



**Технически университет  
гр. Варна**

**Факултет по изчислителна техника  
и автоматизация**



*Катедра  
„Софтуерни и интернет технологии”*

**Предмет: Технология на софтуерното  
производство**

*Летен семестър, 2022 год.*

**Софтуер за автобусни превози**

**Курсов проект  
на**

*Йордан Пламенов Христов, фк. номер 19621802,  
гр.1<sup>Б</sup>*

Срок за представяне: май, 2022г.

Водещ преподавател: /доц.В.Божикова/

# ЗАДАНИЕ

## за курсов проект по дисциплината ”Технология на софтуерното производство”

### Цел на курсовия проект:

Изисква се да се разработи „Система за запазване на билети за автобусни превози”.

### Функционални изисквания:

- Администраторът има достъп до всички функционалности на софтуера. Може да добавя, редактира и изтрива потребители, автобуси, превози и билети.
- Служителят да има достъп само до запазване и редактиране на билети.
- Потребителят да може само да запазва билети.
- Един потребител да може да запазва само по 1 билет за превоз.
- Един автобус да не участва в няколко превоза, ако времето им за пътуване се припокрива.
- Един билет да може да променя статуса си по всяко време от служителя (резервиран, закупен).
- Град на отпътуване и дестинация не трябва да бъдат едни и същи.
- При запазване на билет да се показват само предстоящи превози, а не всички, включително изминали такива.
- След запазване и редактиране на билет служителят/администраторът да има възможност да го принтира.
- Потребителят може да види стари и предстоящи превози, за които има запазен или закупен билет.
- Да е възможно търсене на превоз по зададен град на отпътуване или по дестинация
- Потребителите да имат възможност да се регистрират.
- Системата трябва да съхранява данни (в БД) за:
  - потребители (User),
  - автобуси (Bus)
  - превози (Transport) и
  - билети (Ticket).

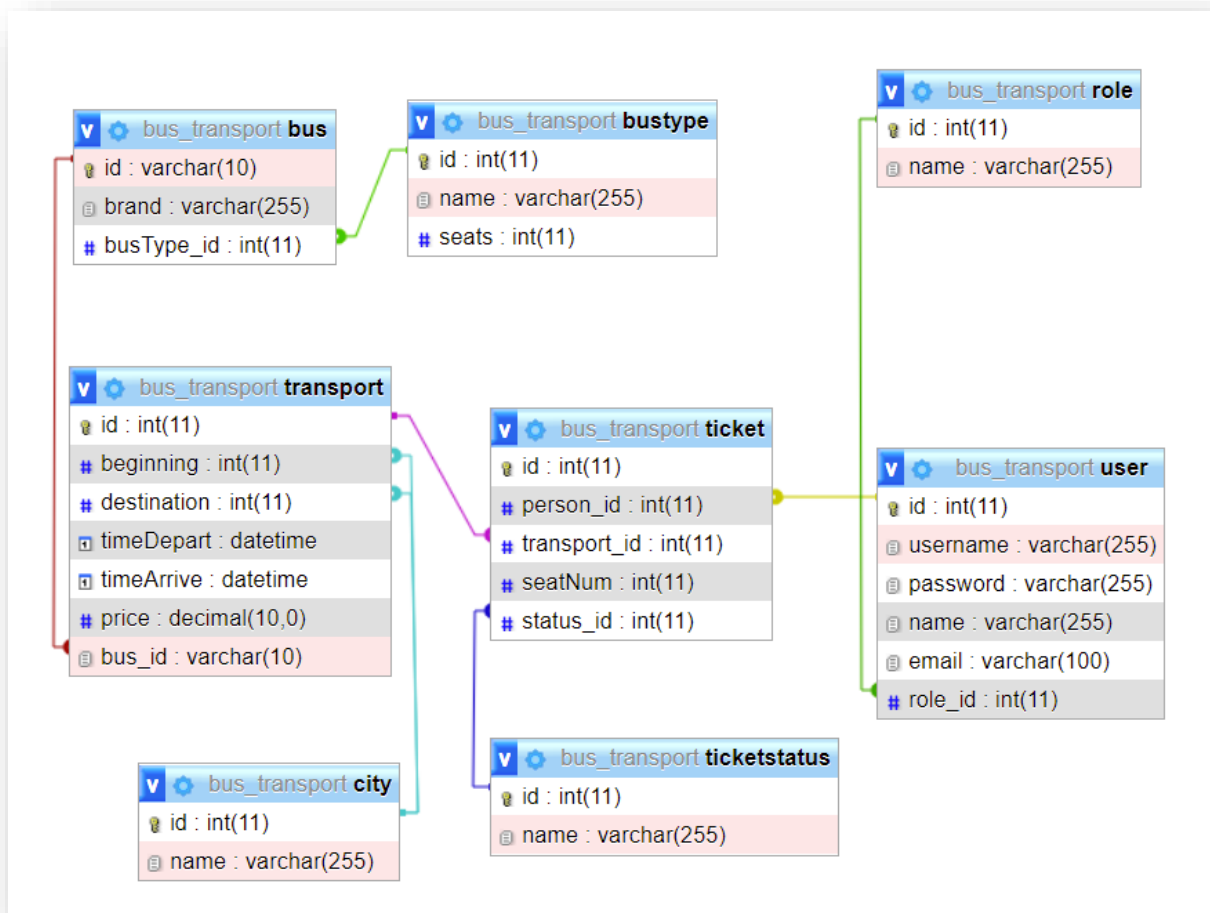
### Нефункционални изисквания:

- Използва се унифицирана форма за логване в системата от всички потребители (достъпът до системата се осъществява след въведено потребителско име и парола)
- За всяка от таблиците в БД се използва унифицирана форма за реализиране на стандартните операции: въвеждане, корекция, изтриване и търсене на данни.
- Генериране на необходимите справки: както на екран, така и на печат.
- Зареждане на формите в главен панел.

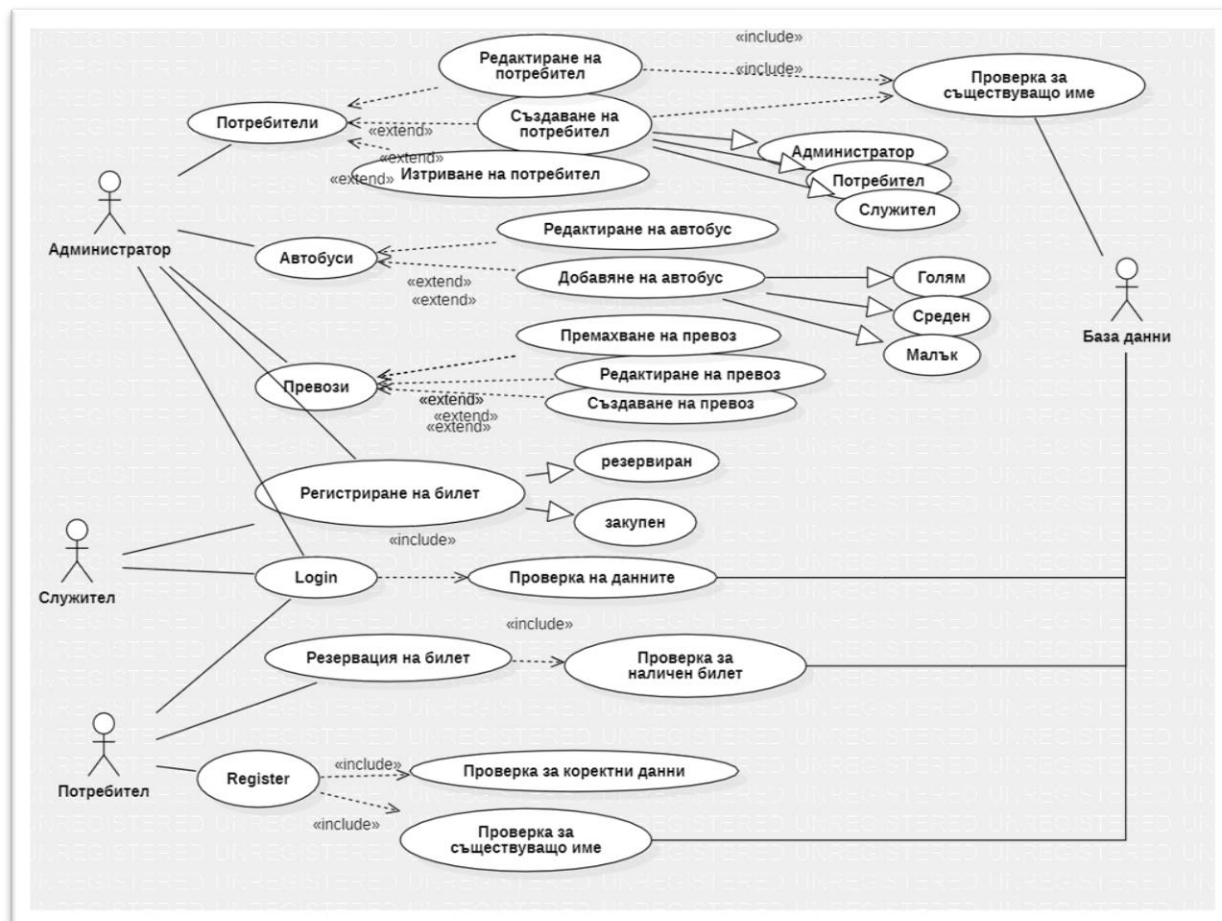
## Реализация

### Реализация на проекта

Entity Relationship Diagram (ERD):



### UML Use Case Diagram:



### Реализация на формите:

**LoginForm** – Предоставя контроли за въвеждане на потребителско име и парола. Проверява в базата данни за коректността на въведените данни. Ако намери такъв потребител със зададените му данни, го вписва и прехвърля в главния прозорец на програмата, в зависимост от това дали е администратор, служител или потребител(клиент). Налична е възможност за препращане към регистрационен прозорец.

