

**ПРОФИЛИРАНА ПРИРОДОМАТЕМАТИЧЕСКА ГИМНАЗИЯ
“АКАДЕМИК НИКОЛА ОБРЕШКОВ” – ГРАД РАЗГРАД**
Ул.Дъбрава 2, тел:0878684362, 0876207320, e-mail:pmgrz@abv.bg, http://www.pmgrz.net

**ПРОФЕСИЯ „ПРИЛОЖЕН ПРОГРАМИСТ“
СПЕЦИАЛНОСТ „ПРИЛОЖНО ПРОГРАМИРАНЕ“**

ДИПЛОМЕН ПРОЕКТ

за придобиване трета степен професионална квалификация

ТЕМА: Проект на система за управление на хотел

Ученик: Калоян Красимиров Ганев

/име, презиме, фамилия/

Клас: 12 г

**GitHub-хранилище: koko462/Hotel-Management: Дипломен проект по
програмиране - Система за управление на хотел**

Ръководител-консултант: Венцеслав Кочанов

/име и фамилия/

Разград

2025 година

**ПРОФИЛИРАНА ПРИРОДОМАТЕМАТИЧЕСКА ГИМНАЗИЯ „АКАД. Н.
ОБРЕШКОВ“ –гр. Разград**

**ЗАДАНИЕ ЗА ДИПЛОМЕН ПРОЕКТ ДЪРЖАВЕН ИЗПИТ ЗА
ПРИДОБИВАНЕ НА ТРЕТА СТЕПЕН НА ПРОФЕСИОНАЛНА
КВАЛИФИКАЦИЯ**

По професия кодХ481030 „Приложен програмист“

Специалност код 4810301 „Приложно програмиране“

Тема: Проект за система за управление на хотел

Изисквания за разработката на дипломния проект (проучване на литература, съдържание, оформяне, указания за изпълнение, инструкции, указания):

Системата трябва да включва:

1. Управление на стаи:

- Добавяне на нови стаи с характеристики (тип, цена, брой легла, статус – свободна/заета).
- Преглед на наличните стаи по критерии (тип, цена, статус).

2. Управление на резервации:

- Създаване на резервации (клиент, стая, дати на престой).
- Обновяване на информация за резервации (напр. промяна на дати или анулиране).

3. Управление на клиенти:

- Регистрация на нови клиенти с лични данни (име, телефон, имейл).
- Преглед и актуализация на клиентска информация.

4. Управление на персонал

- Създаване и управление на профили за персонал (име, позиция, работно време).
- Актуализация на данни за служителите

5. Системата трябва да може да се администрира с администраторски панел

СЪДЪРЖАНИЕ

1. Увод	1
2. Теоретична част	3
2.1. Въведение	
2.2. Основни понятия и дефиниции	
2.3. Теоретични модели и подходи	
2.4. Използвани технологии	
2.5. Обвързване с настоящия проект	
3. Аналитична част	6
3.1. Описание на обекта на изследване	
3.2. Анализ на проблемите без система	
3.3. Практическо приложение на разработената система	
3.3.1. Управление на стаи	
3.3.2. Управление на резервации и клиенти	
3.3.3. Управление на персонал	
3.3.4. Администриране на системата с администраторски панел	
3.3.5. Структура на базата данни myHotel (накратко)	
3.3.6. Връзки между таблиците	
4. Проектна част	22
4.1. Архитектура на системата	
4.2. Връзка между базата данни myHotel и C#	

4.3. Модул "Управление на стаи"	
4.4. Модул "Управление на резервации и клиенти"	
4.5. Модул "Управление на персонал"	
4.6. Модул "Администраторски панел"	
5. Заключение	54
6. Използвана литература	55

1. УВОД

С развитието на туристическата индустрия и увеличаващите се изисквания на клиентите за бързина, удобство и качество на обслужване, хотелският бизнес е изправен пред предизвикателството да оптимизира процесите си чрез внедряване на съвременни информационни технологии. В условията на нарастваща конкуренция и динамична пазарна среда, ефективното управление на ресурсите в един хотел – стаи, резервации, клиенти и персонал – става решаващ фактор за постигане на устойчив успех. Ръчната обработка на информация, използването на оstarели системи или липсата на автоматизация водят до сериозни рискове като загуба на данни, допускане на грешки, забавяне на обслужването и недоволство сред клиентите.

В отговор на тези предизвикателства, настоящият дипломен проект е насочен към създаване на **интегрирана система за управление на хотел**, която да осигури централизиран достъп до данните, автоматизиране на основните процеси и улесняване на работата на персонала. Чрез дигитализацията на операциите хотелът може не само да подобри вътрешната си организация, но и да повиши качеството на услугите си, което е ключов фактор за привличане и задържане на клиенти.

Целта на дипломния проект е разработването на софтуерна система, която позволява ефективно управление на стаите, резервациите, клиентите и персонала на един хотел чрез лесен за използване графичен потребителски интерфейс. Системата трябва да улеснява ежедневните дейности на хотелския персонал, да минимизира вероятността от човешки грешки и да осигурява бърза и сигурна обработка на информацията.

Задачите на дипломната работа включват:

- Проектиране на цялостната архитектура на приложението, съобразена със спецификите на хотелския бизнес.

- Разработка на отделните модули за управление на стаи, резервации, клиенти и персонал.
- Осигуряване на административен панел за управление на достъпа и конфигурацията на системата.
- Създаване на база данни за надеждно съхранение на необходимата информация.
- Тестване на системата в различни сценарии, симулиращи реална работа в хотелска среда.

Използваната методика за реализиране на проекта е базирана на принципите на обектно-ориентираното програмиране, като е избран езикът C# заради своята стабилност, гъвкавост и богати възможности за изграждане на настолни приложения чрез Windows Forms. За съхраняване и обработка на данните се използва релационната база данни Microsoft SQL Server (MSSQL), която осигурява висока надеждност, мащабируемост и добра интеграция с C# приложения.

Ограниченията и информационната обезпеченост на проекта са свързани с това, че системата е проектирана за управление на малки и средни по размер хотели и не включва интеграция с външни платформи за онлайн резервации или мобилни приложения. За целите на разработката и тестването се използват фиктивни (примерни) данни, като реални клиентски или финансови данни не се обработват.

В заключение, настоящият проект цели не само техническото създаване на софтуерна система, но и демонстрацията на важността на автоматизацията за оптимизацията на хотелските процеси. Чрез правилно проектираната и внедрена информационна система, хотелите могат значително да повишат своята ефективност и да подобрят обслужването на клиентите си в условията на съвременната конкурентна среда.

2. ТЕОРЕТИЧНА ЧАСТ

2.1. Въведение

В условията на съвременната икономика информационните технологии играят решаваща роля за ефективното управление на различни бизнес процеси. В сферата на хотелиерството използването на автоматизирани системи за управление на хотели е особено важно за подобряване на качеството на обслужване, оптимизация на вътрешните процеси и увеличаване на конкурентоспособността. Основната цел на информационните системи в хотелите е да улеснят управлението на стаи, резервации, клиенти и персонал, като същевременно осигуряват лесен достъп до данни и висока степен на точност и сигурност.

2.2. Основни понятия и дефиниции

Информационна система е съвкупност от софтуер, хардуер, бази данни, мрежови технологии и хора, които съвместно работят за събиране, обработка, съхранение и разпространение на информация.

Софтуер за управление на хотел представлява специализирано приложение, което автоматизира ежедневните дейности като резервации, настаняване, освобождаване на стаи, фактуриране и управление на клиентски данни. **База данни** е организирана колекция от данни, които могат лесно да бъдат достъпвани, управлявани и актуализирани. В хотелиерството базите данни съхраняват информация за стаи, клиенти, резервации и служители.

2.3. Теоретични модели и подходи

При разработването на информационни системи се използват различни архитектурни модели, които улесняват организацията на софтуерния код и данните. Един от най-често използваните е **трислойната архитектура**, състояща се от:

- **Презентационен слой (Presentation Layer)** – осигурява комуникация с потребителя чрез графичен интерфейс.

- **Бизнес логика (Business Logic Layer)** – съдържа правилата за работа на приложението.
- **Слой за достъп до данни (Data Access Layer)** – осъществява комуникация с базата данни.

Този подход осигурява по-лесна поддръжка, разширяемост и разделяне на отговорностите в системата.

В процеса на разработване на софтуерни приложения се следват няколко основни етапа:

- Анализ на изискванията
- Проектиране на системата
- Имплементация (кодиране)
- Тестване
- Внедряване и поддръжка

2.4. Използвани технологии

В проекта се използват следните основни технологии:

- **C#** – обектно-ориентиран език за програмиране, разработен от Microsoft, който е особено подходящ за създаване на настолни приложения (Desktop Applications) чрез Windows Forms.
- **Microsoft SQL Server (MSSQL)** – система за управление на релационни бази данни, използвана за съхранение и обработка на информацията.
- **Windows Forms** – технология за създаване на графичен потребителски интерфейс в рамките на C#, която улеснява визуалното разработване на приложения.

Тези технологии позволяват бърза и ефективна разработка на стабилни системи, като същевременно предоставят възможности за разширение и интеграция в бъдеще.

2.5. Обвързване с настоящия проект

Изборът на C#, MSSQL и Windows Forms е мотивиран от тяхната стабилност, широката им поддръжка, лесното интегриране между тях и наличието на богата документация. С тези технологии се създава удобна и ефективна система за управление на хотел, отговаряща на съвременните изисквания за качество и надеждност.

3. АНАЛИТИЧНА ЧАСТ

3.1. Описание на обекта на изследване

Хотелите са динамична среда, в която управлението на резервации, клиенти, наличности и персонал е критично за успешната дейност. С нарастващия брой клиенти и конкуренцията на пазара, нуждата от автоматизирани системи за управление на хотел става все по-важна. Ръчните методи на управление водят до забавяния, грешки и нездадоволително клиентско обслужване.

Разработената **Система за управление на хотел** цели да реши тези проблеми чрез ефективна, бърза и лесна за използване информационна система.

3.2. Анализ на проблемите без система

При отсъствие на специализирана система управлението на хотел преминава през редица трудности:

- Риск от двойни или объркани резервации.
- Загуба на информация за клиенти и престои.
- Трудности при следенето на свободните стаи.
- Липса на бърза справка за заетостта или за персонала.
- По-бавна обработка на резервации и освобождаване на стаи.

Това води до по-ниско качество на услугата и загуба на клиенти.

3.3. Практическо приложение на разработената система

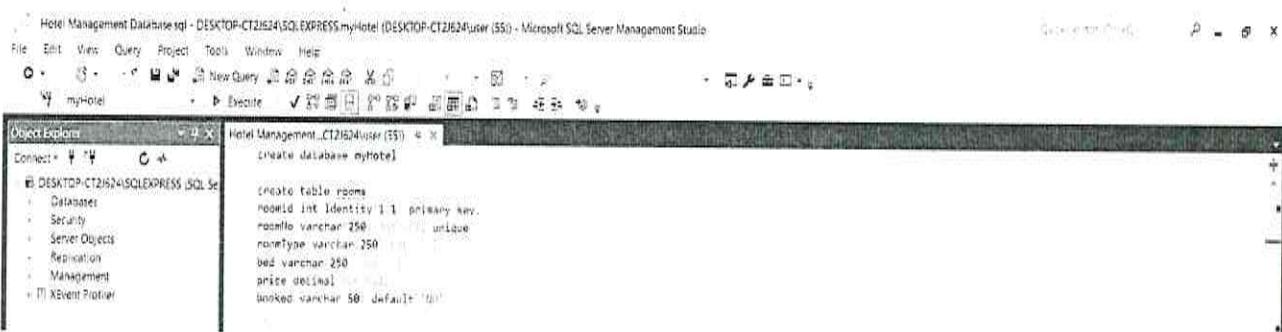
Системата е разработена на **C# с Windows Forms** и използва **Microsoft SQL Server** за съхранение на данни. Основните модули включват:

3.3.1. Управление на стаи

Формата за стаи позволява:

- Добавяне на нова стая със зададени характеристики: номер, тип (единична, двойна, луксозна), цена, брой легла и статус (свободна/заета).
- Преглед на всички стаи с възможност за филтриране по статус, тип или ценови диапазон.
- Обновяване на статуса на стаята при настаняване или освобождаване на клиент.

