

# 

ДИПЛОМЕН ПРОЕКТ

**Тема:**

**РАЗРАБОТКА НА DESKTOP ЗА**

**МАГАЗИН ЗА ЦВЕТЯ**

**Дипломант: Консултант:**

Денис Минчев Танев Милена Христозова

№ 7, 12А клас

**Специалност:** Системно Програмиране

Сопот - 2024

# 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 Control ………………………………………………………………………...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 разрастването на бизнеса в индустрията на цветята и с увеличаването на конкуренцията, се появява нарастваща необходимост от софтуерни решения, които да оптимизират и улеснят бизнес процесите на цветарския магазин "Орхидея". Софтуерните платформи играят ключова роля в тази трансформация, като обединяват различни технологии и компоненти, които работят синхронно за подобряване на ефективността и производителността.

От изключителна важност е възможността за достъп до централизирана информация по всяко време и от всяко място, особено в бранша на цветята, където бързината и точността са от съществено значение. Платформата .NET, създадена от Microsoft, предоставя изключителни възможности за разработка на софтуерни приложения, които да отговарят на специфичните нужди на цветарския магазин "Орхидея".

В настоящата дипломна работа ще бъде представено как са приложени различни .NET технологии в създаването на информационна система, насочена към оптимизиране на бизнес процесите в цветарския магазин "Орхидея".

## Цел на дипломната работа

Настоящата дипломна работа се фокусира върху разработването на информационна система за цветарския магазин "Орхидея", която ще обхване координирането, регистрирането и избора на различни аспекти от дейността на магазина.

Цветарският магазин "Орхидея" е един от пионерите в индустрията и вече няколко години е на върха на своя бранш. Известен с високите си стандарти, "Орхидея" се превърна в емблематично място, символ на престиж, професионализъм и качество в сферата на цветята.

Основната цел на проекта е да се постигне моментално и прецизно регистриране и отчитане на всички дейности в магазина. Това е предизвикателство, което не може да бъде постигнато с традиционната ръчна обработка на информацията върху хартия или с използването на таблици в Microsoft Excel, както и със снимки разпечатани и поставени в албуми за показване на различни аспекти на бизнеса.

За да се постигне тази цел, ще бъде проектирана и реализирана релационна база данни с използване на Microsoft SQL Server за съхранение на данните. Информационната система ще включва две клиентски приложения за двата типа потребители - Администратори и Клиенти. Достъпът до базата данни от клиентските приложения ще се осъществява чрез DataGridView, което ще гарантира сигурност и ефективност при обмена на данни.

**2. Структура на дипломната работа**

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

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.

1. **Проучване**
2. **Използвани технологии и инструменти**

***Visual Studio***

Visual Studio е интегрирана среда за разработка (IDE), която е създадена от Microsoft. Тя предлага обширен комплект от инструменти за разработка на софтуер, включително езици като C#, C++, Visual Basic, F#, JavaScript, Python и други. Visual Studio позволява на програмистите да създават различни видове приложения, включително десктоп софтуер, уеб приложения, мобилни приложения, игри и много други.

Среди най-известните функции на Visual Studio са:

* .Редактор за код с мощни възможности за оформление, подчертаване на синтаксиса, автоматично довършване на кода и интегрирана поддръжка на Git за управление на версиите.
* Инструменти за дебъгване, които позволяват на програмистите да проследяват и отстраняват грешки в кода.
* Вградени шаблони за проекти, които улесняват създаването на нови приложения.
* Интегрирана поддръжка на различни фреймуърки и библиотеки за разработка на софтуер, като .NET Framework, .NET Core, ASP.NET, Xamarin и др.
* Възможности за сътрудничество и управление на екипи, включително интеграция с Microsoft Azure DevOps за управление на проекти, задачи и код.

Visual Studio се използва широко от програмисти по целия свят за разработка на различни видове софтуерни приложения и един от най-популярните и мощни инструменти за програмистите.

***C#***

C# (произнася се "C sharp") е език за програмиране, който е разработен от Microsoft през 2000 година. Той е създаден като част от платформата за разработка на софтуер .NET и е предназначен предимно за създаване на разнообразни видове софтуерни приложения, включително десктоп приложения, уеб приложения, мобилни приложения, уеб услуги и много други.