**RegisterForm** - Предоставя контроли за въвеждане на име, потребителско име, имейл и парола. Изпълнява проверка на всяко едно поле дали покрива зададени изисквания (например паролата да е над 8 символа или имейлът да е валиден формат). Проверява дали въведеното потребителско име е заето. При коректни данни успешно се реализира регистрацията и се прехвърля в прозореца за вход.

**MenuForm** – Представява съвкупност от бутони, зареждащи дадени форми в главния панел. По подразбиране се зарежда Dashboard. Изписва се поздравителен надпис над бутоните. При

натискане на даден бутон той се уголемява и променя нюанса си. При натискане на друг бутон, предишният се връща в нормалния си размер.

**DashboardForm** – Зарежда списък, съдържащ статистическа информация, която се извлича от базата данни на приложението. Визуализира се информация за брой предстоящи превози, общ брой потребители, общ брой регистрирани автобуси и общ брой билети за предстоящите превози.

**UserForm** – Визуализира се таблица с информация за всички регистрирани потребители. Налични са контроли за записване на име, имейл, парола и избор на роля (администратор, служител, потребител). Налична е функция за проверка дали потребителското име, което се въвежда, вече съществува. Добавена е проверка дали всички полета са запълнени. При редакция и изтриване на потребител се взема id-то от текст контрола, който е забранен по подразбиране и се запълва с id на съществуващ потребител при цъкване на ред от таблицата. Преди изтриване на потребителя се извиква функция за изтриване на регистрираните на негово име билети.

**BusForm** - Визуализира се таблица с информация за всички регистрирани автобуси. Налични са контроли за записване на регистрационен номер, марка и избор на големината на превозното средство (малко, средно, голямо). Налична е функция за проверка дали регистрационния номер, който се въвежда, вече съществува. Добавена е проверка дали всички полета са запълнени. При редакция на автобус се взема id-то от текст контрола, който е забранен по подразбиране и се запълва с id на съществуващ автобус при цъкване на ред от таблицата.

**TicketForm** – Налични са две таблици, една, в която се попълват всички предстоящи транспорти и друга, в която се попълват всички регистрирани потребители. Проверката за предстоящ транспорт в заявката към базата данни се осъществява като се подава като условие датата на тръгване да е след сегашното време. При избиране на даден превоз се генерират динамични радио бутони. Те визуализират изгледа на седалките на автобуса като броят им и подредбата е съобразена с големината на автобуса. Чрез друга функция се извлича автобуса от селектирания превоз. При всяко изпълнение на функцията за генериране на седалки се извиква функция за изчистване на предишно генерираните. Налична е функция, която извлича заетите места. Чрез нея местата, които са заети, биват изобразени като радио бутони, които не могат да бъдат натиснати. По този начин се осъществява контрол и валидация на резервирането на билетите. Преди да се добави билета се проверява дали всеки един контрол е избран. Съдържа се и функция, която извлича цената на избрания превоз. Така се събира цялостна информация за билета, който ще бъде генериран. Функцията за принитране на билет се извиква, когато билетът е вече създаден. В нея автоматично е избрано името на текстовия файл, който ще бъде записан. Файлът се именува с номера на транспорта и номера на седалката.

**EditTicketForm** – В тази форма се реализират редактирането и изтриването на билет. Тук транспортите и потребителите се визуализират по различен начин – чрез падащ лист. Налични са бутони за показване, както на предстоящите превози, така и на всички останали. По същия начин има бутон и за билетите, които пък от своя страна са изобразени в таблица. Реализирани са функции като във формата за създаване на билет, които генерират, премахват и проверяват седалките на избрания превоз. По същия начин след успешно редактиране изскача прозорец за записване на билета в текстов файл.

**TransportForm** – Формата разполага с контроли за избор на дестинация и град на отпътуване. Градовете се извличат от базата данни и съхраняват в падащи списъци. За дата и час на отпътуване/пристигане има по два отделни контрола. В първият се сизвлича датата, а във вторият часът, който е въведен. В трети падащ списък се записват автобусите. Налична е функция, която проверява дали даден автобус е свободен в рамките на селектираното време. Ако не е, той не се показва в списъка и така се предотвратява некоректна регистрация на превоз. При всяко променяне на датите и часовете този списък се обновява. Налични са проверки градът на отпътуване и този на пристигане да не бъдат еднакви, както и датата и часа на отпътуване да не са преди датата и часа на пристигане. При коректно попълнени полета, функцията за добавяне или редакция на превоз се изпълнява успешно. Преди да се изпълни изтриването на даден транспорт се извиква функция за изтриване на всички билети за това пътуване.

**EmployeeMenuForm** – Аналогичен на MenuForm, но се показват ограничен брой бутони – за създаване и редактиране на билети. Генерира се, когато потребителят е с роля служител.

**UserMenuForm** – Аналогичен на останалите форми за меню. Генерира се, когато влезият профил е с роля потребител.

**UserDashboardForm** – По подразбиране зарежда списък с информация за закупени/резервирани билети на влезлия потребител, които са за предстоящи превози. Първият бутон зарежда първоначалната информация, която е по-подразбиране. Вторият бутон извиква функция, която извлича всички билети, които потребителят някога е закупил или резервирал и ги поставя в списъка.

**UserTicketForm** – Чрез тази форма се реализира запазването на билет за предстоящ превоз. В таблицата се визуализират само предстоящите пътувания. По аналогия с другите форми за билети се генерират местата на автобуса. Има контроли за търсене по град, като потребителят може да избере дали да търси по място на потегляне или по дестинация. Преди резервирането на билет се проверява дали този потребител не е запазил вече билет. По този начин се отстранява възможността даден човек да резервира повече от веднъж за превоз или да злоупотребява.

## **Ръководство за употреба**

### *За потребителите:*

Потребителят има възможност да се регистрира в системата. Въвежда име, потребителско име, имейл и парола и ако потребителското име не е заето, имейлът е коректен и паролата покрива изискванията, регистрацията се осъществява успешно.

При вход в системата на началната страница се зарежда списък със запазените билети на потребителя за предстоящи превози. Извежда се информация за маршрута, датата на потегляне, номерът на билета и цената му. Има бутон, който при натискане се извеждат всички билети на дадения потребител, които е закупувал/резервирал досега през времето.

Потребителят може да запази само по един билет на превоз. В таблицата се изобразяват само предстоящите превози. При натискане на реда най-отляво се маркира

транспорта и встрани се генерира списък от бутони за избор на място. Заетите седалки не са позволени за маркиране, потребителят може лесно да се ориентира за радпределението на местата в автобуса и да избере своя номер на седалката, която иска да резервира. Има възможност за търсене на превози по филтрите: град на отпътуване и град на дестинация, при което се визуализират превози само от зададения град. В случай, че иска да види всички превози, може да ги зареди повторно с бутона до този за търсене.

### *За служителите:*

При вход в системата на началната страница се зарежда списък с различни статистики като брой предстоящи превози, общ брой потребители, общ брой регистрирани автобуси и общ брой билети за предстоящите превози.

Служителят има възможността да регистрира билети, като може да ги маркира като резервирани или вече закупени. Селектира се дадения транспорт от таблицата, където са генерирани само предстоящите превози, като се натиска най-отляво на реда. По същия начин се избира и потребителят, за който се регистрира билета. Има възможност за търсене на потребител по потребителско име, като по този начин се улеснява търсенето, ако списъка с потребители е твърде дълъг. При натискане на реда на транспорта встрани се генерира списък от бутони за избор на място. Заетите седалки не са позволени за маркиране, служителят може да избира от останалите, като когато маркира и осъществи регистрацията на билета, даденото място ще се отбележи за заето. При регистрацията на билета изскача прозорец за записване на текстов файл, в който се съхранява информация за билета. Името на файла е предварително зададено, използвайки id на превоза и номера на седалката. Служителят може да запише файла или просто да затвори прозореца. И в двата случая, билетът ще е записан в системата. Генерираният текстов файл може да бъде отворен с Notepad и разпечатан.

Налична е опция за редактиране или изтриване на билет. Налична е възможност за показване на билети само за предстоящи превози. При натискане на реда най-отляво в таблицата за билетите се зарежда информацията за дадения билет, като се отбелязва превоза, мястото на седалката и дали то е запазено или закупено и статуса на билета. Служителят може да промени всеки един елемент и да редактира билета, като изскача прозорец за записване на текстов файл, в който аналогоично на регистрацията на билет се съхранява информация за билета. Възможно е показване на всички билети, резервирани досега, като това включва и тези на вече отминали превози. Те също могат да бъдат редактирани и изтривани. Има възможност за показване на всички превози в списъка с транспорти или визуализиране само на предстоящите такива.