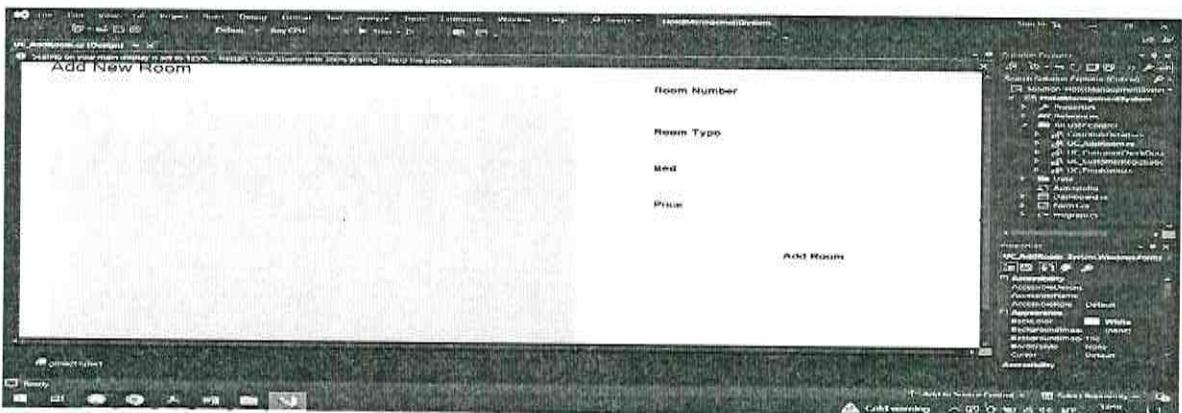
Фигура 1 - База данни: Таблица Rooms съхранява цялата информация за стаите.



```

File Edit View Query Project Tools Window Help
New Query New Query Results Object Explorer Task List
myHotel Execute
Object Explorer
Connect > myHotel
Databases Security Server Objects Replication Management XEvent Profiler
Hotel Management...CT21624User (35) 4 x
CREATE DATABASE myHotel;
CREATE TABLE rooms
(
    roomid INT IDENTITY(1,1) PRIMARY KEY,
    roomno VARCHAR(256) UNIQUE,
    roomtype VARCHAR(256),
    bed VARCHAR(256),
    price DECIMAL,
    booked VARCHAR(50) DEFAULT('No')
)
  
```

Фигура 2



Фигура 3

The screenshot shows the 'Tables' section of the 'myHotel' database in SQL Server Management Studio. A new table named 'Rooms' is being created. The table has six columns: 'roomid' (primary key, int, not null), 'roomNo' (varchar(250), unique, not null), 'roomType' (varchar(250), not null), 'bed' (varchar(250), not null), 'price' (decimal(18, 2), not null), and 'booked' (varchar(50), not null). The 'Allow Nulls' checkboxes for 'price' and 'booked' are checked.

Column Name	Data Type	Allow Nulls
roomid	int	<input type="checkbox"/>
roomNo	varchar(250)	<input type="checkbox"/>
roomType	varchar(250)	<input type="checkbox"/>
bed	varchar(250)	<input type="checkbox"/>
price	decimal(18, 2)	<input checked="" type="checkbox"/>
booked	varchar(50)	<input checked="" type="checkbox"/>

Преди да се създаде таблицата Rooms, първо трябва да има база данни. В конкретния случай, базата данни се нарича myHotel. Базата данни е като „папка“, която държи различни таблици с данни за хотела. След това се изгражда таблицата Rooms, която съдържа уникален номер за всяка стая (roomid), номер на стаята (roomNo), тип на стаята (roomType), вид на леглото (bed), цена за нощувка в стаята (price) и статус дали стаята е резервирана (booked). roomid е първичен ключ (primary key), което гарантира, че има уникална стойност също така се увеличава автоматично с 1 при всяка нова стая, тъй като притежава (Identity(1,1)). roomNo не може да е празно (not null) и е уникално (unique) като типът му е текст до 250 знака (varchar(250)). roomType не може да е празно (not null) и също е текст до 250 знака (varchar(250)). bed не може да е празно (not null) и е текст до 250 знака (varchar(250)). price е дробно число с две цифри зад запетаята (decimal(18,2)) и не може да е празно (not null). booked по подразбиране (default) е „NO“ (не е резервирана) и е текст с дължина до 50 знака (varchar(50)).

Формата Add New Room (UC_AddRoom.cs) съдържа разнообразни контроли за изпълняване на функцията на системата за добавянето на стаи. Тя съдържа четири етикета (labels) с текстове Add New Room, Room Number, Room Type, Bed и Price. Във формичката се намира и решетка с изобразяването на данни от базата myHotel (dataGridView1), оцветена със сив цвят. В нея се намират и две текстови полета (txtRoomNo, txtPrice), които се намират под съответните етикети с текстове

RoomNumber и Price. Разположени са и две комбинирани кутии (txtType, txtBed) под съответните етикети с текстове RoomType и Bed. Под txtPrice се намира бутона за добавяне на нова стая с текст Add Room (btnAddRoom).

3.3.2. Управление на резервации и клиенти

Формите за резервации и клиенти позволяват:

- Създаване на нова резервация чрез избор на клиент, стая и период на престой (дата на настаняване и дата на напускане).
- Обновяване на съществуващи резервации – промяна на дати или стая.
- Регистрация на нови клиенти с данни като име, телефонен номер и имейл.
- Преглед на списък с клиенти.

Фигура 4 - База данни: Таблица Customers съхранява пълната клиентска информация.

```
create table customer(
    cid int Identity(1,1) primary key,
    cname varchar(250) not null,
    mobile bigint not null,
    nationality varchar(250) not null,
    gender varchar(50) not null,
    dob varchar(50) not null,
    idproof varchar(250) not null,
    email varchar(350) not null,
    checkin varchar(250) not null,
    checkout varchar(250),
    chekout varchar(250) not null default 'NO',
    roomid int not null,
    foreign key (roomid) references rooms(roomid)
);
```

Фигура 5

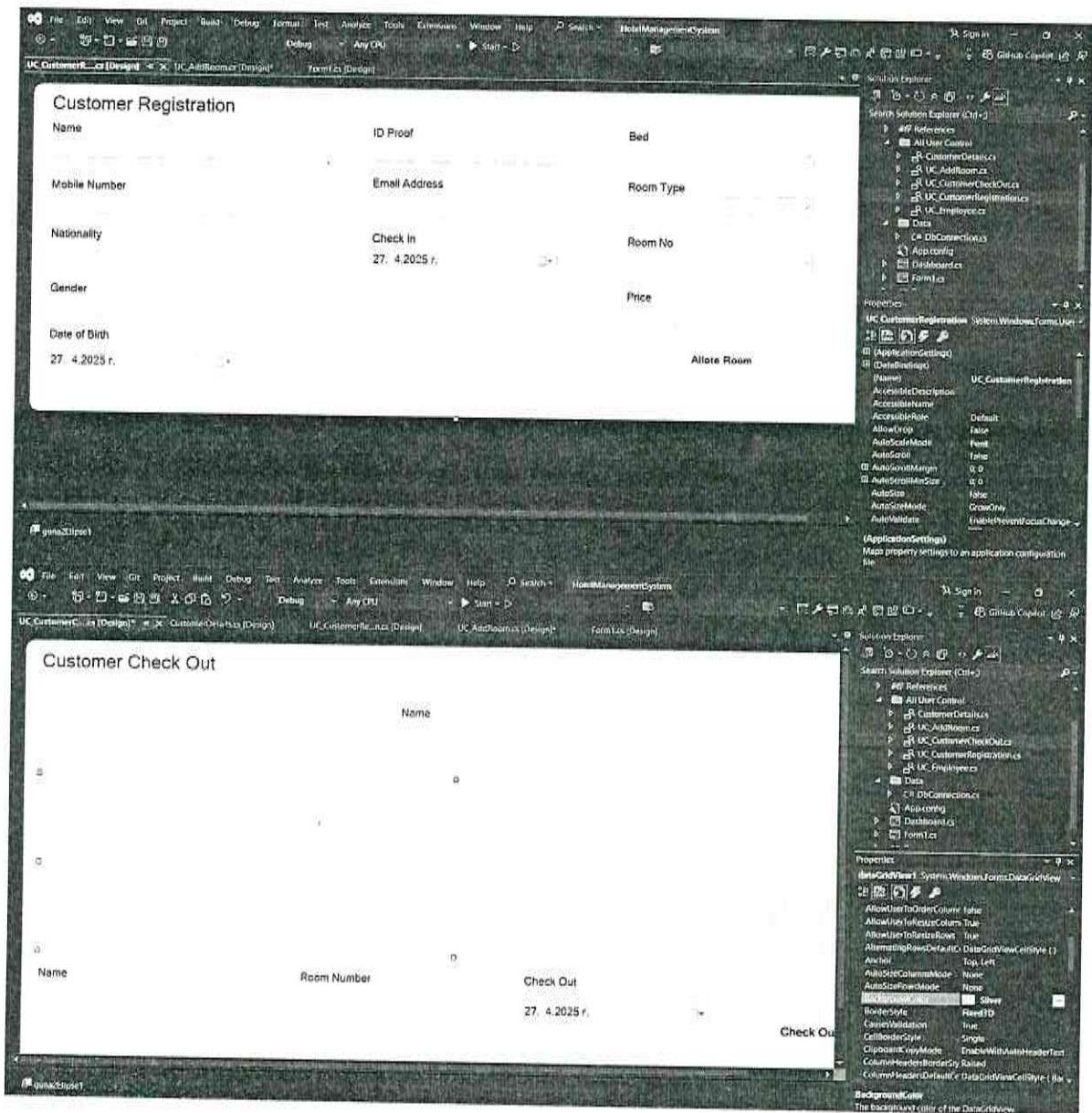
	Column Name	Data Type	Allow Nulls
cid	int	<input type="checkbox"/>	
cname	varchar(250)	<input type="checkbox"/>	
mobile	bigint	<input type="checkbox"/>	
nationality	varchar(250)	<input type="checkbox"/>	
gender	varchar(50)	<input type="checkbox"/>	
dob	varchar(50)	<input type="checkbox"/>	
idproof	varchar(250)	<input type="checkbox"/>	
email	varchar(350)	<input type="checkbox"/>	
checkin	varchar(250)	<input type="checkbox"/>	
checkout	varchar(250)	<input checked="" type="checkbox"/>	
chekout	varchar(250)	<input type="checkbox"/>	
roomid	int	<input type="checkbox"/>	

Таблицата customer се изгражда чрез команда `create table customer ()`. Тя съдържа уникален номер на всеки клиент (cid), име на клиента (cname), телефонен номер (mobile), националност (nationality), пол на клиента (gender), дата на раждане (dob), вид документ за самоличност (idproof), имейл на клиента (email), дата на настаняване (checkin), дата на напускане (checkout), поле за проверка за това дали клиента е напуснал хотела (chekout) и номер на съществуваща стая, която клиент може да заеме (roomid). cid е първичен ключ (primary key) и има уникална стойност, като започва от 1 и се увеличава автоматично при всяко ново добавяне (Identity(1,1)). cname не може да бъде празно (not null). mobile е bigint, тъй като телефонните номера са дълги числа. nationality е текст с дължина до 250 знака(varchar(250)) и не може да е празен (not null). gender е текст с дължина до 50 знака (varchar(50)) и не може да е празен (not null). dob не може да е празен (not null) и е текст с дължина до 250 знака (varchar(250)), тъй като между цифрите има точка и заради това е текст. idproof е текст с дължина до 250 знака (varchar(250)) и не може да е празен (not null). email е текст с дължина до 350 знака (varchar(350)) и не може да е празен (not null). checkin и

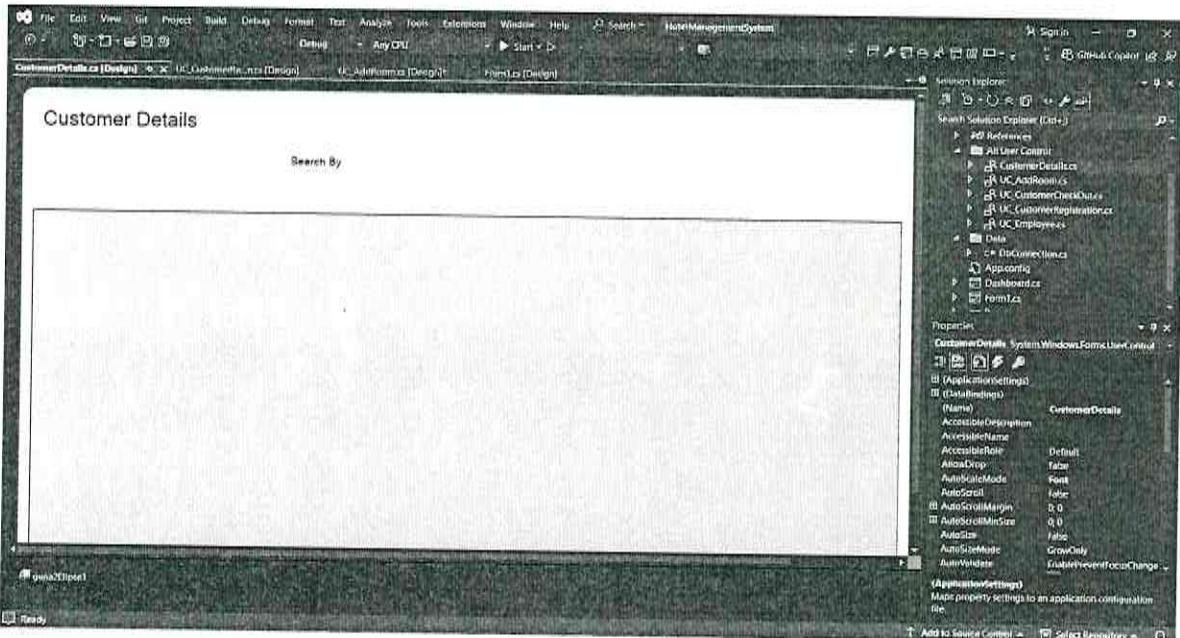
checkout са текстове с дължина до 250 знака (varchar(250)) и не трябва да са празни (not null). checkout по подразбиране (default) е „NO“. roomid е цяло число (int) и не трябва да е празно. Редът foreign key (roomid) references rooms (roomid) създава външен ключ към колоната roomid в таблицата rooms.