Езикът C# е силно типизиран, обектно-ориентиран и поддържа множество концепции от съвременното програмиране, като класове, наследяване, полиморфизъм, абстракция, инкапсулация и др. Той също така предлага широк спектър от вградени библиотеки и инструменти, които улесняват разработката на софтуер.

Някои от характеристиките на C#, които го правят популярен сред програмистите, включват:

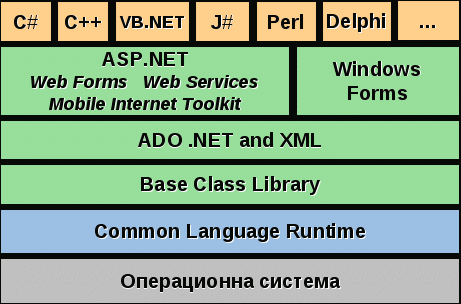
* Синтаксис подобен на C++ и Java: C# има синтаксис, който е подобен на този на други популярни езици като C++ и Java, което улеснява научаването на езика за програмисти, които вече са запознати с тези езици.
* Пълен достъп до платформата .NET: C# позволява програмистите да използват всички възможности на платформата .NET, включително богатата библиотека на класове (Framework Class Library), която предлага готови решения за много често срещани задачи.
* Мултиплатформеност: В последните години C# е развиван и адаптиран за използване на различни платформи, като например .NET Core и Xamarin, което позволява създаването на приложения, които работят на различни операционни системи като Windows, macOS и Linux, както и на мобилни устройства като Android и iOS.
* Интеграция с други технологии на Microsoft: C# се интегрира добре с други продукти и технологии на Microsoft, като например Visual Studio за разработка на софтуер и Microsoft Azure за облачни услуги.
* Активна общност и поддръжка: C# има голяма и активна общност от програмисти, които споделят знания, ресурси и инструменти, което прави езика подходящ за различни видове проекти и задачи.

В резултат на тези характеристики C# е станал един от най-популярните езици за програмиране в света и се използва широко от програмисти за разработка на различни видове софтуерни приложения.

***Microsoft .NET Framework***

.NET Framework e среда за разработка и изпълнение на приложения за .NET платформата. Тя предоставя програмен модел, библиотеки от типове и единна инфраструктура за разработка на приложения и поддържа различни езици за програмиране.

Приложенията, базирани на .NET Framework, се компилират до междинен код (на езика IL) и се изпълня­ват контролирано от средата за изпълнение на .NET Framework. Компилираният .NET код се нарича още управляван код и може да работи без да се прекомпилира върху различни платформи, за които има имплементация за .NET Framework (Windows, Linux, FreeBSD).