### *За администраторите:*

Администраторът има достъп до всички функционалности на приложението. При вход в системата на началната страница се зарежда списък с различни статистики като брой превози за деня, общ брой потребители, общ брой регистрирани автобуси и общ брой билети за деня.

Наличен е прозорец за регистриране на потребители. Въвеждат се име, потребителско име, имейл и парола, като също така има опция за избор на роля (администратор, служител или потребител). Отдясно се визуализира таблица с данните за всички регистрирани профили.

При натискане на реда на профила встрани се запълват полетата за данните и администраторът има възможността да редактира или изтрие селектирания потребител.

В прозореца за автобуси се осъществява регистрирането и редактирането на автобус. Въвежда се регистрационен номер и марка на автомобила и се селектира неговата големина. Предвидени са три типа автобуси - голям, среден и малък, като всеки има точно определен брой седалки и разпределение. Вдясно е налична таблица на вече регистрираните автобуси. При натискане на реда отляво се запълват данните за селектираното превозно средство. Администраторът може да редактира тези данни и да ги запази.

Администраторът има възможността да регистрира билети, като може да ги маркира като резервирани или вече закупени. Селектира се дадения транспорт от таблицата, където са генерирани само предстоящите превози, като се натиска най-отляво на реда. По същия начин се избира и потребителят, за който се регистрира билета. Има възможност за търсене на потребител по потребителско име, като по този начин се улеснява търсенето, ако списъка с потребители е твърде дълъг. При натискане на реда на транспорта встрани се генерира списък от бутони за избор на място. Заетите седалки не са позволени за маркиране, администраторът може да избира от останалите, като когато маркира и осъществи регистрацията на билета, даденото място ще се отбележи за заето. При регистрацията на билета изскача прозорец за записване на текстов файл, в който се съхранява информация за билета. Името на файла е предварително зададено, изпозвайки id на превоза и номера на седалката. Администраторът може да запише файла или просто да затвори прозореца. И в двата случая, билетът ще е записан в системата. Генерираният текстов файл може да бъде отворен с Notepad и разпечатан.

Налична е опция за редактиране или изтриване на билет. Налична е възможност за показване на билети само за предстоящи превози. При натискане на реда най-отляво в таблицата за билетите се зарежда информацията за дадения билет, като се отбелязва превоза, мястото на седалката и дали то е запазено или закупено и статуса на билета. Администраторът може да промени всеки един елемент и да редактира билета, като изскача прозорец за записване на текстов файл, в който аналогично на регистрацията на билет се съхранява информация за билета. Възможно е показване на всички билети, резервирани досега, като това включва и тези на вече отминали превози. Те също могат да бъдат редактирани и изтривани. Има възможност за показване на всички превози в списъка с транспорти или визуализиране само на предстоящите такива.

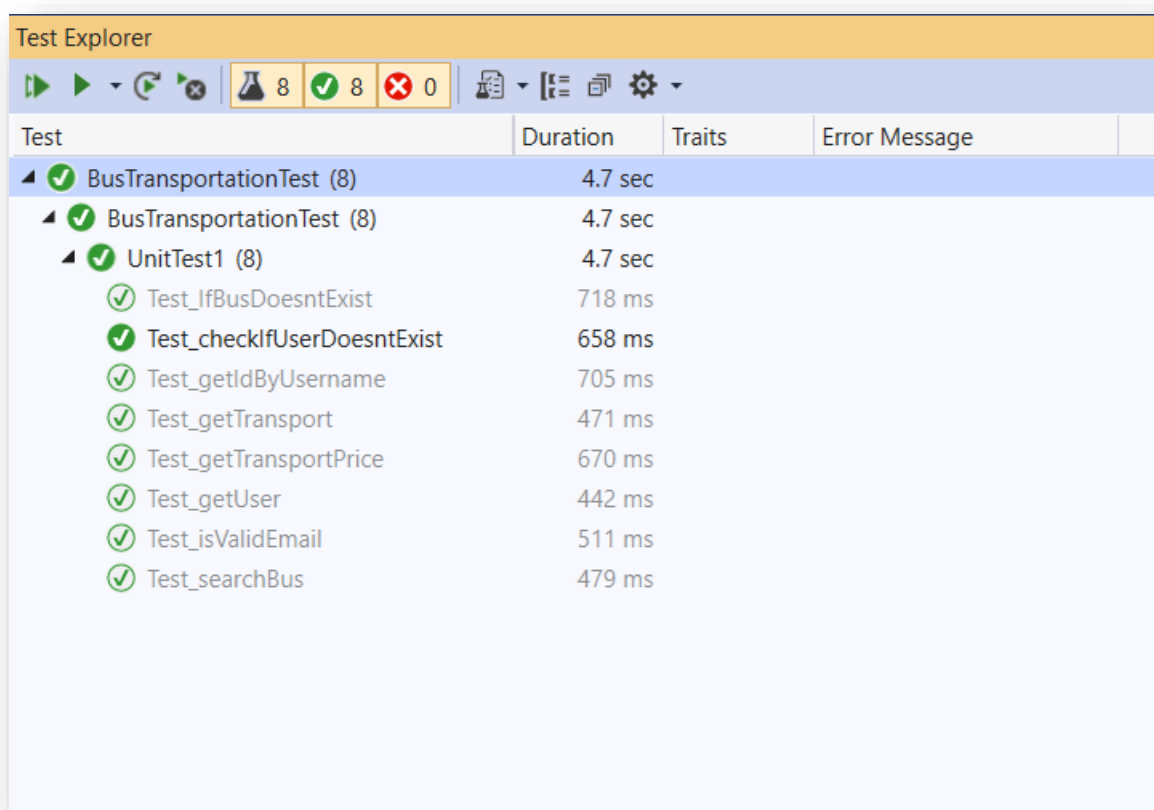
При натискане на бутона за транспорти се зарежда таблица с всички превози и полета за запълване. От падащият списък се селектира град на отпътуване и град на пристигане. Не може да бъде въведен един и същи град в двата списъка. Има две полета за избиране на дата на отпътуване. В първото се селектира датата на отпътуване, като може да се използва календара при цъкване върху контрола. Във второто трябва ръчно да бъде въведен часът на отпътуване. По същият начин е избора за време на пристигане. По подразбиране са селектирани селектиран текущата дата и текущият час. Времето и датата на отпътуване не може да бъде след това на пристигане, в противен случай се генерира съобщение за неуспешна регистрация на превоз. Наличен е още един падащ списък, където се избира автобусът, който ще бъде използван за дадения транспорт. Автобусите се актуализират спрямо въведените дати и часове на отпътуване/пристигане. Зареждат се само тези автомобили, които не са предназначени за друг превоз в същия период от време. Налично е и поле за въвеждане на цена на транспорта, могат да бъдат въвеждани само цифри и символът "." за десетични числа. От таблицата може да се селектира даден превоз чрез натискане отляво на реда след



което се попълват полетата за данни без датите. Датите и часовете трябва да бъдат въведени ръчно. Администраторът може да променя данните и да редактира или изтрива селектирания превоз.

## **Тестове и изводи**

### *Тестове на функциите*



The screenshot shows the Test Explorer window with a toolbar at the top containing icons for running tests, a summary bar showing 8 passed, 8 succeeded, and 0 failed tests, and a table of test results.

Test	Duration	Traits	Error Message
▲ ✓ BusTransportationTest (8)	4.7 sec		
▲ ✓ BusTransportationTest (8)	4.7 sec		
▲ ✓ UnitTest1 (8)	4.7 sec		
✓ Test_IfBusDoesntExist	718 ms		
✓ Test_checkIfUserDoesntExist	658 ms		
✓ Test_getIdByUsername	705 ms		
✓ Test_getTransport	471 ms		
✓ Test_getTransportPrice	670 ms		
✓ Test_getUser	442 ms		
✓ Test_isValidEmail	511 ms		
✓ Test_searchBus	479 ms		

Тествани са успешно функциите, които връщат даден резултат и не са пряко свързани с контролите. Комуникацията с базата данни се осъществява коректно и са тестовете са изпълнени коректно.

### *Изводи и възможности за развитие*

Софтуерът е предназначен да се използва от три типа потребители – администратор, служител и клиент, като всеки от тях има различни права на достъп. Реализира се симулация на истинско управление на транспортни превози в територията на България. Програмирани са главните аспекти на подобен софтуер, целящ запазване на билети за пътувания.