Дизайнът на формите за управлението на резервации и клиенти изглеждат по следния начин:

Фигура 6



Фигура 7



Имената на формите са UC_CustomerRegistration.cs (оцветена в червен цвят), UC_Customer CheckOut.cs (оцветена в бял цвят) и CustomerDetails.cs (оцветена в зелен цвят). UC_CustomerRegistration съдържа контроли, чрез които се осъществява управлението на клиенти и настройване на дата на настаняване, както и филтриране на стаите по критерии (тип, цена, статус). Те са 13 етикета (labels) с текстове Customer Registration, Name, Mobile Number, Nationality, Gender, Date Of Birth, ID Proof, Email Address, Check In, Bed, Room Type, RoomNo, Price. В нея се намира и бутон с текст Allote Room. Под съответните етикети се намират текстови полета (txtName, txtMobile, txtNationality, txtIDProof, txtEmail и txtPrice), комбинирани кутии (txtGender, txtBed, txtType, txtRoomNo) и инструменти за избор на дата и час (txtDob, txtCheckIn). UC_CustomerCheckOut.cs съдържа контроли, чрез които се извършва напускането на клиента от хотела. Те са 5 етикета с текстове (Customer Check Out, Name, Name, Room Number, Check Out), решетка с изобразяването на данни от базата myHotel (dataGridView1), инструмент за избор на дата и час (txtCheckOut), три текстови полета за име и номер на стаята, намиращи се под съответните етикети. Във формата се намира и бутон с текст Check Out, оцветен със синьо. CustomerDetails.cs съдържа контроли, чрез които се показват данните за клиентите от базата. Те са два

етикета с текстове Customer Details и Search By, комбинирана кутия (txtSearchBy) и решетка с изобразяването на данни от базата myHotel (dataGridView1).

3.3.3. Управление на персонал

Формата за персонал осигурява:

- Добавяне на нови служители с данни: име, длъжност (например receptionist, камериерка, мениджър), работно време.
- Изтриване и преглед на профили на служители.

Фигура 8 - База данни: Таблица Employee.

Hotel Management..CT2J624\user (55) ↗ ✎

```
create table employee
(
    eid int Identity(1,1) primary key,
    ename varchar(250) not null,
    worktime varchar(250) not null,
    position varchar(250) not null,
    emailid varchar(250) not null,
    username varchar(150) not null,
    pass varchar(150) not null
);
```

Фигура 9

Column Name	Data Type	Allow Nulls
eid	int	<input type="checkbox"/>
ename	varchar(250)	<input type="checkbox"/>
worktime	varchar(250)	<input type="checkbox"/>
position	varchar(250)	<input type="checkbox"/>
emailid	varchar(250)	<input type="checkbox"/>
username	varchar(150)	<input type="checkbox"/>
pass	varchar(150)	<input type="checkbox"/>

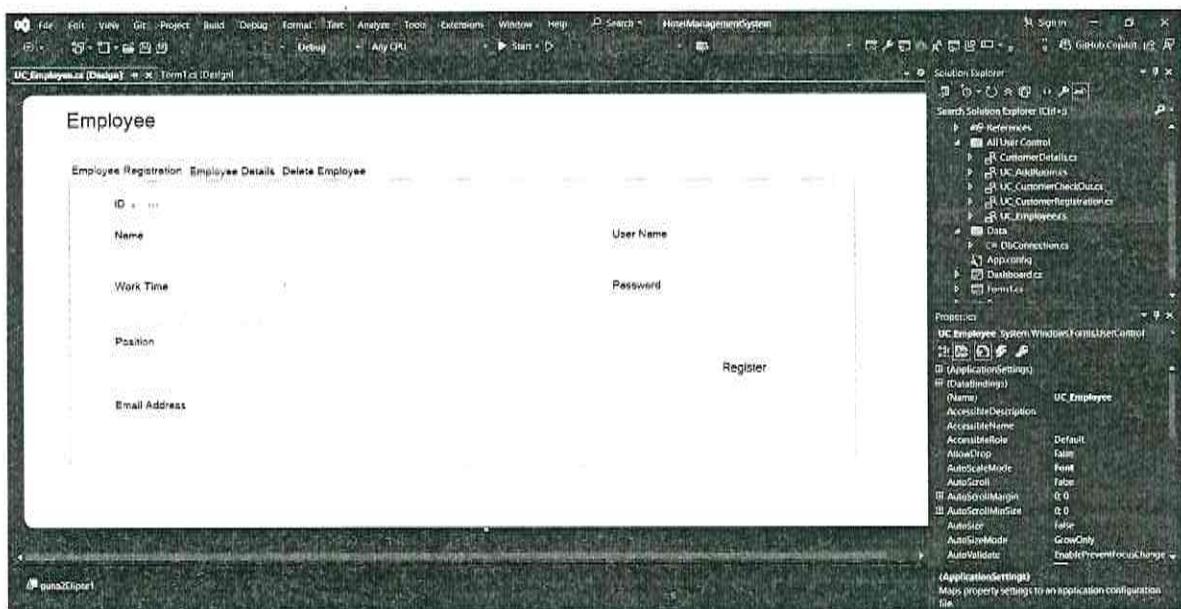
Следният SQL код създава таблица с име `employee` в базата данни. Таблицата е предназначена за съхраняване на информация за служителите на хотела. **Обяснение на структурата:**

- `eid` — Уникален идентификатор на служителя. Число, което автоматично се увеличава (`Identity(1,1)`), като започва от 1. Това поле е дефинирано като **първичен ключ**.
- `ename` — Име на служителя. Тип `varchar(250)`, като стойността е задължителна (`not null`).
- `worktime` — Работно време на служителя (например "08:00-16:00", "Нощна смяна"). Полето е от тип `varchar(250)` и е задължително.
- `position` — Дължност на служителя (например "Рецепционист", "Управител", "Камериерка"). Тип `varchar(250)`, задължително поле.
- `emailid` — Имейл адрес на служителя. Тип `varchar(250)`, задължително поле.
- `username` — Потребителско име за достъп до системата. Тип `varchar(150)`, задължително поле.
- `pass` — Парола за достъп до системата. Тип `varchar(150)`, задължително поле.

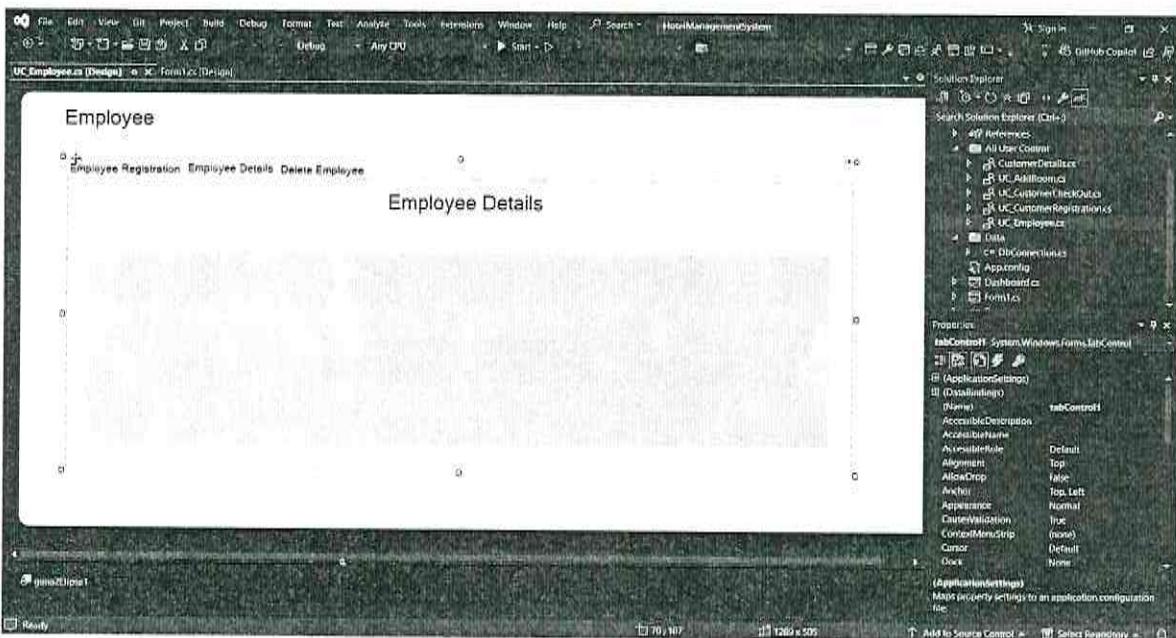
Цел на таблицата: Таблицата employee осигурява съхранение на основната информация за служителите, както и данни за техните потребителски имена и пароли, които могат да се използват за вход в системата за управление на хотела.

Дизайнът на формата за персонал изглежда по следния начин:

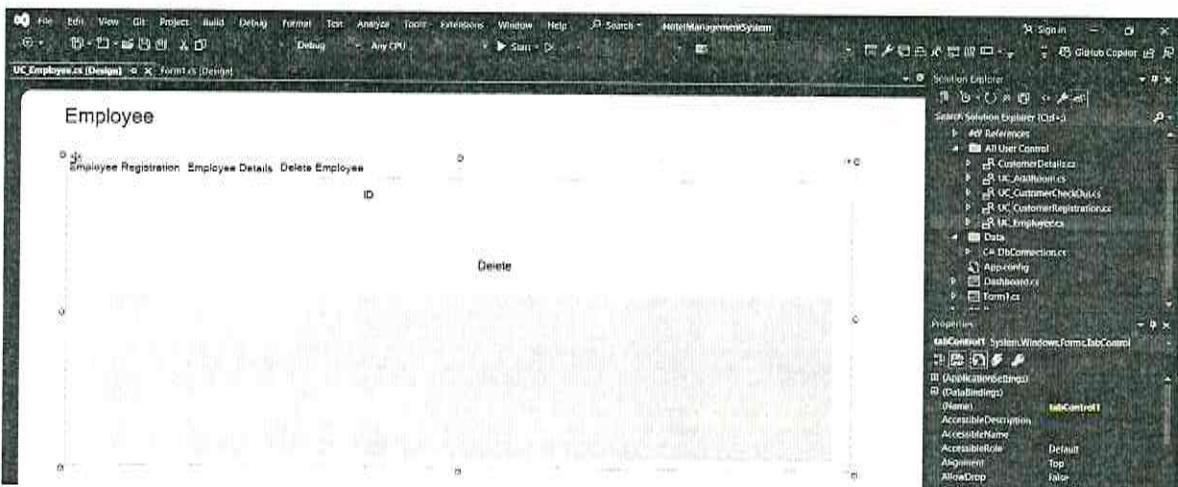
Фигура 10



Фигура 11



Фигура 12



Формата за персонала се нарича UC_Employee.cs. Тя съдържа етикет (label) с текст Employee и управление на раздели (tabControl1). Той съдържа три прозореца: Employee Registration, Employee Details и Delete Employee. Всеки прозорец съдържа необходимите контроли за извършване на съответните функции. Прозореца Employee Registration съдържа 7 етикета с текстове (ID, Name, Work Time, Position, Email Address, User Name, Password) и бутон с текст Register. До етикета с ID се намира друг етикет, който актуализира стойността на ID при добавянето на човек от

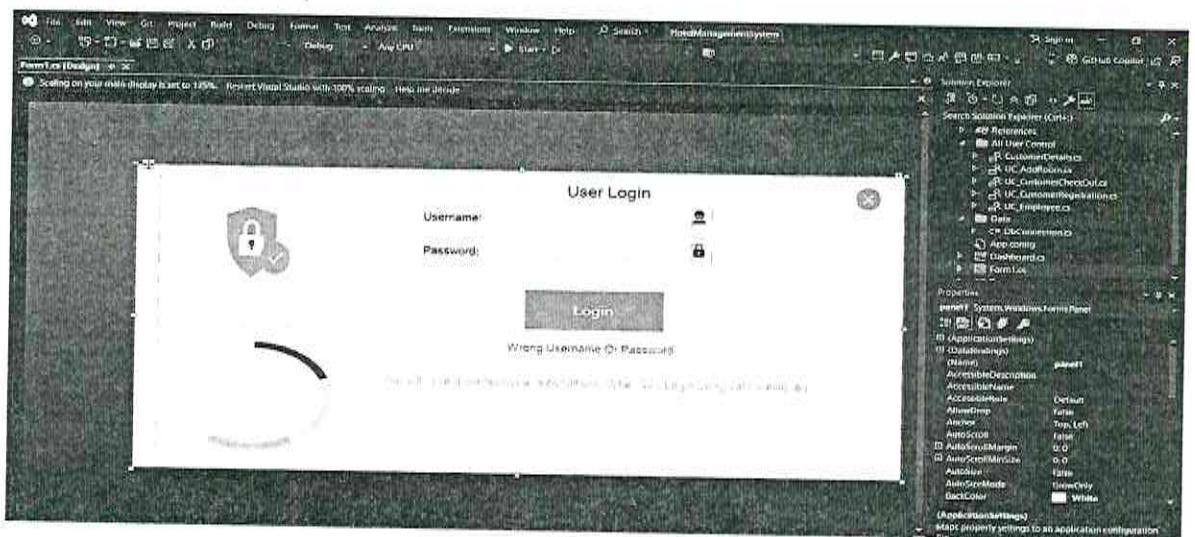
персонала. Под останалите етикети се намират текстови полета (txtName, txtWorkTime, txtEmail, txtUsername и txtPassword) и комбинирана кутия с длъжността на човек от персонала (txtPosition). Прозорецът Employee Details съдържа етикет (label) с текст Employee Details и решетка с показването на данните от базата myHotel (dataGridView1). Прозорецът Delete Employee съдържа етикет с текст (ID), текстова кутия под етикета, бутон с текст Delete и решетка за изобразяване на данните от базата myHotel (dataGridView2).

