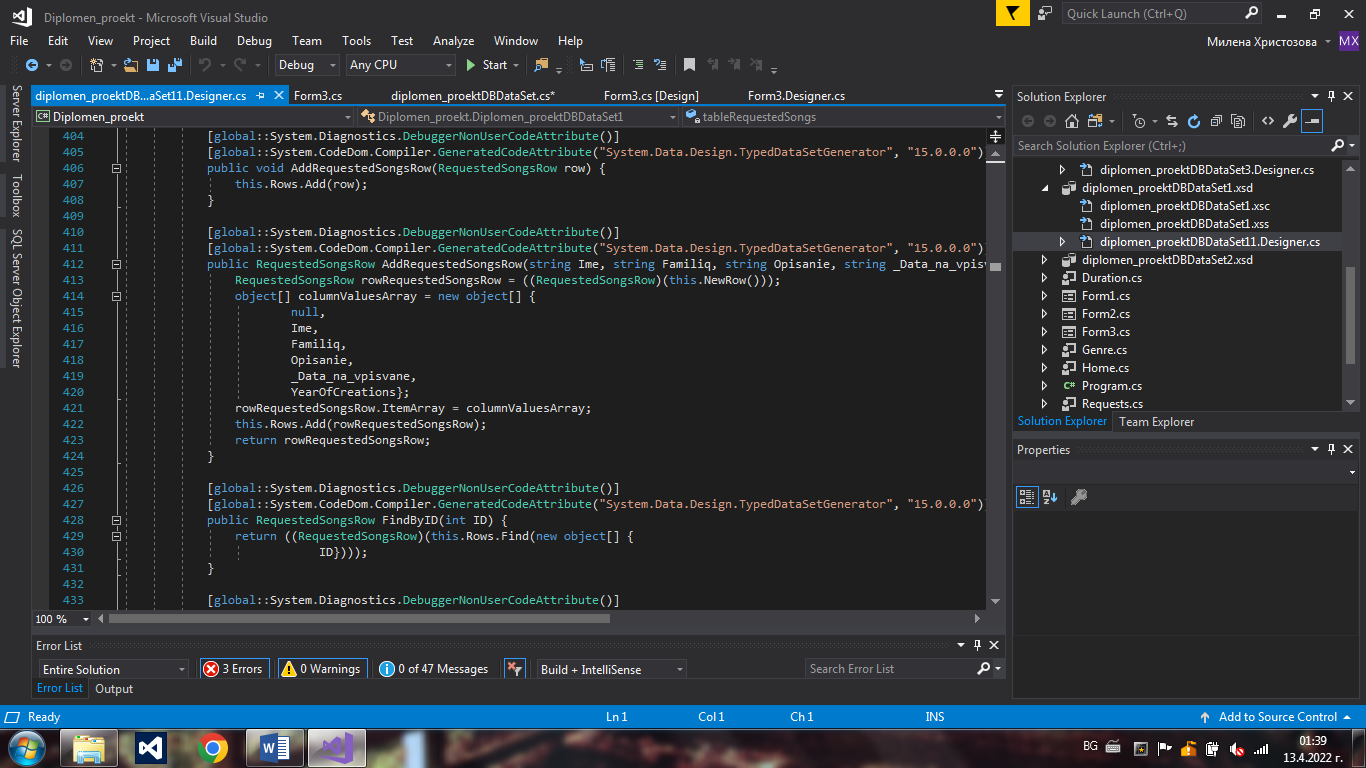
База данни е структурирано множество от постоянна информация, съхранявана от компютърна програма. Терминът постоянен означава, че данните продължават да съществуват и след прекратяването на програмата или на потребителската сесия, която ги е създала. Релационна база данни е съвкупност от такава информация, 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 де нормализирани 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.24: Структура на база данни*



*фиг.25: Код на администраторският панел на база данни*

# 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 във корпоративни фирми, които имат към няколко сайта с по 5 000 потребителя. Скрипта на програмата може да се адаптира и редактира с изискванията на клиента.

Съществува графичен интерфейс за възстановяване.

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

# 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.
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, Saarbrucken (2010).
5. <https://ikovachev.weebly.com/c.html>
6. <http://www.introprogramming.info/intro-csharp-book/-> книгата на Светлин Нанков, Веселин Колев и колектив "Въведение в програмирането със C#"
7. <https://i-clima.com/>
8. <https://bg-bg.facebook.com/iClimaltd/>