Наличен е широк набор от подобрения, които могат да се извършат. Могат да се добавят спирки между градовете и възможност един превоз да не бъде само между две

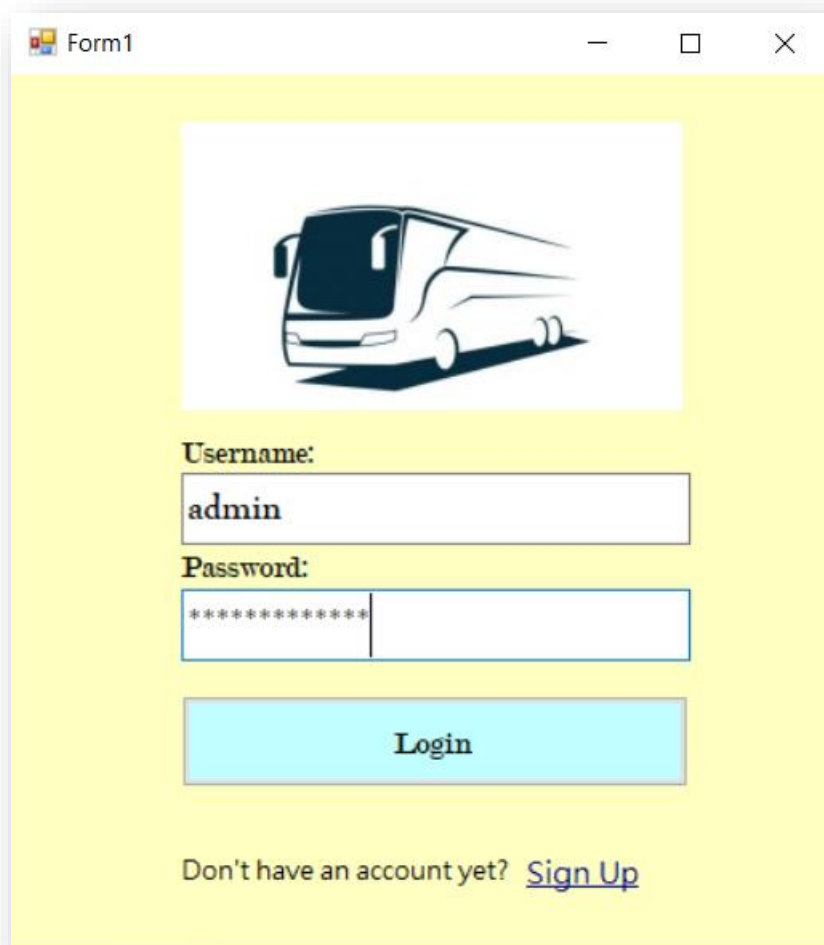
локации. Възможна е реализацията на повече полета за търсене, чрез които потребителите по-лесно да намират дадена информация. Таблиците, в които се генерира информацията, могат да бъдат заменени с друг по-добър контрол, който да осигурява по-добра видимост. Главен недостатък от страна на потребителската сигурност, който може да бъде разрешен, е липсата на хеширане на паролите. Възможна е оптимизация на кода, като се използва вече готов код чрез наследяване между класове.

## Използвана литература


- Youtube, Stackoverflow, W3Schools, CodeProject
- Решени задачи по дисциплината ТСП

## Програма

### *Login Form*



Form1



Username:

Password:

Login

Don't have an account yet? [Sign Up](#)

```
public partial class LoginForm : Form
```

```

{
    private MySqlConnection conn;
    public static String user = "";
    public LoginForm()
    {
        string connection =
"datasource=127.0.0.1;port=3306;username=root;password=;database=bus_transport";
        conn = new MySqlConnection(connection);

        InitializeComponent();
    }

    private void button1_Click(object sender, EventArgs e)
    {
        login();
    }

    private void login()
    {
        string query = $"Select * from User where username=@Username and
password=@Password";
        try
        {
            conn.Open();
            MySqlCommand cmd = new MySqlCommand(query, conn);
            cmd.Parameters.AddWithValue("@Username", usernameTb.Text);
            cmd.Parameters.AddWithValue("@Password", passTb.Text);
            MySqlDataAdapter msda = new MySqlDataAdapter(cmd);
            DataTable dt = new DataTable();
            msda.Fill(dt);
            cmd.ExecuteNonQuery();
            if(dt.Rows.Count > 0)
            {
                user = usernameTb.Text;
                string role = dt.Rows[0].ItemArray[5].ToString();
                if (role == "1")
                {
                    MenuForm menu = new MenuForm();
                    menu.Show();
                }
                if (role == "2")
                {
                    EmployeeMenuForm menu = new EmployeeMenuForm();
                    menu.Show();
                }
                if (role == "3")
                {
                    UserMenuForm menu = new UserMenuForm();
                    menu.Show();
                }
                this.Hide();
            }
            else
            {
                messageLbl.Visible = true;
                usernameTb.Text = "";
                passTb.Text = "";
            }
            conn.Close();
        }
        catch(Exception e)

```

```

        {
            MessageBox.Show("Query error:" + e.Message);
            conn.Close();
        }
    }

    private void linkLabel1_LinkClicked(object sender,
    LinkLabelLinkClickedEventArgs e)
    {
        Form1 registerForm = new Form1();
        registerForm.Show();
        this.Hide();
    }
}

```

### ***RegisterForm***

The screenshot shows a Windows application window titled "Form1" with a yellow background. The window contains a registration form with the following elements:

- Title:** "Register" in a large, black, serif font.
- Name:** A text input field containing "Yordan".
- Username:** A text input field containing "dancho123".
- Email:** A text input field containing "dancho123@gmail.com".
- Password:** A text input field containing "\*\*\*\*\*".
- Repeat password:** A text input field containing "\*\*\*\*\*".
- Sign Up:** A light blue button at the bottom of the form.

```

public partial class Form1 : Form
{
    private MySqlConnection conn;
    public Form1()

```

```

{
    string connection =
"datasource=127.0.0.1;port=3306;username=root;password=;database=bus_transport";
    conn = new MySqlConnection(connection);

    InitializeComponent();
}

private void button1_Click(object sender, EventArgs e)
{
    if (conn.State == ConnectionState.Open)
    {
        conn.Close();
    }
    register();
}

public bool validate()
{
    if (nameTb.Text.Length < 3)
    {
        messageLbl.Visible = true;
        messageLbl.Text = "Name is too short.";
        return false;
    }
    else if (usernameTb.Text.Length < 3)
    {
        messageLbl.Visible = true;
        messageLbl.Text = "Username is too short.";
        return false;
    }
    else if (IsValidEmail(emailTb.Text) == false)
    {
        messageLbl.Visible = true;
        messageLbl.Text = "Incorrect email format.";
        return false;
    }
    else if (passTb.Text.Length < 8)
    {
        messageLbl.Visible = true;
        messageLbl.Text = "Password must be 8 or more symbols.";
        return false;
    }
    else if (!passTb.Text.Equals(repeatPassTb.Text))
    {
        messageLbl.Visible = true;
        messageLbl.Text = "Passwords do not match.";
        return false;
    }
    else if (checkIfUserDoesntExist(usernameTb.Text) == false)
    {
        messageLbl.Visible = true;
        messageLbl.Text = "User already exists.";
        return false;
    }
    return true;
}

public bool IsValidEmail(string email)
{
    var trimmedEmail = email.Trim();

    if (trimmedEmail.EndsWith("."))

```

```

        {
            return false;
        }
        try
        {
            var addr = new System.Net.Mail.MailAddress(email);
            return addr.Address == trimmedEmail;
        }
        catch
        {
            return false;
        }
    }

    public bool checkIfUserDoesntExist(string username)
    {
        try
        {
            conn.Open();
            using (var sqlCommand = new MySqlCommand("SELECT * FROM User WHERE
username = '" + username + "'", conn))
            {
                MySqlDataReader reader = sqlCommand.ExecuteReader();
                if (reader.HasRows)
                {
                    reader.Close();
                    reader.Dispose();
                    conn.Close();
                    return false;
                }
                else
                {
                    reader.Close();
                    reader.Dispose();
                    conn.Close();
                    return true;
                }
            }
        }
        catch (Exception e)
        {
            MessageBox.Show("Query error:" + e.Message);
        }
        return false;
    }

    private void register()
    {
        if (validate()==true) {
            string query = $"INSERT INTO User (id, name, email, username,
password, role_id) VALUES" +
            "(NULL, @Name, @Email, @Username, @Password, @Role)";
            try
            {
                conn.Open();
                MySqlCommand cmd = new MySqlCommand(query, conn);
                cmd.Parameters.AddWithValue("@Name", nameTb.Text);
                cmd.Parameters.AddWithValue("@Email", emailTb.Text);
                cmd.Parameters.AddWithValue("@Username", usernameTb.Text);
                cmd.Parameters.AddWithValue("@Password", passTb.Text);
            }
        }
    }

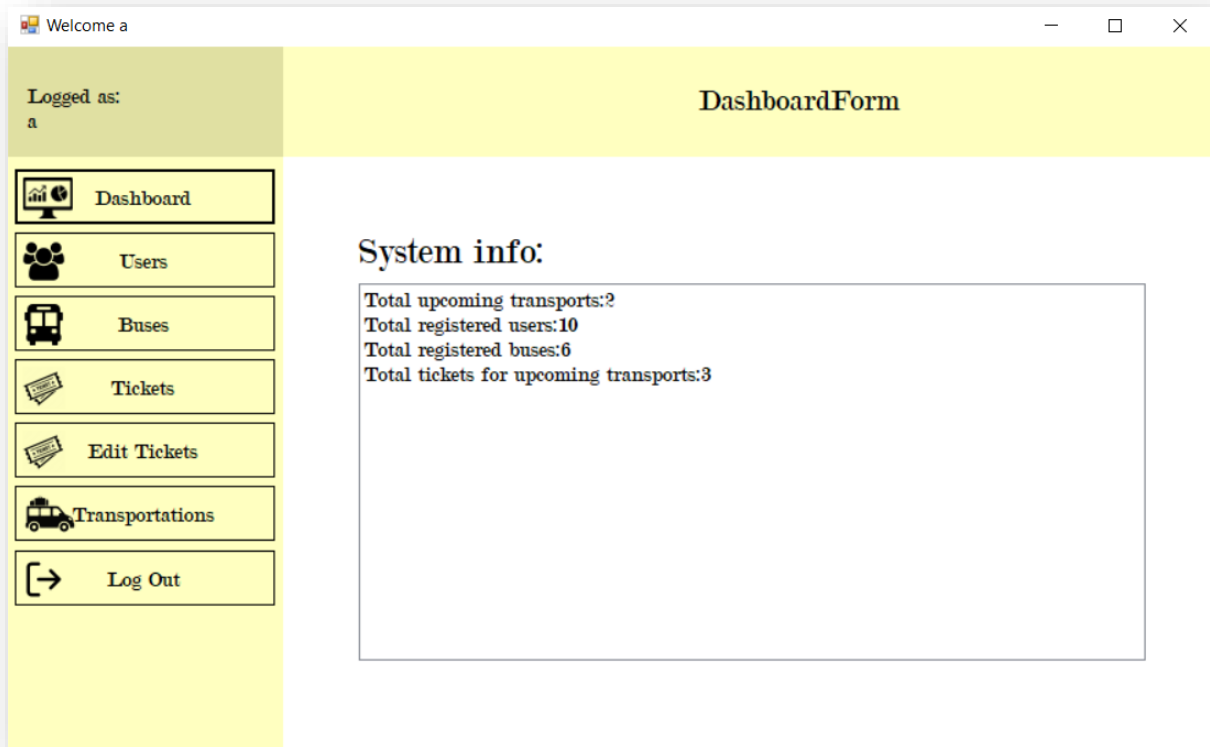
```

```

        cmd.Parameters.AddWithValue("@Role", 3);
        cmd.ExecuteNonQuery();
        conn.Close();
        LoginForm lg = new LoginForm();
        lg.Show();
        this.Close();
    }
    catch (Exception e)
    {
        MessageBox.Show("Query error:" + e.Message);
    }
}
}
}