3.3.4. Администриране на системата с администраторски панел

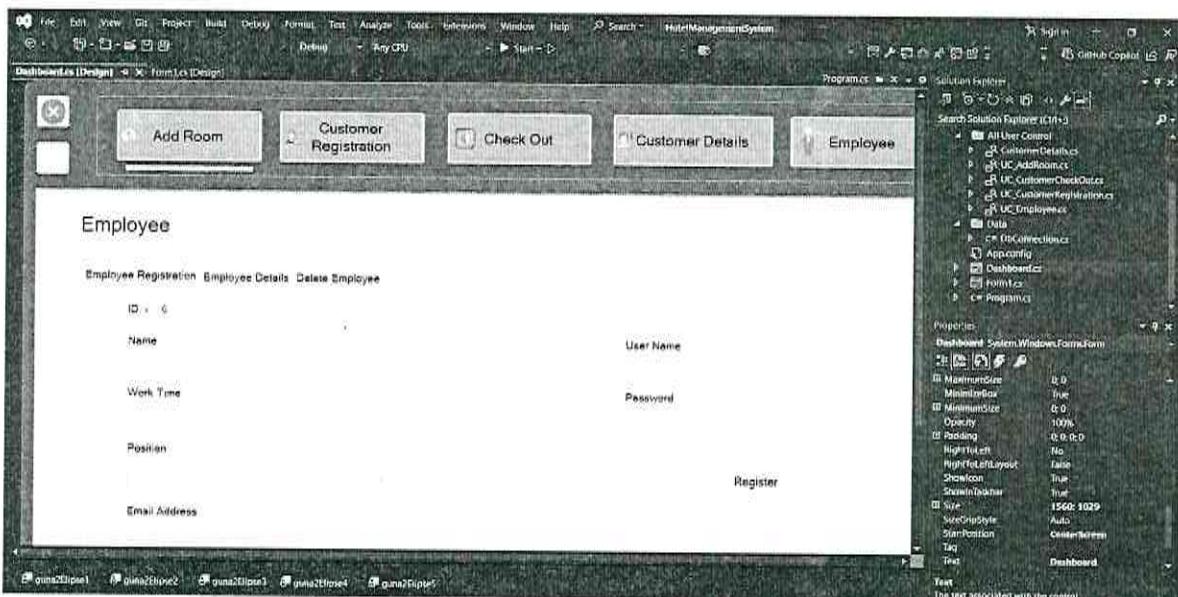
Login формата (форма за администриране на системата с администраторски панел) позволява:

- Влизане на администратора в системата чрез потребителско име и парола.
- Пълен контрол над функционалностите на системата за управление на хотела.

Фигура 13



Фигура 14



Login формата се нарича Form1.cs. Тя се състои от един панел, оцветен в бял цвят. В панела се намират 4 етикета с текстове (User Login, Username, Password и Wrong Username Or Password, оцветен в червен цвят). Освен това тя съдържа още и два бутона (Login в зелен цвят и бутон, който при натискане затваря приложението, означен с червено, разположен горе вдясно). След успешното регистриране на администратора, се появява друга форма, наречена DashBoard.cs. Тази форма е свързана с другите форми (CustomerDetails.cs, UC_AddRoom.cs, UC_CustomerCheckOut.cs, UC_CustomerRegistration.cs и UC_Employee.cs) чрез натискане на бутоните с текстове (Add Room, Customer Registration, Check Out, Customer Details и Employee). При натискането на всеки бутон изскочат съответните UserControls, които са локализирани в папката All User Control. Това се осъществява чрез guna2Elipse1, guna2Elipse2, guna2Elipse3, guna2Elipse4, guna2Elipse5, които са специфични контроли, добавени чрез библиотеката Guna.UI2.dll. Тези контроли се намират и в останалите формички, за да се осъществи връзката с DashBoard-a.

3.3.5 Структура на базата данни myHotel (накратко)

Базата данни myHotel е проектирана за управление на хотелска система и включва следните основни таблици:

1. Таблица: rooms

Таблицата **rooms** съхранява информация за всички стаи в хотела.

Таблица 1

Име на колона	Тип данни	Описание
roomid	int, Identity(1,1), Primary Key	Уникален идентификатор на стаята. Автоматично нарастващ.
roomid	int, Identity(1,1), Primary Key	Уникален идентификатор на стаята. Автоматично нарастващ.
Room No	varchar(250), Not Null, Unique	Номер на стаята. Уникален за всяка стая.
Room Type	varchar(250), Not Null	Тип на стаята (например "Single", "Double", "Suite").
bed	varchar(250), Not Null	Тип на леглото (например "Single Bed", "Double Bed").
price	decimal(18,2), Not Null	Цена за една нощувка. Десетично число с два знака след запетаята.

2. Таблица: customer

Таблицата **customer** съхранява данните на клиентите на хотела.

Таблица 2

Име на колона	Тип данни	Описание
cid	int, Identity(1,1), Primary Key	Уникален идентификатор на клиента.
cname	varchar(250), Not Null	Име на клиента.
mobile	.bigint, Not Null	Телефонен номер на клиента.
nationality	varchar(250), Not Null	Националност на клиента.
gender	varchar(50), Not Null	Пол на клиента.
dob	varchar(50), Not Null	Дата на раждане на клиента (запазена като текст).
idproof	varchar(250), Not Null	Документ за самоличност (напр.

		паспорт, лична карта).
email	varchar(350), Not Null	Имейл адрес на клиента.
checkin	varchar(250), Not Null	Дата на настаняване.
checkout	varchar(250)	Дата на напускане (може да бъде празна при текущ престой).
chekout	varchar(250), Not Null, Default 'NO'	Статус на напускането. (Има правописна грешка в името, препоръчва се промяна на checkout_status.)
roomid	int, Not Null, Foreign Key	Идентификатор на стаята, заемана от клиента. Свързан е с roomid от таблицата rooms.

Външна връзка:

- roomid е външен ключ към rooms(roomid), което гарантира, че клиентите могат да заемат само съществуващи стаи.

3. Таблица: employee

Таблицата **employee** съхранява информация за служителите на хотела.

Таблица 3

Име на колона	Тип данни	Описание
eid	int, Identity(1,1), Primary Key	Уникален идентификатор на служителя.
ename	varchar(250), Not Null	Име на служителя.
worktime	varchar(250), Not Null	Работно време на служителя (например "08:00-16:00").
position	varchar(250), Not Null	Дължност на служителя (например "Рецепционист", "Мениджър").
emailid	varchar(250), Not Null	Имейл адрес на служителя.
username	varchar(150), Not Null	Потребителско име за вход в

		системата.
pass	varchar(150), Not Null	Парола за вход (за по-голяма сигурност препоръчително е да бъде криптирана).

3.3.6. Връзки между таблиците

- `rooms` и `customer` са свързани чрез външния ключ `roomid`.
- Всеки клиент е асоцииран с една стая, но една стая може да бъде заета от максимум един клиент в даден момент.
- `employee` е отделна таблица без външни връзки, предназначена за управление на потребителите на системата.

4. ПРОЕКТНА ЧАСТ

4.1. Архитектура на системата

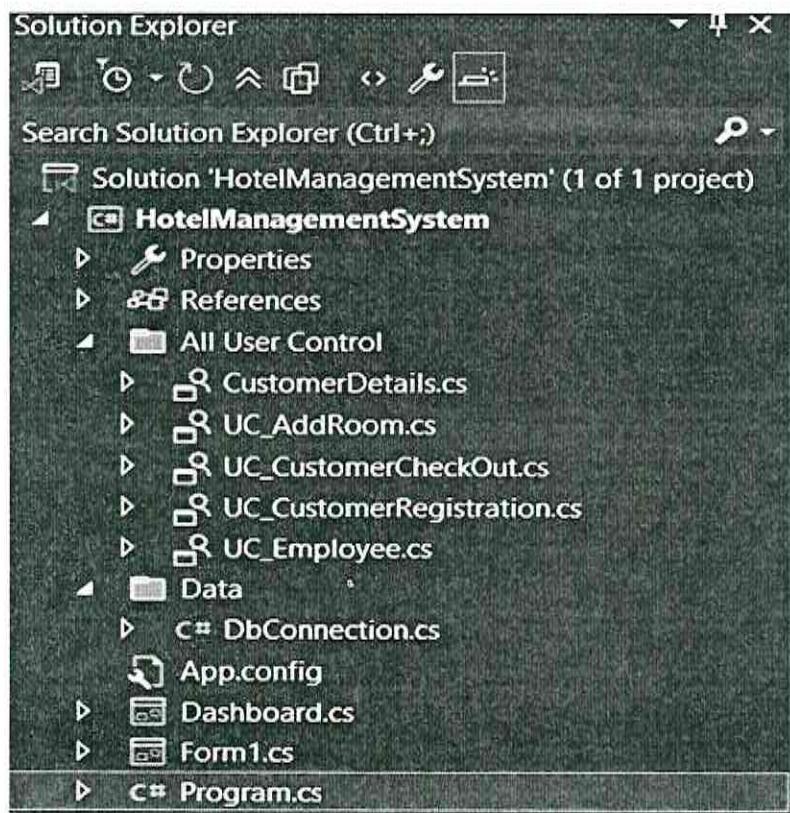
Системата за управление на хотел е реализирана като клиентско приложение с графичен интерфейс, разработен на платформата Windows Forms. Базата данни е изградена с Microsoft SQL Server. Комуникацията между приложението и базата данни се осъществява чрез ADO.NET технологии.

Тази архитектура осигурява бърза работа на приложението и лесна поддръжка на данните.

4.2. Връзка между базата данни myHotel и C#

За решаването на проблемите и за да може системата за управление на хотел да работи мултифункционално, е нужна връзка между базата данни и приложението. Тя се осъществява чрез клас `DbConnection`, разположен в папката `Data`:

Фигура 15



Кластьт **DbConnection** отговаря за управлението на връзката между приложението и базата данни **myHotel**. Той предоставя методи за извлечане, добавяне, изтриване и обновяване на данни, както и за зареждане на данни в **ComboBox**.

Кластьт използва **ADO.NET** и работи със **SQL Server**.

Основни елементи:

1. Метод: `protected SqlConnection getConnection()`

```
protected SqlConnection getConnection()
```

- Създава нов обект `SqlConnection`.
- Настройва `connection string` за свързване към SQL Server инстанция `DESKTOP-CT2J624\SQLEXPRESS` и базата данни `myHotel`.
- Използва **Integrated Security**, което означава, че Windows акаунтът на потребителя ще се използва за автентикация.
- Връща създадената връзка.

2. Метод: `public DataSet getData (String query)`

```
public DataSet getData (String query)
```

- Изпълнява дадена **SELECT** заявка.
- Връща резултатите като обект от тип `DataSet`.
- Използва `SqlDataAdapter`, който автоматично отваря и затваря връзката с базата данни.
- Подходящ за извлечане на данни, които после ще се визуализират в контроли като `DataGridView`.

3. Метод: `public void setData(String query, String message)`

```
public void setData(String query, String message)
```

- Изпълнява заявка за **INSERT**, **UPDATE** или **DELETE** върху базата данни.
- Отваря връзка, изпълнява заявката и затваря връзката.
- След успешно изпълнение показва съобщение чрез `MessageBox`, съдържащо предоставения текст.

4. Метод: public SqlDataReader getForCombo(String query)

```
public SqlDataReader getForCombo(String query)
```

- Изпълнява дадена заявка и връща резултатите като SqlDataReader.
- Използва се най-често за зареждане на динамични данни в **ComboBox** или други падащи менюта.
- Изиска ръчно затваряне на връзката след приключване на четенето.

Класът **DbConnection** служи като централизирано място за всички операции свързани с базата данни. Това улеснява поддръжката и развитието на приложението, като осигурява:

- Централизирано управление на връзката към базата данни.
- Лесни за използване методи за четене и писане на данни.
- Повишена модулност и повторна употреба на кода.