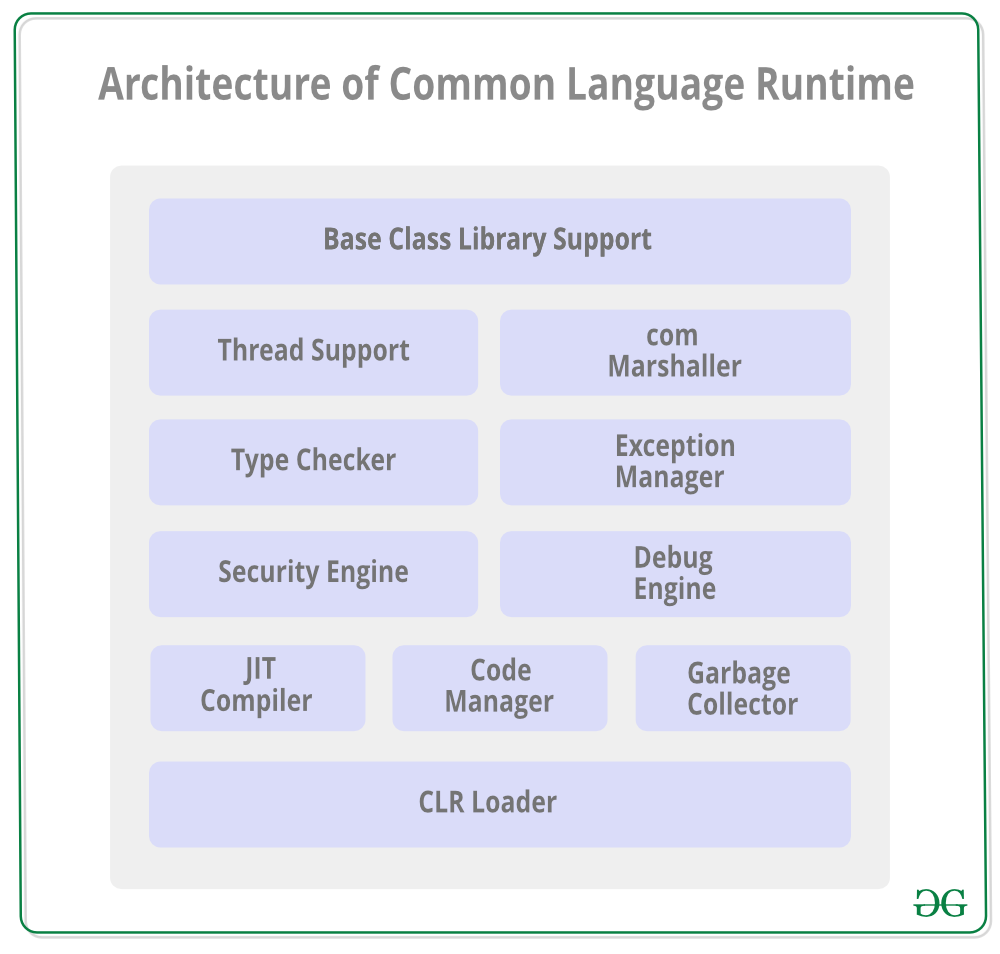


*Фиг1. Архитектура на .NET Framework*

***Компоненти на .NET Framework***

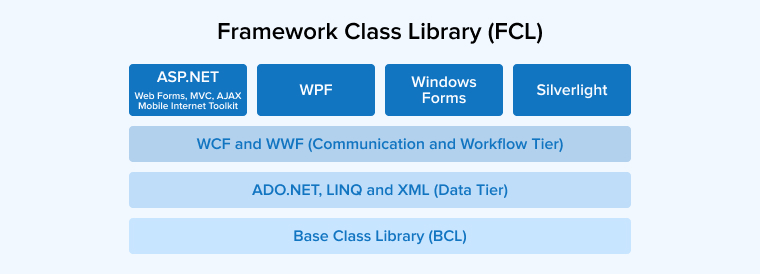
Можем да разделим .NET Framework на два основни компонента:

* **Common Language Runtime (CLR)** – средата, в която се изпълнява управляваният код на .NET приложенията. Представлява виртуална машина, която контролирано изпълнява .NET кода и осигурява раз­лични услуги, като управление на сигурността, управление на паметта и др.



*Фиг2. Архитектура на Common Language Runtime (CLR)*

* **Framework Class Library** **(FCL)**– представлява основната библио­тека от типове, които се използват при изграждането на .NET приложения. Съдържа основната функционалност за разработка, необходима за пове­чето приложения, като вход/изход, връзка с бази данни, работа с XML, изграждане на уеб приложения, използване на уеб услуги, изграждане на графичен потребителски интерфейс и др. Стандартните класове и типове от FCL можем да използваме нався­къде, където има инсталиран .NET Framework.



*Фиг3. Архитектура на Framework Class Library (FCL)*

***Microsoft Access***

Microsoft Access е релационна система за управление на бази данни (СУБД), разработена от Microsoft. Тя предоставя графичен потребителски интерфейс и мощни инструменти за създаване, редактиране и управление на бази данни без да е необходимо написване на код. Access позволява лесно създаване на форми, отчети и заявки за достъп до и манипулиране на данни в базата данни. Това го прави подходящо решение за малки и средни бизнеси, които имат нужда от управление на данни, без значителни технически умения.

1. **Проектиране**

## Модел на случаи на употреба- (Use Case) модел

**1.1 С*лучай на употреба - Идентифициране на Клиент (Log in)***

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

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

*Основен поток от действия*:

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

*Алтернативен поток 1*:

А1 1. Клиентът 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 клиента форма с име Клиентски профил с меню за възможни заявки за определена услуга”;

А1 4. Изход.

*Алтернативен поток 2*:

А2 1. Клиентът не е успешно идентифициран.

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

А2 3. Край.

**1.2. 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 клиент, 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.

**1.3. 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*: Клиента 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.

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

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

*Предусловия*: Клиентът 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. Клиентът преминава отново последователно по формите, в които е въвел информацията за дейността. Като във формите му се показват въведените или търсените от него данни.

**1.5. 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 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.

**1.6. С*лучай на употреба - Оffline работа с приложението (Offline)***

Клиентът 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. Клиентът 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.