```

### MenuForm



```

public partial class MenuForm : Form
{
    private Button currentButton;
    private Form activeForm;

    public MenuForm()
    {
        InitializeComponent();
    }

    private void MenuForm_Load(object sender, EventArgs e)

```

```

{
    welcomeLbl.Text = LoginForm.user;
    this.Text = "Welcome " + LoginForm.user;
    loadDashboard(new DashboardForm(), sender);
}

private void loadDashboard(Form childForm, object sender)
{
    if (activeForm != null)
    {
        activeForm.Close();
    }
    activeForm = childForm;
    childForm.TopLevel = false;
    childForm.FormBorderStyle = FormBorderStyle.None;
    childForm.Dock = DockStyle.Fill;
    this.panelDesktop.Controls.Add(childForm);
    this.panelDesktop.Tag = childForm;
    childForm.BringToFront();
    childForm.Show();
    lblTitle.Text = childForm.Text;
}

private void busesBt_Click(object sender, EventArgs e)
{
    openChildForm(new BusForm(), sender);
}

private void logoutBt_Click(object sender, EventArgs e)
{
    LoginForm lg = new LoginForm();
    lg.Show();
    this.Hide();
}

private void dashboardBt_Click(object sender, EventArgs e)
{
    openChildForm(new DashboardForm(), sender);
}

private void usersBt_Click(object sender, EventArgs e)
{
    openChildForm(new UserForm(), sender);
}

private void ticketsBt_Click(object sender, EventArgs e)
{
    openChildForm(new TicketForm(), sender);
}

private void transportationsBt_Click(object sender, EventArgs e)
{
    openChildForm(new TransportForm(), sender);
}

private void activateButton(object sender)
{
    if (sender != null)
    {
        if (currentButton != (Button)sender)
        {

```



```

        disableButton();
        currentButton = (Button)sender;
        currentButton.BackColor = Color.FromArgb(255, 255, 222);
        currentButton.Font = new System.Drawing.Font("Modern No. 20", 13F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)0));
    }
}

private void disableButton()
{
    foreach(Control previousBtn in panelSide.Controls)
    {
        if(previousBtn.GetType() == typeof(Button))
        {
            previousBtn.BackColor = Color.FromArgb(255, 255, 192);
            previousBtn.Font = new System.Drawing.Font("Modern No. 20", 12F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)0));
        }
    }
}

private void openChildForm(Form childForm, object sender)
{
    if(activeForm != null)
    {
        activeForm.Close();
    }
    activateButton(sender);
    activeForm = childForm;
    childForm.TopLevel = false;
    childForm.FormBorderStyle = FormBorderStyle.None;
    childForm.Dock = DockStyle.Fill;
    this.panelDesktop.Controls.Add(childForm);
    this.panelDesktop.Tag = childForm;
    childForm.BringToFront();
    childForm.Show();
    lblTitle.Text = childForm.Text;
}

private void editTicketBt_Click(object sender, EventArgs e)
{
    openChildForm(new EditTicket(), sender);
}
}

```

## EmployeeMenuForm

The screenshot shows a Windows application window titled "Welcome e". The window has a yellow header bar with the text "DashboardForm" on the right. On the left, there is a vertical sidebar with four buttons: "Dashboard" (with a monitor icon), "Tickets" (with a ticket icon), "Edit Tickets" (with a ticket icon), and "Log Out" (with a right arrow icon). The main area of the window displays "System info:" followed by a box containing the following text: "Total upcoming transports:2", "Total registered users:10", "Total registered buses:6", and "Total tickets for upcoming transports:3".

```
public partial class EmployeeMenuForm : Form
{
    private Button currentButton;
    private Form activeForm;
    public EmployeeMenuForm()
    {
        InitializeComponent();
    }

    private void logoutBt_Click(object sender, EventArgs e)
    {
        LoginForm lg = new LoginForm();
        lg.Show();
        this.Hide();
    }

    private void editTicketBt_Click(object sender, EventArgs e)
    {
        openChildForm(new EditTicket(), sender);
    }

    private void ticketsBt_Click(object sender, EventArgs e)
    {
        openChildForm(new TicketForm(), sender);
    }

    private void dashboardBt_Click(object sender, EventArgs e)
    {
        openChildForm(new DashboardForm(), sender);
    }
}
```

```

private void loadDashboard(Form childForm, object sender)
{
    if (activeForm != null)
    {
        activeForm.Close();
    }
    activeForm = childForm;
    childForm.TopLevel = false;
    childForm.FormBorderStyle = FormBorderStyle.None;
    childForm.Dock = DockStyle.Fill;
    this.panelDesktop.Controls.Add(childForm);
    this.panelDesktop.Tag = childForm;
    childForm.BringToFront();
    childForm.Show();
    lblTitle.Text = childForm.Text;
}

private void EmployeeMenuForm_Load(object sender, EventArgs e)
{
    welcomeLbl.Text = LoginForm.user;
    this.Text = "Welcome " + LoginForm.user;
    loadDashboard(new DashboardForm(), sender);
}

private void openChildForm(Form childForm, object sender)
{
    if (activeForm != null)
    {
        activeForm.Close();
    }
    activateButton(sender);
    activeForm = childForm;
    childForm.TopLevel = false;
    childForm.FormBorderStyle = FormBorderStyle.None;
    childForm.Dock = DockStyle.Fill;
    this.panelDesktop.Controls.Add(childForm);
    this.panelDesktop.Tag = childForm;
    childForm.BringToFront();
    childForm.Show();
    lblTitle.Text = childForm.Text;
}

private void activateButton(object sender)
{
    if (sender != null)
    {
        if (currentButton != (Button)sender)
        {
            disableButton();
            currentButton = (Button)sender;
            currentButton.BackColor = Color.FromArgb(255, 255, 222);
            currentButton.Font = new System.Drawing.Font("Modern No. 20", 13F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)0));
        }
    }
}

private void disableButton()
{
    foreach (Control previousBtn in panelSide.Controls)
    {

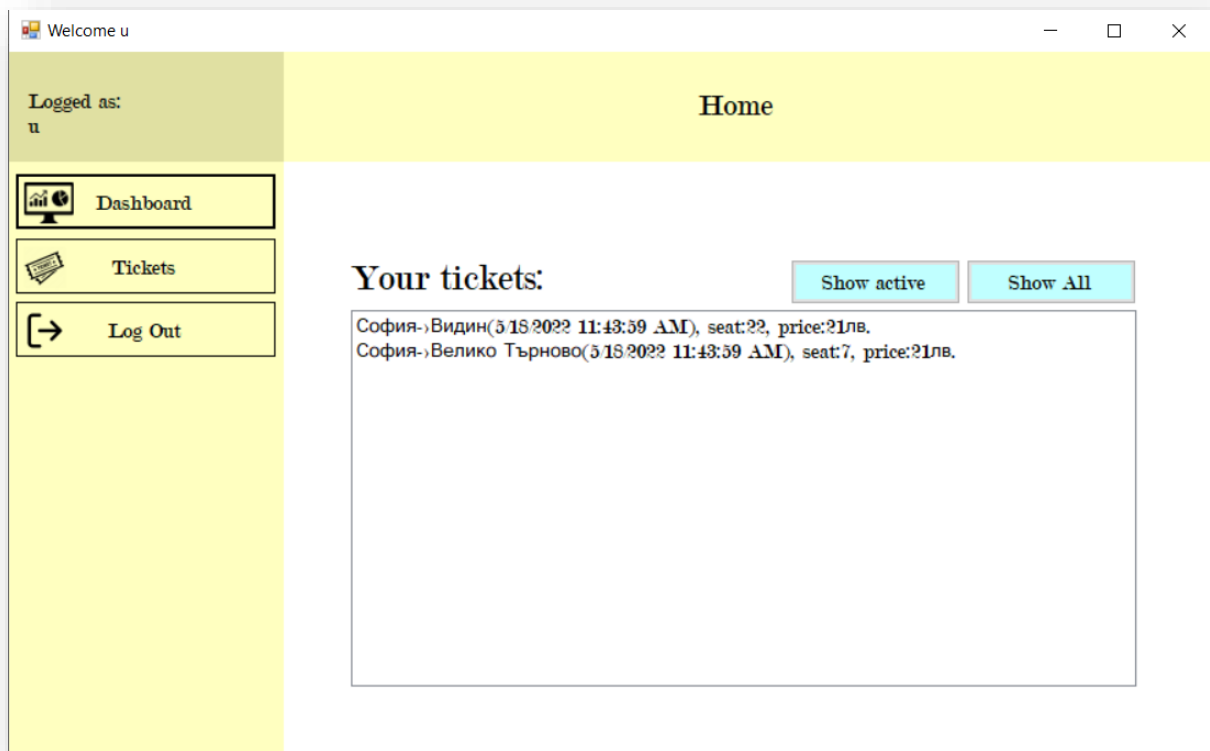
```

```

        if (previousBtn.GetType() == typeof(Button))
        {
            previousBtn.BackColor = Color.FromArgb(255, 255, 192);
            previousBtn.Font = new System.Drawing.Font("Modern No. 20", 12F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)0));
        }
    }
}