4.3. Модул "Управление на стаи"

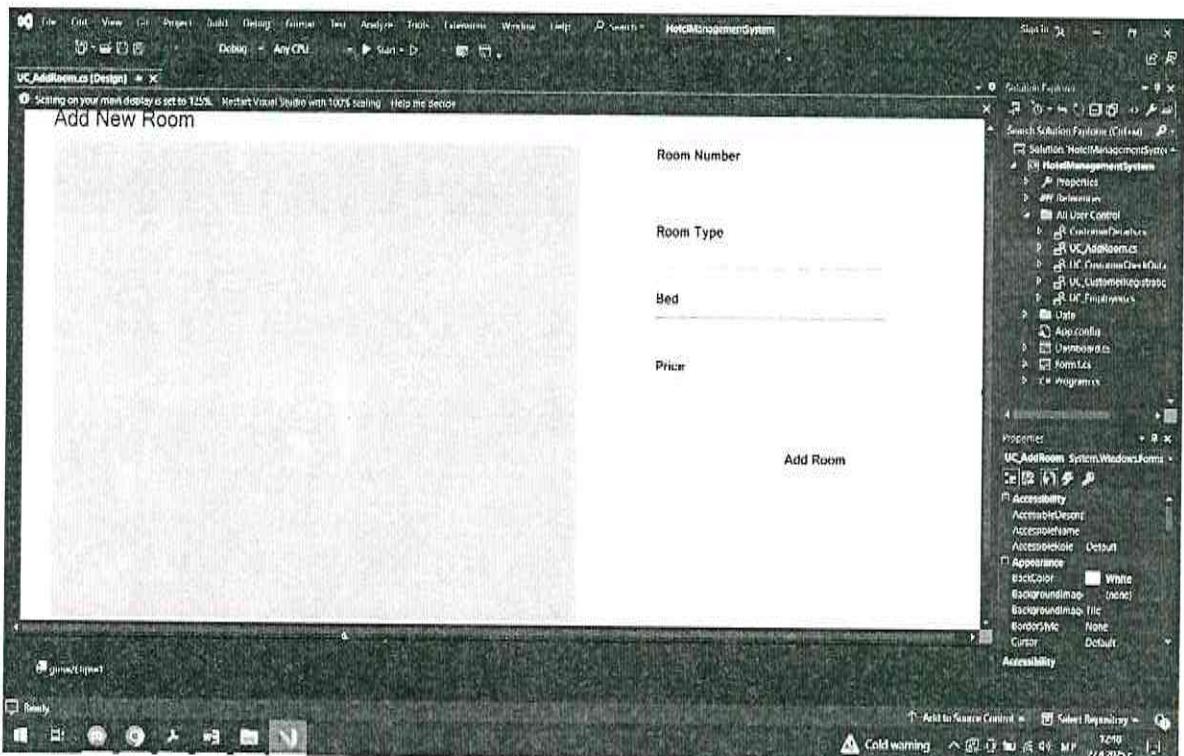
За по-добра организация на наличностите е създаден модул за стаите. Потребителят може да:

- Преглежда всички стаи по статус (свободна/заета/за почистване);
- Добавя нова стая или редактира данни за съществуваща стая.

Това решава проблема с невъзможността за бързо проверяване на свободни стаи.

Управлението на стаи се осъществява чрез формата UC_AddRoom.cs и кода към нея:

Фигура 18



Фигура 19

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using HotelManagementSystem.Data;

namespace HotelManagementSystem.All_User_Control
{
    public partial class UC_AddRoom : UserControl
    {
        DbConnection db = new DbConnection();
        String query;

        public UC_AddRoom()
        {
            InitializeComponent();
        }

        private void UC_AddRoom_Load(object sender, EventArgs e)
        {
            query = "Select * from rooms";
            DataSet ds = db.getData(query);
            dataGridView1.DataSource = ds.Tables[0];
        }
    }
}
```

Фигура 20

```
UC_AddRoom.cs  X  UC_AddRoom.cs (Disc00)
```

```
HotelManagementSystem
private void btnAddRoom_Click(object sender, EventArgs e)
{
    if (txtRoomNo.Text != "" && txtType.Text != "" && txtBed.Text != "" && txtPrice.Text != "")
    {
        String roomNo = txtRoomNo.Text;
        String type = txtType.Text;
        String bed = txtBed.Text;
        decimal price = decimal.Parse(txtPrice.Text);

        query = "insert into rooms (roomNo, roomType, bed, price) values ('" + roomNo + "','" + type + "','" + bed + "','" + price + "')";
        db.setData(query, "Room Added.");
    }

    UC_AddRoom_Load(this, null);
    clearAll();
}
else
{
    MessageBox.Show("Fill All Fields", "Warning !!", MessageBoxButtons.OK, MessageBoxIcon.Information);
}

public void clearAll()
{
    txtRoomNo.Clear();
    txtType.SelectedIndex = -1;
    txtBed.SelectedIndex = -1;
    txtPrice.Clear();
}

private void UC_AddRoom_Leave(object sender, EventArgs e)
{
    clearAll();
}
```

Фигура 21

```
private void UC_AddRoom_Enter(object sender, EventArgs e)
{
    UC_AddRoom_Load(this, null);
}
```

Основни елементи:

- Пространства от имена:

```
using HotelManagementSystem.Data;
```

Импортира се пространството от имена, в което се намира класът DbConnection, отговорен за връзката с базата данни.

- Полета:

```
DbConnection db = new DbConnection();
```

```
String query;
```

db: Създава нов обект за комуникация с базата данни.

query: Променлива за съхранение на SQL заявки.

- Конструктор:

```
public UC_AddRoom()
{
    InitializeComponent();
}
```

Инициализира графичните компоненти на контрола.

- Събития:

1. Загрузка на контрола: UC_AddRoom_Load

```
private void UC_AddRoom_Load(object sender, EventArgs e)
```

Изпълнява се, когато контролът се зареди.

Извлича всички данни от таблицата rooms.

Задава извлечените данни като източник за dataGridView1, за да се визуализират всички налични стаи.

2. Натискане на бутона за добавяне на стая: btnAddRoom_Click

```
private void btnAddRoom_Click(object sender, EventArgs e)
```

Проверява дали всички задължителни полета (номер на стая, тип, легло, цена) са попълнени.

Ако да:

Извлича данните от текстовите полета.

Създава SQL заявка за добавяне на нова стая в таблицата rooms.

Изпълнява заявката чрез метода setData и показва съобщение за успех.

Презарежда списъка със стаи чрез UC_AddRoom_Load.

Изчиства полетата с метода clearAll.

Ако не:

Показва предупредително съобщение да се попълнят всички полета.

3. Метод за изчистване на полетата: clearAll

```
public void clearAll()
```

Изчиства текстовите полета и избора в падащите менюта (тип стая и вид легло).

4. Напускане на контрола: UC_AddRoom_Leave

```
private void UC_AddRoom_Leave(object sender, EventArgs e)
```

При напускане на контрола всички въведени данни се изчистват автоматично.

5. Влизане в контрола: UC_AddRoom_Enter

```
private void UC_AddRoom_Enter(object sender, EventArgs e)
```

При влизане в контрола, се презареждат данните от базата, за да се покаже актуализирания списък със стаи.

Кластьт **UC_AddRoom** предоставя основни функционалности за:

- Добавяне на нови хотелски стаи в базата данни.
- Автоматично опресняване на списъка със стаи.
- Поддържане на потребителския интерфейс чист и актуален.

Той работи в комбинация с класа **DbConnection**, който управлява директната връзка и заявки към базата данни.

4.4. Модул "Управление на резервации и клиенти"

Създаден е модул за регистриране на резервации, който:

Проверява автоматично дали стаята е свободна за избрани дати;

Свързва резервацията с конкретен клиент и стая;

Така се решава проблемът с човешките грешки при резервации.

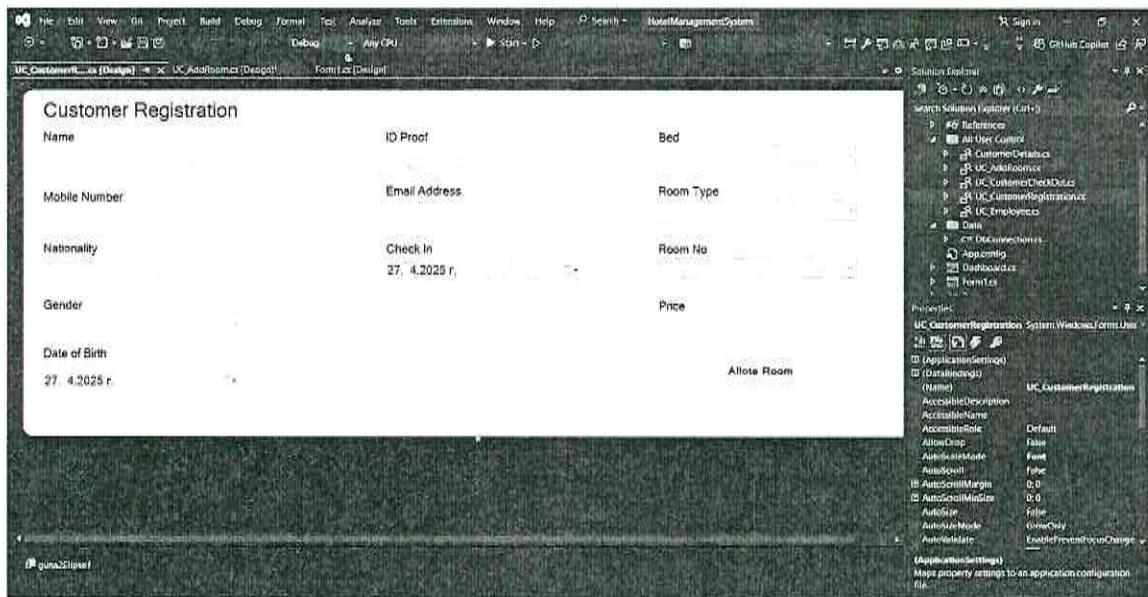
За решаване на проблема с ръчното записване на клиентски данни, е създаден модул за управление на клиенти. Чрез него потребителят може да:

- Добавя нов клиент;
- Редактира съществуваща информация;

По този начин се избягват грешки и се ускорява обслужването.

Те се илюстрират чрез три User Controls, които са разположени в папката All User Control (UC_Registration.cs, UC_CustomerCheckOut.cs и CustomerDetails.cs):

Фигура 22



Този код дефинира потребителски контрол **UC_CustomerRegistration**, част от проекта **HotelManagementSystem**, който осигурява интерфейс за регистриране на нови клиенти и резервиране на хотелски стаи.

Основни елементи:

Пространства от имена:

```
using HotelManagementSystem.Data;
```

Импортира се пространството, където е дефиниран класът **DbConnection**, отговарящ за връзката и комуникацията с базата данни.

Полета:

```
DbConnection db = new DbConnection();
String query;
int rid;

db: Създава обект за достъп до базата данни.
query: Съхранява текущата SQL заявка.
rid: Записва ID на избраната стая от базата данни.
```

Конструктор:

```
public UC_CustomerRegistration()
{
    InitializeComponent();
}
```

Инициализира компонентите на формата.

Методи:

1. setComboBox

```
public void setComboBox(String query, ComboBox combo)
```

Изпълнява подадената SQL заявка и добавя върнатите стойности в падащото меню (ComboBox).

Използва SqlDataReader за четене на данните от базата.

2. Събития (Events):

2.1. Зареждане на контрола: UC_CustomerRegistration_Load

```
private void UC_CustomerRegistration_Load(object sender, EventArgs e)
```

Празно за момента. Може да се използва за бъдещи начални зареждания на данни.

2.2. Смяна на избора в txtType

```
private void txtType_SelectedIndexChanged(object sender, EventArgs e)
```

При избор на тип стая, изчиства списъка с налични стаи и зарежда от базата данни номерата на свободните стаи (booked = 'NO'), съобразно избраните тип стая и легло.

2.3. Смяна на избора в txtBed

```
private void txtBed_SelectedIndexChanged(object sender, EventArgs e)
```

При смяна на типа легло:

Нулира избора на тип стая.

Изчиства списъка с налични стаи.

2.4. Смяна на избора в txtRoomNo

```
private void txtRoomNo_SelectedIndexChanged(object sender, EventArgs e)
```

При избор на номер на стая:

Зарежда цената (price) и ID-то (roomid) на избраната стая от базата данни.

Показва цената в полето за цена (txtPrice).

3. Натискане на бутона за регистриране: btnAlloteRoom_Click

Копиране Редактиране

```
private void btnAlloteRoom_Click(object sender, EventArgs e)
```

Проверява дали всички задължителни полета са попълнени.

Ако да:

Извлича данните от формата.

Създава SQL заявка за добавяне на нов клиент в таблицата customer.

Актуализира таблицата rooms, като маркира избраната стая като заета (booked = 'YES').

Използва метода setData за изпълнение на заявката и показване на съобщение за успешна резервация.

Ако не:

Показва съобщение, че всички полета са задължителни.

4. Изчистване на всички полета: clearAll

```
public void clearAll()
```

Изчиства съдържанието на всички текстови полета, комбобоксове и ресетва датите.

5. Напускане на контрола: UC_CustomerRegistration_Leave