**1.7. 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 желан букет или различна услуга.
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.

**1.8. С*лучай на употреба - Въвеждане на данни (Enter data)***

Клиентът 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, клиентът няма възможност да редактира типа на дейността, адреса на дестинацията

*Предусловия*: Клиентът е успешно идентифицира.

*Основен поток от действия*:

1. Клиентът въвежда данни.

*Алтернативен поток 1*:

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

А1 2. Въведените данни се валидират.

А1 3.Приложението показва следвавщата форма на клиента *Алтернативен поток 2*:

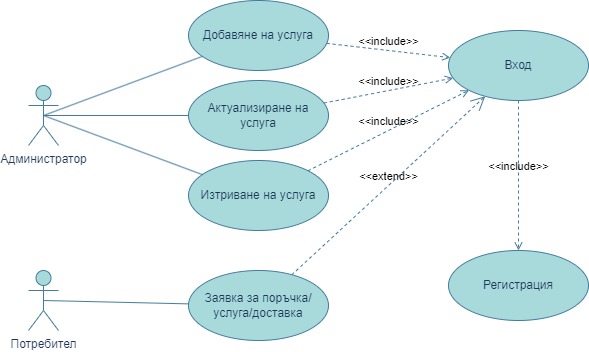
А2 1. Клиентът натиска бутона *„Затвори”*.

А2 2. Приложението показва предходната форма.

А2 3. Клиентът редактира вече въведените от него данни.

1. **Модел на случаите на употреба на приложението**

Определяме един актьор за приложението – Администратор, този потребител, който извършва всички взаимодействия с него.



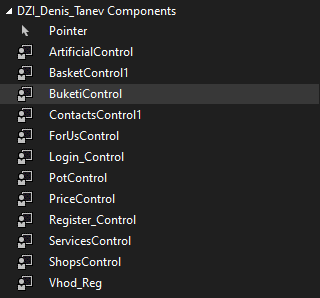
*Фиг4. Use Case Diagram*

#### IV. Реализация

#### 4.1. User Control

User controls в WinForms са компоненти за създаване на потребителски интерфейсни елементи, които съдържат комбинация от вградени контроли (като бутони, текстови полета, списъци и други) и функционалности. Те позволяват на разработчиците да групират свързани елементи на интерфейса и да ги използват повторно в различни части на приложението или дори в различни приложения.

Като по-сложни компоненти, user controls позволяват създаването на персонализирани интерфейсни елементи със специфични функционалности и поведения. Те могат да бъдат проектирани, съдържащи както визуални елементи, така и код за обработка на събития или изпълнение на допълнителна логика. Използването на user controls води до по-добра организация на кода и по-лесно преизползване на компонентите, като същевременно подобрява поддръжката и разширяемостта на приложението.

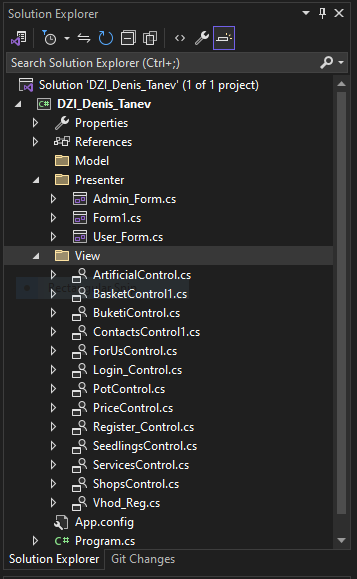
****

*Фиг1. UserControls*

#### 4.2. Динамична реализация

#### 4.3. Структура на приложението

#### При създаването на приложението първоначално създадох три папки отговарящи на името на изискването ми проекта да е MVP. В папката Model ще стои базата данни, в папката View поставих „User Controls“, а в най-важната папка Presenter поставих презентационния слой. Наименувах всеки елемент подобаващо за да ми е по-лесно при търсене и попълване на данните в класовете.