```

### UserMenuForm



```

public partial class UserMenuForm : Form
{
    private Button currentButton;
    private Form activeForm;
    public UserMenuForm()
    {
        InitializeComponent();
    }

    private void UserMenuForm_Load(object sender, EventArgs e)
    {
        welcomeLbl.Text = LoginForm.user;
        this.Text = "Welcome " + LoginForm.user;
        loadDashboard(new UserDashboardForm(welcomeLbl.Text), sender);
    }

    private void dashboardBt_Click(object sender, EventArgs e)

```

```

{
    openChildForm(new UserDashboardForm(welcomeLbl.Text), sender);
}

private void ticketsBt_Click(object sender, EventArgs e)
{
    openChildForm(new UserTicketForm(welcomeLbl.Text), sender);
}

private void logoutBt_Click(object sender, EventArgs e)
{
    LoginForm lg = new LoginForm();
    lg.Show();
    this.Hide();
}

private void loadDashboard(Form childForm, object sender)
{
    if (activeForm != null)
    {
        activeForm.Close();
    }
    activeForm = childForm;
    childForm.TopLevel = false;
    childForm.FormBorderStyle = FormBorderStyle.None;
    childForm.Dock = DockStyle.Fill;
    this.panelDesktop.Controls.Add(childForm);
    this.panelDesktop.Tag = childForm;
    childForm.BringToFront();
    childForm.Show();
    lblTitle.Text = childForm.Text;
}

private void openChildForm(Form childForm, object sender)
{
    if (activeForm != null)
    {
        activeForm.Close();
    }
    activateButton(sender);
    activeForm = childForm;
    childForm.TopLevel = false;
    childForm.FormBorderStyle = FormBorderStyle.None;
    childForm.Dock = DockStyle.Fill;
    this.panelDesktop.Controls.Add(childForm);
    this.panelDesktop.Tag = childForm;
    childForm.BringToFront();
    childForm.Show();
    lblTitle.Text = childForm.Text;
}

private void activateButton(object sender)
{
    if (sender != null)
    {
        if (currentButton != (Button)sender)
        {
            disableButton();
            currentButton = (Button)sender;
            currentButton.BackColor = Color.FromArgb(255, 255, 222);
            currentButton.Font = new System.Drawing.Font("Modern No. 20", 13F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)0));

```

```

        }
    }
}

private void disableButton()
{
    foreach (Control previousBtn in panelSide.Controls)
    {
        if (previousBtn.GetType() == typeof(Button))
        {
            previousBtn.BackColor = Color.FromArgb(255, 255, 192);
            previousBtn.Font = new System.Drawing.Font("Modern No. 20", 12F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)0));
        }
    }
}
}

```

```

public partial class DashboardForm : Form
{
    private MySqlConnection conn;
    public DashboardForm()
    {
        string connection =
"datasource=127.0.0.1;port=3306;username=root;password=;database=bus_transport";
        conn = new MySqlConnection(connection);
        InitializeComponent();
    }

    private void DashboardForm_Load(object sender, EventArgs e)
    {
        displayList();
    }

    private void displayList()
    {
        listBox1.Items.Add("Total upcoming
transports:"+getTodaysAmountOfTransports().ToString());
        listBox1.Items.Add("Total registered users:" +
getUsersCount().ToString());
        listBox1.Items.Add("Total registered buses:" +
getBusesCount().ToString());
        listBox1.Items.Add("Total tickets for upcoming transports:" +
getTicketsForTheDayCount());
    }

    private int getTodaysAmountOfTransports()
    {
        int count = 0;
        string query = "SELECT id FROM Transport where timeDepart >=
CURRENT_TIMESTAMP";
        try
        {
            conn.Open();
            MySqlCommand cmd = new MySqlCommand(query, conn);
            MySqlDataReader reader = cmd.ExecuteReader();
            while (reader.Read())
            {
                count++;
            }
        }
    }
}

```

```

        }
        conn.Close();
    }
    catch (Exception e)
    {
        MessageBox.Show("Query error:" + e.Message);
        conn.Close();
    }
    return count;
}

private int getUsersCount()
{
    int count = 0;
    string query = "SELECT id FROM User";
    try
    {
        conn.Open();
        MySqlCommand cmd = new MySqlCommand(query, conn);
        MySqlDataReader reader = cmd.ExecuteReader();
        while (reader.Read())
        {
            count++;
        }
        conn.Close();
    }
    catch (Exception e)
    {
        MessageBox.Show("Query error:" + e.Message);
        conn.Close();
    }
    return count;
}

private int getBusesCount()
{
    int count = 0;
    string query = "SELECT id FROM Bus";
    try
    {
        conn.Open();
        MySqlCommand cmd = new MySqlCommand(query, conn);
        MySqlDataReader reader = cmd.ExecuteReader();
        while (reader.Read())
        {
            count++;
        }
        conn.Close();
    }
    catch (Exception e)
    {
        MessageBox.Show("Query error:" + e.Message);
        conn.Close();
    }
    return count;
}

private int getTicketsForTheDayCount()
{
    int count = 0;
    string query = "SELECT tt.id FROM Ticket tt join Transport t on
tt.transport_id=t.id where t.timeDepart >= CURRENT_TIMESTAMP";

```

```

try
{
    conn.Open();
    MySqlCommand cmd = new MySqlCommand(query, conn);
    MySqlDataReader reader = cmd.ExecuteReader();
    while (reader.Read())
    {
        count++;
    }
    conn.Close();
}
catch (Exception e)
{
    MessageBox.Show("Query error:" + e.Message);
    conn.Close();
}
return count;
}
}

```

### EditTicketForm

**EditTicket**

## Edit ticket:

**Select transport:** Show active Show all

1
1: Благоевград->Кюстендил(5/1/2022 12:12:51 ▾)

**Select person:**

1
1: user1 ▾

**Ticket type:** Reserved ▾

**Pick a seat:**

☐ 1   ☐ 9   ☐ 17  
☐ 2   ☐ 10   ☐ 18  
☐ 3   ☐ 11   ☐ 19  
☐ 4   ☐ 12   ☐ 20  
☐ 5   ☐ 13   ☐ 21  
☐ 6   ☐ 14   ☐ 22  
☐ 7   ☐ 15   ☐ 23  
☐ 8   ☐ 16   ☐ 24

Edit  
Delete

**Selected ticket:** Show active Show all

	id	User	transp
▶	35	u	5
	37	u	6
	38	nana123	5
*			

<>

```

public partial class EditTicket : Form
{
    private MySqlConnection conn;

```



```

public EditTicket()
{
    string connection =
"datasource=127.0.0.1;port=3306;username=root;password=;database=bus_transport";
    conn = new MySqlConnection(connection);

    InitializeComponent();
}

private void EditTicket_Load(object sender, EventArgs e)
{
    loadActiveTickets();
    loadUsers();
    personCb.SelectedIndex = 0;
    loadTransports();
    transportCb.SelectedIndex = 0;

    ticketCb.Items.Add("Reserved");
    ticketCb.Items.Add("Purchased");
    ticketCb.SelectedIndex = 0;
}

private void loadTickets()
{
    string query = "SELECT t.id, u.username as \"User\", t.transport_id,
t.seatNum, s.name as \"Status\" FROM Ticket t join User u on t.person_id = u.id join
Ticketstatus s on t.status_id = s.id";
    try
    {
        conn.Open();
        MySqlDataAdapter adapter = new MySqlDataAdapter(query, conn);
        DataTable dt = new DataTable();
        adapter.Fill(dt);
        usersDV.DataSource = dt;
        conn.Close();
    }
    catch (Exception exception)
    {
        MessageBox.Show("Query error:" + exception.Message);
        conn.Close();
    }
}

public string getIdByUsername(string username)
{
    string query = "SELECT id FROM User where username='" + username + "'";
    string id = "";
    try
    {
        conn.Open();
        MySqlCommand cmd = new MySqlCommand(query, conn);
        MySqlDataReader reader = cmd.ExecuteReader();
        while (reader.Read())
        {
            id = reader["id"].ToString();
        }
        conn.Close();
    }
    catch (Exception e)
    {
        MessageBox.Show("Query error:" + e.Message);
        conn.Close();
    }
}