```
private void UC_CustomerRegistration_Leave(object sender, EventArgs e)
```

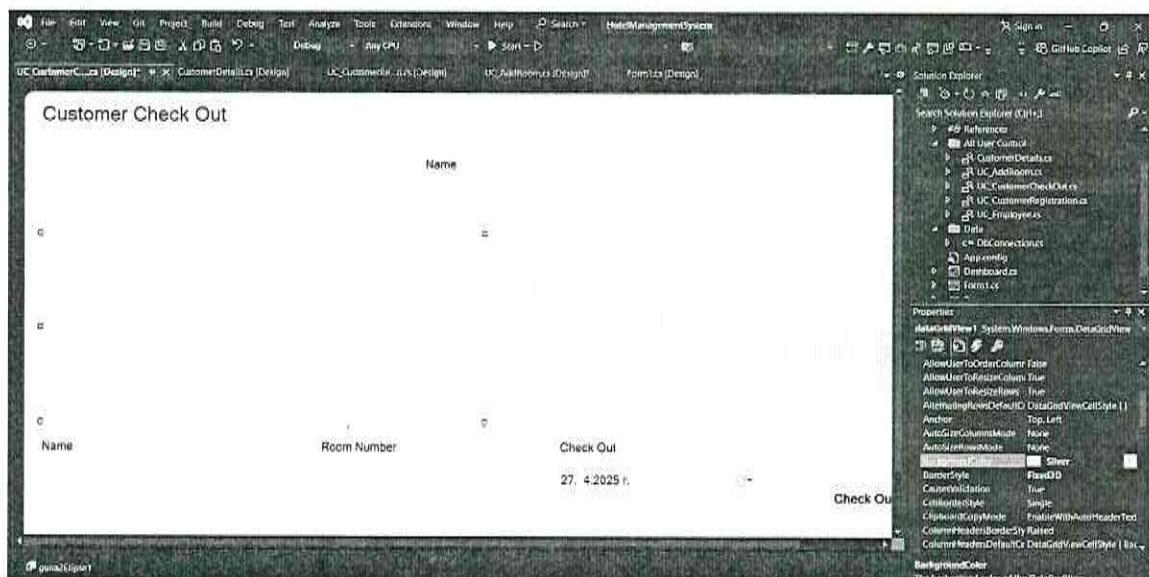
При напускане на потребителския контрол се извика методът clearAll, за да се изчистят всички въведени данни.

Класът **UC_CustomerRegistration**:

- Позволява избор на налична стая според тип стая и тип легло.

- Позволява на потребителя да регистрира клиент, като автоматично запазва стаята и отбелязва в базата данни, че тя вече е заета.
- Поддържа динамично зареждане на информацията от базата данни към формата.
- Осигурява лесен начин за почистване на полетата при напускане на формата.

Фигура 23



Фигура 24

```
UC_CustomerCheckOut.cs  X: UC_CustomerCh_ults [Design]  HotelManagementSystem  ucCustomerCheckOut.cs.csproj  HotelManagementSystem\All_Users\Control\UC_CustomerCheckOut.cs  btnCheckOut_Click(object sender, EventArgs e)
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using HotelManagementSystem.Data;

namespace HotelManagementSystem.All_Users_Control
{
    public partial class UC_CustomerCheckOut : UserControl
    {
        DbConnection db = new DbConnection();
        String query;
        1 reference
        public UC_CustomerCheckOut()
        {
            InitializeComponent();
        }

        2 references
        private void UC_CustomerCheckOut_Load(object sender, EventArgs e)
        {
            query = "select customer.cid,customer.cname,customer.mobile,customer.nationality,customer.gender,customer.dob,customer.idproof," +
                    "customer.email,customer.checkin,rooms.roomNo,rooms.roomType,rooms.bed,rooms.price from customer inner join rooms on " +
                    "customer.roomid = rooms.roomid where checkout = 'NO'";
            DataSet ds = db.getData(query);
            dataGridView1.DataSource = ds.Tables[0];
        }
    }
}
```

Фигура 25

```
UC_CustomerCheckOut.cs  X: UC_CustomerCh_ults [Design]  HotelManagementSystem  ucCustomerCheckOut.cs.csproj  HotelManagementSystem\All_Users\Control\UC_CustomerCheckOut.cs  btnCheckOut_Click(object sender, EventArgs e)
```

```
1 reference
private void txtName_TextChanged(object sender, EventArgs e)
{
    query = "select customer.cid,customer.cname,customer.mobile,customer.nationality,customer.gender,customer.dob,customer.idproof," +
            "customer.email,customer.checkin,rooms.roomNo,rooms.roomType,rooms.bed,rooms.price from customer inner join rooms on " +
            "customer.roomid = rooms.roomid where cname like '" + txtName.Text + "%' and checkout='NO'";
    DataSet ds = db.getData(query);
    dataGridView1.DataSource = ds.Tables[0];
}

int id;
1 reference
private void dataGridView1_CellContentClick(object sender, DataGridViewCellEventArgs e)
{
    if (dataGridView1.Rows[e.RowIndex].Cells[e.ColumnIndex].Value != null)
    {
        id = int.Parse(dataGridView1.Rows[e.RowIndex].Cells[0].Value.ToString());
        txtName.Text = dataGridView1.Rows[e.RowIndex].Cells[1].Value.ToString();
        txtRoom.Text = dataGridView1.Rows[e.RowIndex].Cells[2].Value.ToString();
        ExtRoom.Text = dataGridView1.Rows[e.RowIndex].Cells[3].Value.ToString();
    }
}
```

Фигура 26

```
private void btnCheckOut_Click(object sender, EventArgs e)
{
    if (txtName.Text != "")
    {
        if (MessageBox.Show("Are You Sure?", "Confirmation", MessageBoxButtons.OKCancel, MessageBoxIcon.Warning) == DialogResult.OK)
        {
            String cdate = txtCheckOut.Text;
            query = "update customer set checkout = 'YES', checkoutdate = '" + cdate + "' where id = " + id + " update rooms set booked = 'NO'" +
                    " where roomno = '" + txtRoom.Text + "'";
            db.setData(query, "Check Out Successfully.");
            UC_CustomerCheckOut_Load(this, null);
            clearAll();
        }
    }
    else
    {
        MessageBox.Show("No Customer Selected.", "Message", MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
}

public void clearAll()
{
    txtName.Clear();
    txtName.Clear();
    txtRoom.Clear();
    txtCheckOut.ResetText();
}

private void UC_CustomerCheckOut_Leave(object sender, EventArgs e)
{
    clearAll();
}
```

Този код дефинира потребителски контрол **UC_CustomerCheckOut**, който е част от проекта **HotelManagementSystem**.

Той предоставя функционалност за напускане на клиенти (Check-Out) от хотела, като актуализира съответните записи в базата данни.

Основни елементи:

Пространства от имена:

using HotelManagementSystem.Data;

Импортира класове за работа с базата данни чрез `DbConnection`.

Полета:

`DbConnection db = new DbConnection();`

`String query;`

`int id;`

`db`: обект за работа с базата данни.

`query`: динамична SQL заявка.

`id`: записва ID на избрания клиент от таблицата.

Конструктор:

```
public UC_CustomerCheckOut()  
{  
    InitializeComponent();  
}
```

Инициализира компонентите на потребителския контрол.

Методи:

1. Зареждане на всички клиенти без чек-аут: UC_CustomerCheckOut_Load

```
private void UC_CustomerCheckOut_Load(object sender, EventArgs e)
```

Изпълнява SQL заявка, която избира всички клиенти, които все още не са направили чек-аут (chekout = 'NO').

Данните се визуализират в DataGridView.

2. Филтриране на клиенти по име: txtName_TextChanged

```
private void txtName_TextChanged(object sender, EventArgs e)
```

При въвеждане на текст в полето за име (txtName), се търсят клиенти, чиито имена започват със съответния текст и все още не са направили чек-аут.

Филтрираните резултати се зареждат отново в DataGridView.

3. Избор на клиент: dataGridView1_CellContentClick

```
private void dataGridView1_CellContentClick(object sender, DataGridViewCellEventArgs e)
```

При клик върху ред в таблицата:

Съхранява се id на клиента.

Показва се името на клиента (txtCName) и номера на стаята (txtRoom) за лесно потвърждение.

4. Извършване на чек-аут: btnCheckOut_Click

```
private void btnCheckOut_Click(object sender, EventArgs e)
```

Проверява дали има избран клиент.

Ако има:

Пита за потвърждение чрез MessageBox.

Ако се потвърди, изпълнява:

Актуализация на таблицата customer като задава chekout = 'YES' и записва датата на чек-аут.

Актуализация на таблицата rooms, като отбелязва, че стаята вече е свободна (booked = 'NO').

Обновява визуализираните данни и изчиства всички полета.

Ако няма избран клиент:

Показва съобщение за липса на селекция.

5. Изчистване на всички полета: clearAll

```
public void clearAll()
```

Изчиства текстовите полета и ресетва датата за чек-аут.

6. Изчистване при напускане на формата: UC_CustomerCheckOut_Leave

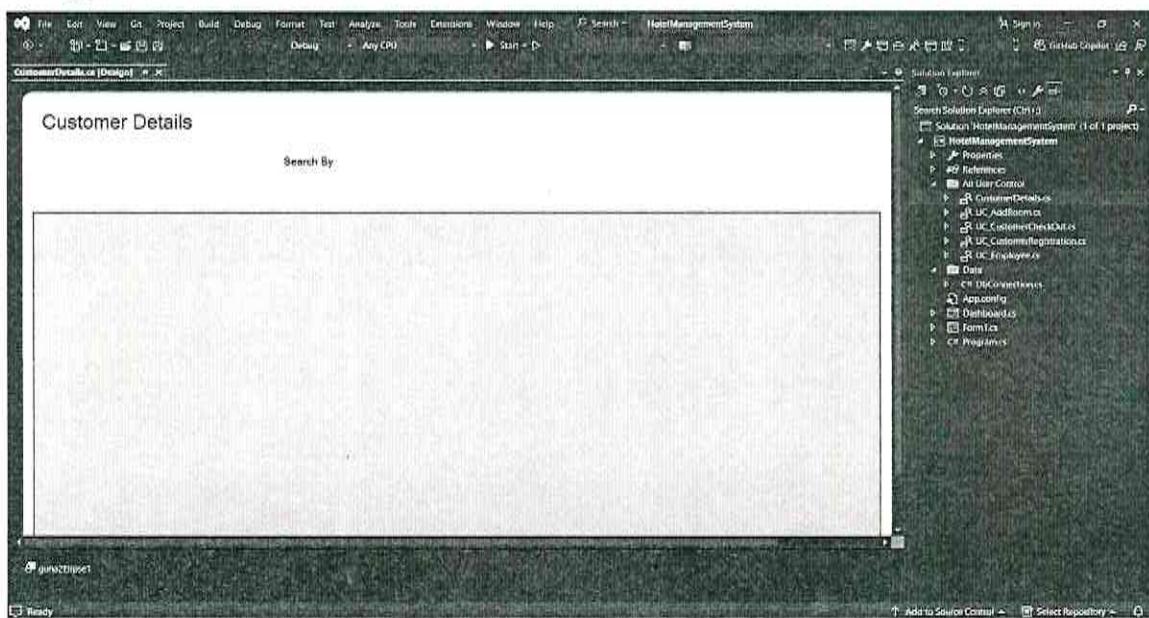
```
private void UC_CustomerCheckOut_Leave(object sender, EventArgs e)
```

При напускане на потребителския контрол всички полета се изчистват автоматично.

Класът UC_CustomerCheckOut:

- Позволява лесно намиране на клиенти, които все още са настанени.
- Позволява извършване на чек-аут на клиент с едно натискане на бутон.
- Актуализира както информацията за клиента, така и статуса на стаята в базата данни.
- Поддържа чистота и коректност на данните в базата.
- Подобрява удобството на потребителя чрез лесна навигация и автоматично изчистване на формата.

Фигура 27



Фигура 28



Фигура 29

```
CustomerDetails.cs  F - X - ListHotelDetails.cs [Design]  HotelManagementSystem All User Control CustomerDetails.cs  References  private void txtSearchBy_SelectedIndexChanged(object sender, EventArgs e) {    if (txtSearchBy.SelectedIndex == 0)    {      query = "select customer.cid,customer.cname,customer.mobile,customer.nationality,customer.gender,customer.dob,customer.idproof," +      "customer.email,customer.checkin,customer.checkout,rooms.roomNo,rooms.roomType,rooms.bed,rooms.price from customer" +      "inner join rooms on customer.roomid = rooms.roomid";      getRecord(query);    }    else if (txtSearchBy.SelectedIndex == 1)    {      query = "select customer.cid,customer.cname,customer.mobile,customer.nationality,customer.gender,customer.dob,customer.idproof," +      "customer.email,customer.checkin,customer.checkout,rooms.roomNo,rooms.roomType,rooms.bed,rooms.price from customer" +      "inner join rooms on customer.roomid = rooms.roomid where checkout is null";      getRecord(query);    }    else if (txtSearchBy.SelectedIndex == 2)    {      query = "select customer.cid,customer.cname,customer.mobile,customer.nationality,customer.gender,customer.dob,customer.idproof," +      "customer.email,customer.checkin,customer.checkout,rooms.roomNo,rooms.roomType,rooms.bed,rooms.price from customer" +      "inner join rooms on customer.roomid = rooms.roomid where checkout is not null";      getRecord(query);    }  }  References  private void getRecord(string query)  {    DataSet ds = db.getData(query);    dataGridView1.DataSource = ds.Tables[0];  }
```

Класът CustomerDetails.cs представлява потребителски контрол в системата за управление на хотел (HotelManagementSystem). Той служи за показване на всички клиенти, като дава възможност да се филтрират по различни критерии.

Основни части на кода:

Пространства от имена:

```
using HotelManagementSystem.Data;
```

Импортира достъп до базата данни чрез класа DbConnection.

Полета:

```
DbConnection db = new DbConnection();
```

```
String query;
```

db – създава обект за връзка с базата данни.

query – съдържа SQL заявката, която ще бъде изпълнена.

Конструктор:

```
public CustomerDetails()  {    InitializeComponent();}
```

}

Инициализира компонентите на потребителския контрол.

Методи:

1. Смяна на критерия за търсене: txtSearchBy_SelectedIndexChanged