****

*Фиг2. МVP Модел*

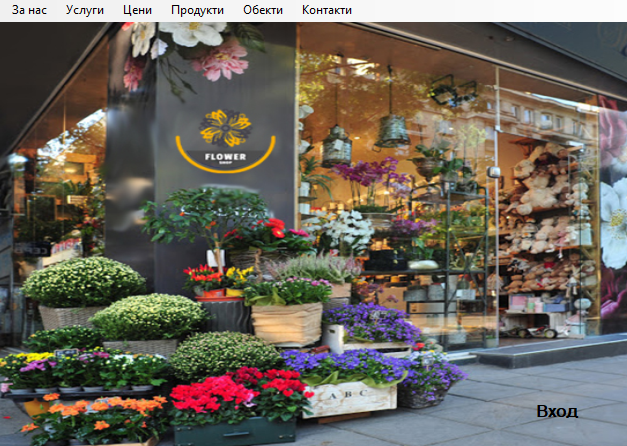
## 4.4. Имплементация

За създаването на приложението съм използвал Windows Forms и езика за програмиране C#. В глава I са представени потребителските изисквания към приложението. За осъществянето се използват главно контроли и форми,а за връзки с базата данни MSQL сървър.

### **4.5.1. 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.

В началната страница, Ви представям потребителския интерфейс. Тя е изградена от „MenuStrip“, три слоя с цел по-лесна работа и връзка с класовете при повикване. На следващата фигура 8 е показана визуално как изглжда началната страница на приложението.



*Фиг3. Потребителски интерфейс*

**4.5.2 Регистрация**

** Потребителският интерфейс за регистрация, представен на фигура 4, е съставен от Label-и(текстови полета поясняващи информацията, която трябва да се въведе в TextBox-a), TextBox-ове(Текстова кутия, в която се въвежда информацията посочна от Label-a) и Button(Бутон, който вкарва потребителя в базата данни). За да може успешно да бъде регистриран един потребител, трябва коректно да попълни следните полета „Име“, „Фамилия“, „Username“, „Парола“. След което като натисне бутона „РЕГИСТРИРАНЕ“ ще бъде успешно вкаран в базата.

*Фиг4. Регистрационен интерфейс*

**4.5.3 Вход Администратор/Потребител**

Потребителският интерфейс за вход, представен на фигура 5, е съставен от

два Label-a(посочващи нужната информация за вход), два TextBox-a(където да бъде въведена информацията) и Button(който усъшествява влизането при правилно въведени данни). За да се влезе в профила трябва да се напишат

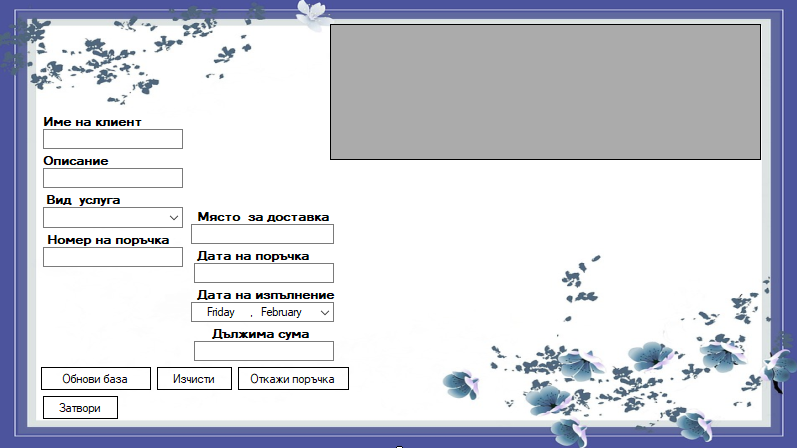
правилно „Usename-a” и „Паролата“.

*Фиг5. Интерфейс за вход на админ и регистриран потребител*

**4.5.4 Влезнал Админ**

При правилно въведени данни в полетата на фиг. 5 и активиране на бутона вход се активира интерфейса на фигура 6. При грешни данни излиза съобщение за грешно поле.

Тази форма се появява когато администратор успешно влезе в акаунта си. Интерфейса се състои от Label-и(текстови полета поясняващи информацията, която трябва да се въведе в TextBox-a), пет TextBox-a, четири бутона, календар, ComboBox, и DataGridView. В дясната част първоначално базата не е зареденеа. При попълване на полетата и активиране на бутона обнови база се зарежда първоначалната база от последното влизане и новите данни които са въведени с пореден номер. След натискането на бутона „Обнови база“ тя се пълни с информацията от горе посочените полета. При натискане на бутона „Изчисти“ всички полета се зануляват. При натискане на бутона „Откажи поръчка“ се изтрива маркирания ред в базата даннни, а именно самата поръчка. При натискане на бутона „Затвори“ прозореца се затваря и се връщаш обратно в началната страница.

**

*Фиг6. Влезнал Админ*