```

```

    }
    return id;
}

private void usersDV_RowHeaderMouseClick(object sender,
DataGridViewCellMouseEventArgs e)
{
    idTb.Text = usersDV.Rows[e.RowIndex].Cells[0].Value.ToString();
    string transportId, userId, status;
    int seat;
    transportId = usersDV.Rows[e.RowIndex].Cells[2].Value.ToString();
    userId =
getIdByUsername(usersDV.Rows[e.RowIndex].Cells[1].Value.ToString());
    seat = (int)usersDV.Rows[e.RowIndex].Cells[3].Value;
    status = usersDV.Rows[e.RowIndex].Cells[4].Value.ToString();

    if (status == "reserved") ticketCb.SelectedIndex = 0;
    if (status == "purchased") ticketCb.SelectedIndex = 1;

    transportCb.SelectedIndex =
transportCb.FindStringExact(getTransport(transportId));
    personCb.SelectedIndex = personCb.FindStringExact(getUser(userId));

    int type = searchBus(transportId);

    if (type == 1) generateSeats(8, 120, seat);
    if (type == 2) generateSeats(10, 200, seat);
    if (type == 3) generateSeats(14, 200, seat);
}

private void loadUsers()
{
    string query = "SELECT id, username FROM User";
    try
    {
        conn.Open();
        MySqlCommand cmd = new MySqlCommand(query, conn);
        MySqlDataReader reader = cmd.ExecuteReader();
        while (reader.Read())
        {
            personCb.Items.Add(reader["id"]+": "+reader["username"]);
        }
        conn.Close();
    }
    catch (Exception e)
    {
        MessageBox.Show("Query error:" + e.Message);
        conn.Close();
    }
}

private void loadTransports()
{
    string query = "SELECT t.id as \"Id\", c.name as \"Beginning\", cc.name as
\"Destination\", t.timeDepart as \"Departure\", t.timeArrive, price, bus_id FROM
Transport t join City c on t.beginning = c.id join City cc on t.destination = cc.id";
    try
    {
        conn.Open();
        MySqlCommand cmd = new MySqlCommand(query, conn);

```

```

        MySqlDataReader reader = cmd.ExecuteReader();
        while (reader.Read())
        {
            transportCb.Items.Add(reader["Id"] + ": " + reader["Beginning"] +
"->" + reader["Destination"] + "(" + reader["Departure"] + ")");
        }
        conn.Close();
    }
    catch (Exception e)
    {
        MessageBox.Show("Query error:" + e.Message);
        conn.Close();
    }
}

public string getUser(string id)
{
    string query = "SELECT id, username FROM User where id="+id;
    string result = "";
    try
    {
        conn.Open();
        MySqlCommand cmd = new MySqlCommand(query, conn);
        MySqlDataReader reader = cmd.ExecuteReader();
        while (reader.Read())
        {
            result = reader["id"] + ": " + reader["username"];
        }
        conn.Close();
    }
    catch (Exception e)
    {
        MessageBox.Show("Query error:" + e.Message);
        conn.Close();
    }
    return result;
}

public string getTransport(string id)
{
    string query = "SELECT t.id as \"Id\", c.name as \"Beginning\", cc.name as
\"Destination\", t.timeDepart as \"Departure\", t.timeArrive, price, bus_id FROM
Transport t join City c on t.beginning = c.id join City cc on t.destination = cc.id
where t.id="+id;
    string result = "";
    try
    {
        conn.Open();
        MySqlCommand cmd = new MySqlCommand(query, conn);
        MySqlDataReader reader = cmd.ExecuteReader();
        while (reader.Read())
        {
            result = reader["Id"] + ": " + reader["Beginning"] + "->" +
reader["Destination"] + "(" + reader["Departure"] + ")";
        }
        conn.Close();
    }
    catch (Exception e)
    {
        MessageBox.Show("Query error:" + e.Message);
        conn.Close();
    }
}

```

```

        return result;
    }

    public int searchBus(string transportId)
    {
        int i = 0;
        string query = "SELECT b.busType_id FROM Bus b join Transport t on
b.id=t.bus_id where t.id=" + transportId;
        try
        {
            conn.Open();
            MySqlCommand cmd = new MySqlCommand(query, conn);
            MySqlDataReader reader = cmd.ExecuteReader();
            while (reader.Read())
            {
                i = (int)reader["busType_id"];
            }
            conn.Close();
        }
        catch (Exception e)
        {
            MessageBox.Show("Query error:" + e.Message);
            conn.Close();
        }
        return i;
    }

    public bool checkInfo()
    {
        if(idTb.Text.Length < 1)
        {
            MessageBox.Show("Select a ticket");
            return false;
        }
        if (transportIdTb.Text.Length < 1)
        {
            MessageBox.Show("Choose a transport");
            return false;
        }
        if (this.Controls.OfType<RadioButton>().FirstOrDefault(r => r.Checked) ==
null)
        {
            MessageBox.Show("Choose a seat");
            return false;
        }
        if (personIdTb.Text.Length < 1)
        {
            MessageBox.Show("Choose a person");
            return false;
        }
        return true;
    }

    private void editBt_Click(object sender, EventArgs e)
    {
        if (checkInfo() == true)
        {
            string statusId = (ticketCb.SelectedIndex + 1).ToString();
            string seat = this.Controls.OfType<RadioButton>().FirstOrDefault(r =>
r.Checked).Text;

```

```

        string query = $"Update Ticket SET person_id = @Person, transport_id =
@Transport, seatNum = @Seat, status_id = @Status WHERE ID = @Id";
        try
        {
            conn.Open();
            MySqlCommand cmd = new MySqlCommand(query, conn);
            cmd.Parameters.AddWithValue("@Person", personIdTb.Text);
            cmd.Parameters.AddWithValue("@Transport", transportIdTb.Text);
            cmd.Parameters.AddWithValue("@Seat", seat);
            cmd.Parameters.AddWithValue("@Status", statusId);
            cmd.Parameters.AddWithValue("@Id", idTb.Text);
            cmd.ExecuteNonQuery();
            conn.Close();
            printTicket();
            MessageBox.Show("Ticket successfully edited.");
            RadioButton rb = (RadioButton)Controls[seat];
            rb.Enabled = false;
            rb.Checked = false;
            loadTickets();
        }
        catch (Exception exception)
        {
            MessageBox.Show("Query error:" + exception.Message);
            conn.Close();
        }
    }

    public string getTransportPrice(String transportId)
    {
        string price = "";
        string query = "SELECT price FROM Transport where id = '" + transportId +
""";
        try
        {
            conn.Open();
            MySqlCommand cmd = new MySqlCommand(query, conn);
            MySqlDataReader reader = cmd.ExecuteReader();
            while (reader.Read())
            {
                price = reader["price"].ToString();
            }
            conn.Close();
        }
        catch (Exception e)
        {
            MessageBox.Show("Query error:" + e.Message);
            conn.Close();
        }
        return price;
    }

    private void printTicket()
    {
        string seat = this.Controls.OfType<RadioButton>().FirstOrDefault(r =>
r.Checked).Text;
        //suzdavane na dialogovata kutiq za zapis
        SaveFileDialog saveFileDialog1 = new SaveFileDialog();
        string filePath = saveFileDialog1.FileName;
        saveFileDialog1.Filter = "txt files (*.txt)|*.txt";//|All files
(*.*)|*.*";
    }

```

```

        saveFileDialog1.FilterIndex = 2;
        saveFileDialog1.FileName = "ticket#route" + transportIdTb.Text + "seat" +
seat;

        saveFileDialog1.RestoreDirectory = true;
        if (saveFileDialog1.ShowDialog() == DialogResult.OK)
        {
            filePath = saveFileDialog1.FileName;
        }

        string path = transportCb.GetItemText(transportCb.SelectedItem), date =
path;

        path = path.Substring(0, path.IndexOf("("));
        date = date.Substring(date.IndexOf("(") + 1);

        string[] ticket = {"Билет на
"+personCb.GetItemText(personCb.SelectedItem), "Разписание: " + path, "Отпътуване -
Пристигане " + date,
                            "Цена:
"+getTransportPrice(transportIdTb.Text).ToString() + "лв.", "Място: "+
seat, "Статус на билета: "+ticketCb.Text};

        if (filePath != "")
        {
            using (StreamWriter sw = new StreamWriter(filePath, false))
            {
                foreach (string s in ticket)
                {
                    sw.WriteLine(s);
                }
                sw.Close();
            }
        }

        private void personCb_SelectedIndexChanged(object sender, EventArgs e)
        {
            string selected = personCb.GetItemText(personCb.SelectedItem);
            if (selected.Length > 0)
            {
                string stringBeforeChar = selected.Substring(0,
selected.IndexOf(":"));
                personIdTb.Text = stringBeforeChar;
            }
        }

        private void transportCb_SelectedIndexChanged(object sender, EventArgs e)
        {
            string selected = transportCb.GetItemText(transportCb.SelectedItem);
            if (selected.Length > 0)
            {
                string stringBeforeChar = selected.Substring(0,
selected.IndexOf(":"));
                transportIdTb.Text = stringBeforeChar;

                int type = searchBus(transportIdTb.Text);

                if (type == 1) generateSeats(8, 120, 0);
                if (type == 2) generateSeats(10, 200, 0);
            }
        }