```
private void txtSearchBy_SelectedIndexChanged(object sender, EventArgs e)
```

Този метод се изпълнява автоматично, когато потребителят избере нова стойност от падащото меню txtSearchBy.

Според избора, се съставя съответна SQL заявка:

Ако е избрано "Всички клиенти" (SelectedIndex == 0):

Извличат се всички клиенти, независимо дали са напуснали или не.

Ако е избрано "Настанени клиенти" (SelectedIndex == 1):

Извличат се само клиенти, които все още **не са напуснали** (checkout is null).

Ако е избрано "Клиенти, които са напуснали" (SelectedIndex == 2):

Извличат се клиенти, които **вече са напуснали** (checkout is not null).

След съставяне на заявката, методът вика помощен метод getRecord(query) за изпълнението ѝ.

2. Изпълнение на заявката и показване на резултатите: getRecord

```
private void getRecord(String query)
```

Изпълнява подадената SQL заявка чрез обекта db.

Зарежда резултатите в **DataGridView** (dataGridView1), за да може потребителят да ги разгледа визуално.

Какво съдържа всяка заявка?

Извличат се данни от двете таблици:

customer – за данни за клиента: ID, име, телефон, националност, пол, рождена дата, личен документ, имейл, дата на настаняване и напускане.

rooms – за данни за стаята: номер на стая, тип на стаята, тип легло и цена.

Таблиците се свързват чрез:

```
inner join rooms on customer.roomid = rooms.roomid
```

Тоест, взимат се клиенти заедно със стаите, които обитават или са обитавали.

Класът CustomerDetails:

Позволява на администратора на хотела да вижда всички клиенти в системата.

Осигурява **фильтрация** на клиентите според тяхното състояние (настанени, напуснали или всички).

Използва отделен метод getRecord за по-добра поддръжка и преизползваемост на кода.

Удобно показва информацията в табличен формат.

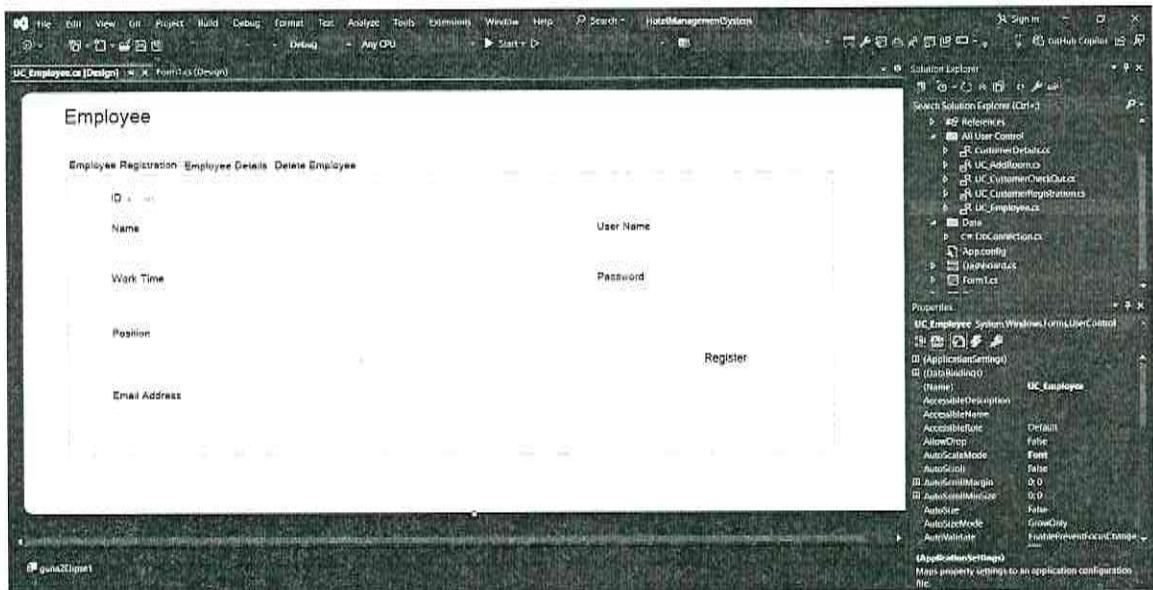
4.5. Модул "Управление на персонал"

За решаване на проблема с липсата на централизирана информация за служителите, е разработен модул за управление на персонал. Чрез него администраторът на хотела може да:

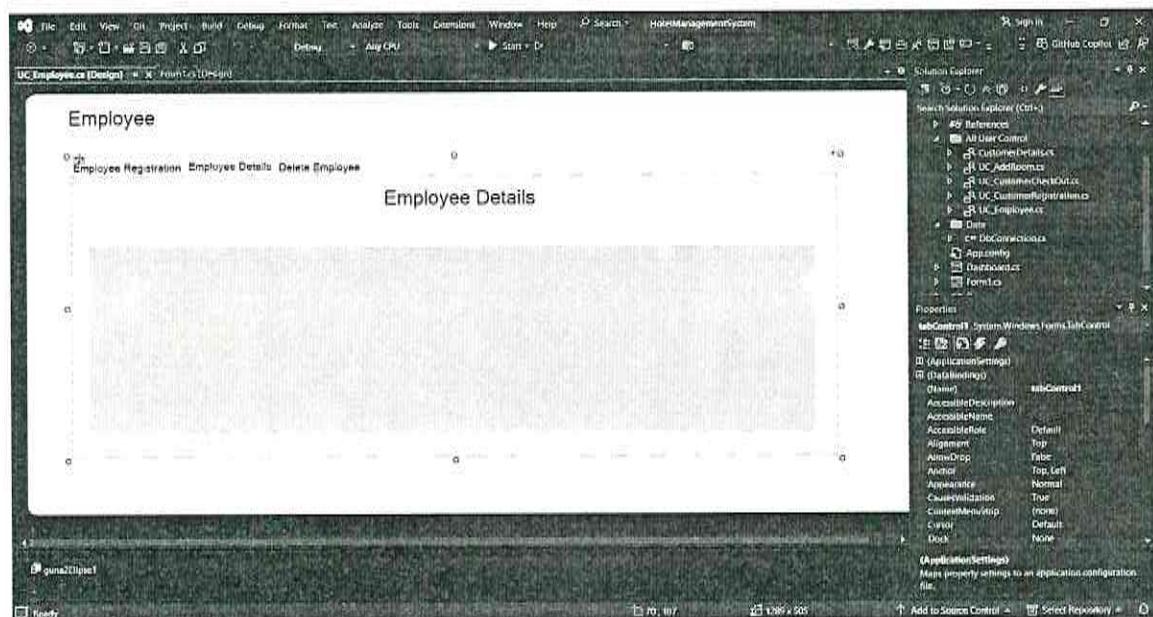
- Регистрира нов служител, като попълни данни за име, длъжност, дата на започване;
- Изтрива профили на служители, които напускат;
- Преглежда справки за всички служители, сортирани по длъжност или дата на назначаване.

Този модул осигурява по-добро управление на човешките ресурси в хотела и улеснява ръководството при взимане на решения относно персонала. Той се илюстрира чрез User Control (UC_Employee.cs):

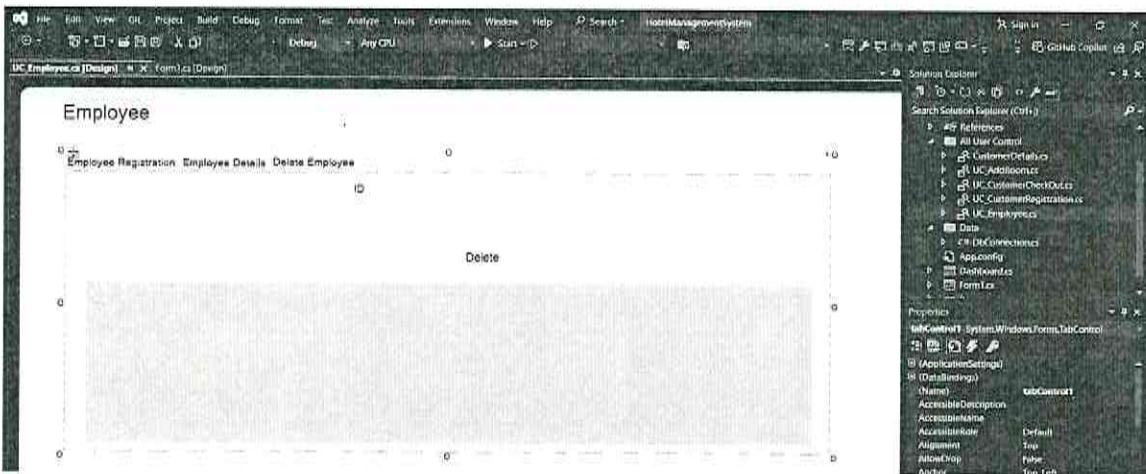
Фигура 30



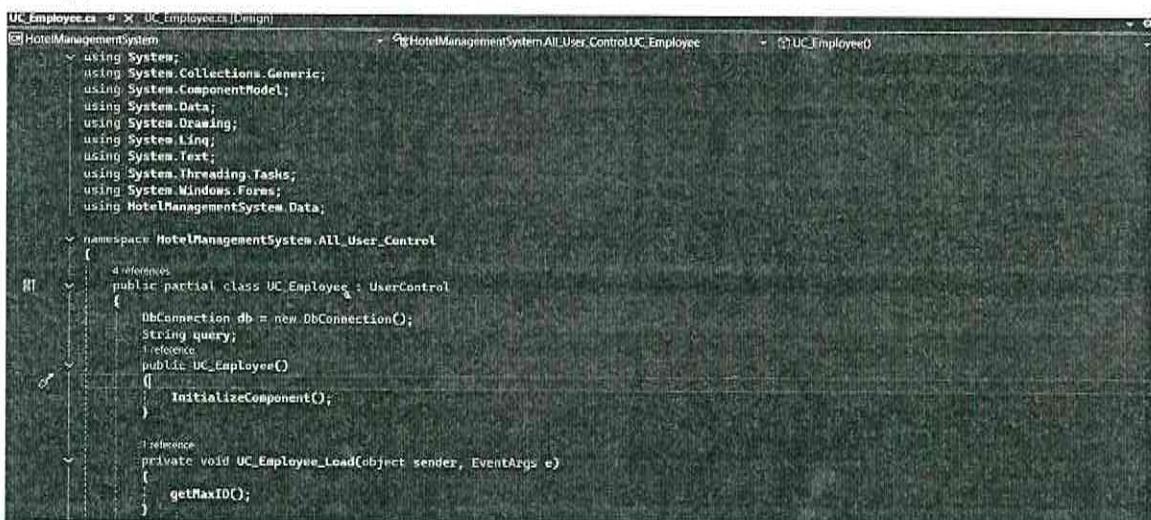
Фигура 31



Фигура 32



Фигура 33



Фигура 34

```
private void btnRegister_Click(object sender, EventArgs e)
{
    if (txtName.Text != "" && txtWorkTime.Text != "" && txtPosition.Text != "" && txtEmail.Text != "" && txtUsername.Text != "" && txtPass
    {
        String name = txtName.Text;
        String workTime = txtWorkTime.Text;
        String position = txtPosition.Text;
        String email = txtEmail.Text;
        String username = txtUsername.Text;
        String password = txtPassword.Text;

        query = "insert into employee (name,worktime,position,emailid,username,pass) values ('" + name + "','" + workTime + "','" + position + "','" + em
        db.setData(query, "Employee Registered.");
        clearAll();
        getMaxID();
    }
    else
    {
        MessageBox.Show("Fill all Fields.", "Warning...", MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
}

//----- REQUIRED METHOD -----
public void getMaxID()
{
    query = "select max(eid) from employee";
    DataSet ds = db.getData(query);

    if (ds.Tables[0].Rows[0][0].ToString() != "")
    {
        Int64 num = Int64.Parse(ds.Tables[0].Rows[0][0].ToString());
        labelToSET.Text = (num + 1).ToString();
    }
}
```

Фигура 35

```
public void clearAll()
{
    txtName.Clear();
    txtWorkTime.Clear();
    txtPosition.SelectedIndex = -1;
    txtEmail.Clear();
    txtUsername.Clear();
    txtPassword.Clear();
}

public void setEmployee(DataGridView dgv)
{
    query = "select * from employee";
    dataset ds = db.getData(query);
    dgv.DataSource = ds.Tables[0];
}

private void tabControl1_SelectedIndexChanged(object sender, EventArgs e)
{
    if (tabControl1.SelectedIndex == 1)
    {
        setEmployee(dataGridView1);
    }
    else if (tabControl1.SelectedIndex == 2)
    {
        setEmployee(dataGridView2);
    }
}
```

Фигура 36

```
1. Reference
2. private void btnDelete_Click(object sender, EventArgs e)
3. {
4.     if (txtID.Text != "")
5.     {
6.         if (MessageBox.Show("Are You Sure?", "Confirmation...!", MessageBoxButtons.YesNo, MessageBoxIcon.Warning) == DialogResult.Yes)
7.         {
8.             query = "delete from employee where eid =" + txtID.Text + "";
9.             db.setdata(query, "Record Deleted.");
10.            tabControl1_SelectedIndexChanged(this, null);
11.        }
12.    }
13.    else
14.    {
15.        MessageBox.Show("Fill field for ID", "Information !!!", MessageBoxButtons.OK, MessageBoxIcon.Information);
16.    }
17.
18.    private void UC_Employee_Leave(object sender, EventArgs e)
19.    {
20.        clearAll();
21.    }
22. }
```

Това е потребителски контрол (**UserControl**) за регистрация, показване и изтриване на служители в хотелската система.

Основни компоненти в кода:

Пространства от имена:

using HotelManagementSystem.Data;

Зарежда връзката към базата данни чрез DbConnection.

Полета:

DbConnection db = new DbConnection();

String query;

db – създава връзка с базата данни.

query – съхранява SQL заявката, която ще се изпълнява.

Конструктор:

public UC_Employee()

{

InitializeComponent();

}

Стандартен конструктор, който инициализира контролата.

Основни методи:

1. При зареждане на контролата: UC_Employee_Load

```
private void UC_Employee_Load(object sender, EventArgs e)
```

Когато контролата се зареди, автоматично вика getMaxID(), за да покаже следващия свободен ID за нов служител.

2. Регистрация на нов служител: btnRegister_Click

```
private void btnRegister_Click(object sender, EventArgs e)
```

Проверява дали всички текстови полета са попълнени.

Ако са:

Взима данните от полетата (име, работно време, позиция, имейл, потребителско име и парола).

Съставя SQL заявка за вмъкване (**INSERT**) на нов запис в таблицата employee.

Изпълнява заявката и показва съобщение „Employee Registered.“

След регистрацията:

Изчиства всички полета (clearAll()).

Обновява показвания ID (getMaxID()).

Ако не са попълнени всички полета, показва предупредително съобщение.

3. Вземане на следващото ID: getMaxID

```
public void getMaxID()
```

Взима най-голямото (MAX) съществуващо eid (ID на служител) от базата данни.

Добавя 1 към него и го показва в етикет labelToSET.

Ако няма нито един служител, ще трябва да се добави първият (ID = 1).

4. Изчистване на всички полета: clearAll

```
public void clearAll()
```

Изчиства текстовите кутии и нулира избора на позиция.

5. Зареждане на всички служители в таблица: setEmployee

```
public void setEmployee(DataGridView dgv)
```

Взима всички записи от таблицата employee.

Поставя ги в таблицата **DataGridView** (dgv), за да се визуализират.

6. Превключване между табове: tabControl1_SelectedIndexChanged

```
private void tabControl1_SelectedIndexChanged(object sender, EventArgs e)
```

Когато потребителят смени таба в интерфейса:

Ако отиде на таба за преглед на служители (SelectedIndex == 1 или 2), зареждат се всички служители в съответния DataGridView.

7. Изтриване на служител по ID: btnDelete_Click

```
private void btnDelete_Click(object sender, EventArgs e)
```

Проверява дали е въведено ID (txtID).

Ако е:

Пита за потвърждение чрез прозорец „Are you sure?“

Ако потребителят натисне Yes:

Изпълнява **DELETE** заявка за изтриване на служителя с въведеното ID.

Обновява данните в табовете.

Ако ID не е въведено, изкарва съобщение, че трябва да се попълни.

8. Когато се излезе от контролата: UC_Employee_Leave

```
private void UC_Employee_Leave(object sender, EventArgs e)
```

Изчиства всички текстови полета, когато потребителят напусне този UserControl.

4.6. Модул "Администраторски панел"

За повишаване на сигурността и контрола върху системата, е създаден специален модул "Администраторски панел".

При влизане в системата потребителят въвежда своите данни за вход. Въз основа на зададената роля в базата данни, системата предоставя достъп само до съответните функционалности.

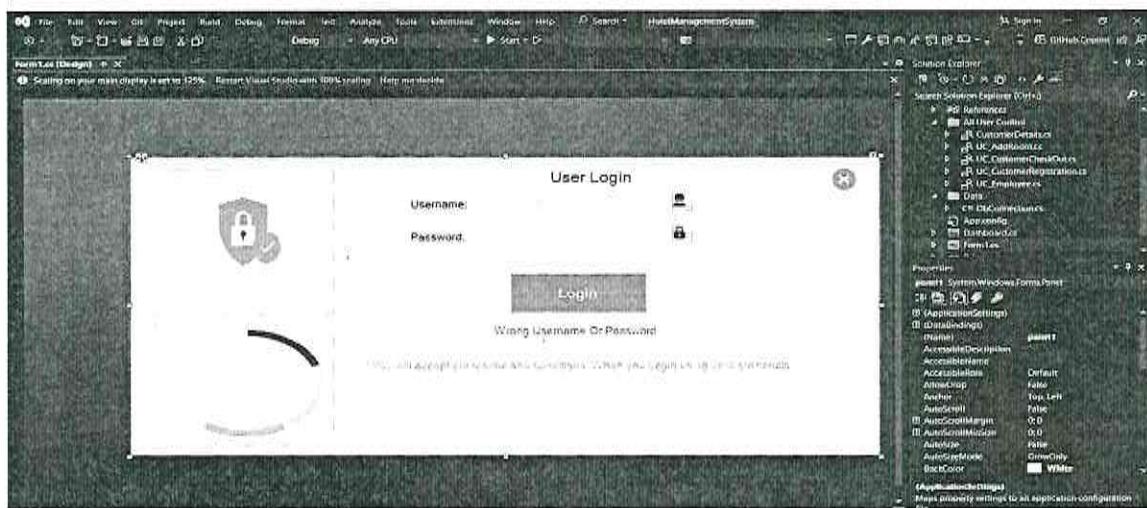
Администраторският панел включва следните възможности:

- Управление на потребителите на системата — добавяне, редактиране, изтриване;

- Преглед и промяна на служители;
- Промяна на параметри като типове стаи и цени;

Това решение позволява централизирано администриране и повишава сигурността на системата. Това се осъществява чрез Login формата(Form1.cs) и Dashboard.cs, който е свързан с останалите User Control:

Фигура 37



Това е главната форма за **логин** (вход) в приложението.

Основни части:

Пространства от имена:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.Common;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
```

Зареждат се всички необходими библиотеки за работа с форми, данни, текст, събития и компоненти в Windows Forms.

Конструктор:

```
public Form1()
{
    InitializeComponent();
}
```

Това инициализира всички елементи на формата, като бутони, текстови полета, етикети и други.

Събития:

1. При зареждане на формата: Form1_Load

```
private void Form1_Load(object sender, EventArgs e)
```

В момента този метод е празен.

Това означава, че когато формата се зарежда, **няма действия**, които да се изпълняват автоматично. (Може по-късно да се добавят настройки като например фокус върху текстовото поле.)

2. Клик на бутон "Exit": btnExit_Click

```
private void btnExit_Click(object sender, EventArgs e)
{
    Application.Exit();
}
```

Когато потребителят натисне бутона "Exit", приложението ще се затвори напълно.

3. Клик на бутон "Login": btnLogin_Click

csharp

Копиране Редактиране

```
private void btnLogin_Click(object sender, EventArgs e)
```

Това е най-важното събитие тук:

Проверява дали:

txtUsername.Text == "Kaloyan Ganev" && txtPassword.Text == "pass1234"

Тоест дали потребителското име и паролата са точни:

Потребителско име: **Kaloyan Ganev**

Парола: **pass1234**

Ако данните са правилни:

Скрива съобщението за грешка (labelError.Visible = false).

Създава нов обект от формата **Dashboard** (таблото за управление на хотела).

Скрива настоящата форма (**Form1**) чрез this.Hide().

Отваря **Dashboard** формата с ds.Show().

Ако данните са грешни:

Показва съобщението за грешка (labelError.Visible = true).

Изчиства текстовото поле за парола (txtPassword.Clear()), за да може потребителят да въведе нова парола.

Фигура 38



Това е основното меню на хотелската система, където имаш бутони за:

- Регистрация на клиенти
- Добавяне на стаи
- Напускане на клиенти (Check-Out)
- Подробности за клиенти
- Управление на служители

Основни части:

Конструктор:

```
public Dashboard()
{
    InitializeComponent();
}
```

Инициализира компонентите (бутони, панели, потребителски контроли и т.н.) на формата.

Събития:

1. Бутон Exit:

```
private void btnExit_Click(object sender, EventArgs e)
{
    Application.Exit();
}
```

При натискане на **Exit** бутона, **цялото приложение се затваря**.

2. Бутон Customer Registration:

```
private void btnCustomerRegistration_Click(object sender, EventArgs e)
{
    MovingPanel.Left = btnCustomerRegistration.Left + 18;
    uC_CustomerRegistration1.Visible = true;
    uC_CustomerRegistration1.BringToFront();
}
```

Премества **MovingPanel** (малък панел за подчертаване) под бутона "Customer Registration".

Прави **видим** потребителския контрол **uC_CustomerRegistration1** (екранът за регистрация на клиент).

Изкарва го най-отпред (**BringToFront()**).

3. Бутон Add Room:

```
private void btnAddRoom_Click(object sender, EventArgs e)
```

```
{  
    MovingPanel.Left = btnAddRoom.Left + 18;  
    uC_AddRoom1.Visible = true;  
    uC_AddRoom1.BringToFront();  
}
```

Същата логика като горе, но за контролата за **добавяне на стая**.

4. Бутон Check-Out:

```
private void btnCheckOut_Click(object sender, EventArgs e)  
{  
    MovingPanel.Left = btnCheckOut.Left + 18;  
    uC_CustomerCheckOut1.Visible = true;  
    uC_CustomerCheckOut1.BringToFront();  
}
```

Активира се контролата за **освобождаване на стая** (Check-Out на клиент).

5. Бутон Customer Details:

csharp

Копиране Редактиране

```
private void btnCustomerDetails_Click(object sender, EventArgs e)  
{  
    MovingPanel.Left = btnCustomerDetails.Left + 18;  
    customerDetails1.Visible = true;  
    customerDetails1.BringToFront();  
}
```

Зарежда се екранът за **подробности за клиентите**.

6. Бутон Employee Management:

```
private void btnEmployee_Click(object sender, EventArgs e)  
{  
    MovingPanel.Left = btnEmployee.Left + 18;
```

```
uC_Employee1.Visible = true;  
uC_Employee1.BringToFront();  
}  
Показва се формата за служители (регистрация и управление на работници).
```

7. Бутон Minimize:

```
private void btnMinimize_Click(object sender, EventArgs e)  
{  
    this.WindowState = FormWindowState.Minimized;  
}
```

Минимизира главния прозорец на системата.

Допълнителни неща:

Зареждане на Dashboard (Form Load):

```
private void Dashboard_Load(object sender, EventArgs e)  
{  
    uC_Employee1.Visible = false;  
    uC_AddRoom1.Visible = false;  
    uC_CustomerRegistration1.Visible = false;  
    btnAddRoom.PerformClick();  
}
```

Когато Dashboard формата се зареди:

Скрива всички потребителски контроли.

Симулира клик на бутона "**Add Room**", за да се покаже по подразбиране първият экран (добавяне на стаи).

5. ЗАКЛЮЧЕНИЕ

Разработката на система за управление на хотел е от ключово значение за оптимизацията на административните процеси в хотелиерския бизнес. Настоящият дипломен проект постигна поставените цели – създаване на приложение, което осигурява ефективно управление на стаи, резервации, клиенти и персонал.

Чрез използването на съвременни технологии като C# Windows Forms и Microsoft SQL Server бе изградена стабилна, интуитивна и надеждна система. По време на реализацията бяха преодолени редица технически предизвикателства, като особено внимание бе отделено на осигуряването на стабилна връзка с базата данни, валидацията на входните данни и изграждането на удобен за крайния потребител интерфейс.

Проведеното тестване потвърди правилната работа на всички функционалности и стабилността на системата при нормални условия на експлоатация. Постигнатите резултати доказват, че разработеното приложение може да намери реално практическо приложение в малки и средни хотели, допринасяйки за подобряване на обслужването и административната ефективност.

В бъдеще системата може да бъде надградена с допълнителни възможности, като онлайн резервации, мобилно приложение за персонала или автоматизирано известяване на клиенти, което допълнително би увеличило нейната приложимост и стойност.

6. ИЗПОЛЗВАНА ЛИТЕРАТУРА

1. Христова, М. Тодорова, Р. & Тодорова, П. (2012). Иновативен подход в обучението по бази от данни. Механика, транспорт, комуникации- научно списание, том 10, брой 3/3, ISSN 1312-3823.
2. Шиндер, Д. (2003). Компютърни мрежи. София, издателство „СофтПрес“.
3. ДОС (2018) за придобиване на квалификация по професията „Приложен програмист“, <https://www.navet.government.bg/bg/dosdv>
4. Pressman, Roger S., and David Lowe. *Web Engineering: A Practitioner's Approach*. New York: McGraw-Hill, 2012.
5. Sommerville, Ian. *Software Engineering*. 10th ed. Boston: Addison-Wesley, 2015.
6. Topalov, I., N. Shopov, T. Dessev & R. Ilarionov, "Computer system for characterization of cosmetic foams," 2019 IEEE XXVIII International Scientific Conference Electronics (ET), 2019, pp. 1-4, doi: 10.1109/ET.2019.8878508.
7. <https://learn.microsoft.com/en-us/aspnet/core/>
8. <https://learn.microsoft.com/en-us/dotnet/csharp/>