```

```

        if (type == 3) generateSeats(14, 200, 0);
    }
}

public List<int> getTakenSeats(string transportId)
{
    List<int> seats = new List<int>();

    string query = "SELECT seatNum FROM Ticket where transport_id = '" +
transportId + "'";
    try
    {
        conn.Open();
        MySqlCommand cmd = new MySqlCommand(query, conn);
        MySqlDataReader reader = cmd.ExecuteReader();
        while (reader.Read())
        {
            seats.Add((int)reader["seatNum"]);
        }
        conn.Close();
    }
    catch (Exception e)
    {
        MessageBox.Show("Query error:" + e.Message);
        conn.Close();
    }

    return seats;
}

public void clearSeats()
{
    foreach (Control item in this.Controls.OfType<RadioButton>().ToList())
    {
        this.Controls.Remove(item);
    }
}

public void generateSeats(int seats, int yEnd, int s)
{
    List<int> takenSeats = getTakenSeats(transportIdTb.Text);
    clearSeats();
    int plus = 0;
    int x = 110;
    int y = 155;
    int seat = 1;
    for (int j = 40; j <= yEnd; j += 40)
    {
        if (j == 120) j += 40;
        for (int i = 1; i <= seats; i++)
        {
            RadioButton rdo = new RadioButton();
            rdo.Name = seat.ToString();
            rdo.Text = seat.ToString();
            rdo.ForeColor = Color.Black;
            rdo.Location = new Point(x + j, y + plus);
            rdo.AutoSize = true;
            if (takenSeats.Contains(seat)) rdo.Enabled = false;
            if (seat == s) rdo.Checked = true;
            this.Controls.Add(rdo);
            plus = plus + 20;
            seat++;
        }
    }
}

```

```

        }
        plus = 0;
    }
}

private void button1_Click(object sender, EventArgs e)
{
    if (idTb.TextLength > 0)
    {
        string query = "DELETE FROM Ticket WHERE id='" + idTb.Text + "'";
        MySqlCommand command = new MySqlCommand(query, conn);
        conn.Open();
        command.CommandText = query;
        command.Connection = conn;
        command.ExecuteNonQuery(); //zaduljitelni, inache nqma da se izpulni
zaqvkata

        MessageBox.Show("Ticket removed.");
        conn.Close();
        loadActiveTickets();
    }
    else
    {
        MessageBox.Show("Please select a ticket");
    }
}

private void showAllBt_Click(object sender, EventArgs e)
{
    loadTickets();
}

private void showActiveBt_Click(object sender, EventArgs e)
{
    loadActiveTickets();
}

private void loadActiveTickets()
{
    string query = "SELECT t.id, u.username as \"User\", t.transport_id,
t.seatNum, s.name as \"Status\" FROM Ticket t"+
        " join User u on t.person_id = u.id join Ticketstatus s on t.status_id
= s.id join Transport tr on t.transport_id=tr.id where tr.timeDepart >=
CURRENT_TIMESTAMP";
    try
    {
        conn.Open();
        MySqlDataAdapter adapter = new MySqlDataAdapter(query, conn);
        DataTable dt = new DataTable();
        adapter.Fill(dt);
        usersDV.DataSource = dt;
        conn.Close();
    }
    catch (Exception exception)
    {
        MessageBox.Show("Query error:" + exception.Message);
        conn.Close();
    }
}

private void button3_Click(object sender, EventArgs e)
{
    transportCb.Items.Clear();
}

```



```

        loadTransports();
        transportCb.SelectedIndex = 0;
    }

    private void button4_Click(object sender, EventArgs e)
    {
        transportCb.Items.Clear();
        loadActiveTransports();
        transportCb.SelectedIndex = 0;
    }

    private void loadActiveTransports()
    {
        string query = "SELECT t.id as \"Id\", c.name as \"Beginning\", cc.name as \"Destination\", t.timeDepart as \"Departure\", t.timeArrive, price, bus_id"+
            " FROM Transport t join City c on t.beginning = c.id join City cc on t.destination = cc.id where t.timeDepart >= CURRENT_TIMESTAMP";
        try
        {
            conn.Open();
            MySqlCommand cmd = new MySqlCommand(query, conn);
            MySqlDataReader reader = cmd.ExecuteReader();
            while (reader.Read())
            {
                transportCb.Items.Add(reader["Id"] + ": " + reader["Beginning"] +
                    "->" + reader["Destination"] + "(" + reader["Departure"] + ")");
            }
            conn.Close();
        }
        catch (Exception e)
        {
            MessageBox.Show("Query error:" + e.Message);
            conn.Close();
        }
    }
}

```

## TicketForm

**TicketForm**

**Select transport:**

	id	Beginning	Destination
	5	София	Видин
▶	6	София	Велико Т...
✱			

6    София - Велико Търново | 5/18/2022 11:43:59 AM - :

**Select person:**

	id	username	name
▶	1	user1	Ivan
	2	nana123	Nana

1    user1 | Ivan | Ivan

**Ticket type:**

Purchased(2) ▼

Buy

**Pick a seat:**

☐ 1    ☐ 9    ☐ 17  
☐ 2    ☐ 10    ☐ 18  
☐ 3    ☐ 11    ☐ 19  
☐ 4    ☐ 12    ☐ 20  
☐ 5    ☐ 13    ☐ 21  
☐ 6    ☒ 14    ☐ 22  
☐ 7    ☐ 15    ☐ 23  
☐ 8    ☐ 16    ☐ 24

```

public partial class TicketForm : Form
{
    private MySqlConnection conn;
    public TicketForm()
    {
        string connection =
"datasource=127.0.0.1;port=3306;username=root;password=;database=bus_transport";
        conn = new MySqlConnection(connection);

        InitializeComponent();
    }

    private void TicketForm_Load(object sender, EventArgs e)
    {
        ticketCb.Items.Add("Reserved(1)");
        ticketCb.Items.Add("Purchased(2)");
        ticketCb.SelectedIndex = 0;
        loadUsers();
        loadTransports();
    }

    private void loadUsers()
    {
        string query = "SELECT id, username, name, email FROM User";
    }
}

```

```

        try
        {
            conn.Open();
            MySqlDataAdapter adapter = new MySqlDataAdapter(query, conn);
            DataTable dt = new DataTable();
            adapter.Fill(dt);
            personDV.DataSource = dt;
            conn.Close();
        }
        catch (Exception e)
        {
            MessageBox.Show("Query error:" + e.Message);
            conn.Close();
        }
    }

    private void loadTransports()
    {
        DateTime today = DateTime.Today;
        string query = "SELECT t.id, c.name as \"Beginning\", cc.name as \"Destination\", t.timeDepart, t.timeArrive, price, bus_id FROM Transport t" +
            " join City c on t.beginning = c.id join City cc on t.destination = cc.id where t.timeDepart >= CURRENT_TIMESTAMP";
        try
        {
            conn.Open();
            MySqlDataAdapter adapter = new MySqlDataAdapter(query, conn);
            DataTable dt = new DataTable();
            adapter.Fill(dt);
            transportDV.DataSource = dt;
            conn.Close();
        }
        catch (Exception e)
        {
            MessageBox.Show("Query error:" + e.Message);
            conn.Close();
        }
    }

    private void transportDV_RowHeaderMouseClick(object sender, DataGridViewCellMouseEventArgs e)
    {
        string transport;
        transport = transportDV.Rows[e.RowIndex].Cells[1].Value.ToString()
            + " - " + transportDV.Rows[e.RowIndex].Cells[2].Value.ToString() + " | "
            + transportDV.Rows[e.RowIndex].Cells[3].Value.ToString()
            + " - " + transportDV.Rows[e.RowIndex].Cells[4].Value.ToString();
        transportTb.Text = transport;
        int type =
            searchBus(transportDV.Rows[e.RowIndex].Cells[6].Value.ToString());
        transportIdTb.Text =
            transportDV.Rows[e.RowIndex].Cells[0].Value.ToString();

        if (type == 1) generateSeats(8, 120);
        if (type == 2) generateSeats(10, 200);
        if (type == 3) generateSeats(14, 200);
    }

    public int searchBus(string busId)
    {
        int i = 0;
        string query = "SELECT busType_id FROM Bus where id = '" + busId + "'";
    }

```

```

        try
        {
            conn.Open();
            MySqlCommand cmd = new MySqlCommand(query, conn);
            MySqlDataReader reader = cmd.ExecuteReader();
            while (reader.Read())
            {
                i = (int)reader["busType_id"];
            }
            conn.Close();
        }
        catch (Exception e)
        {
            MessageBox.Show("Query error:" + e.Message);
            conn.Close();
        }
        return i;
    }

    private void personDV_RowHeaderMouseClick(object sender,
DataGridViewCellEventArgs e)
    {
        string person;
        person = personDV.Rows[e.RowIndex].Cells[1].Value.ToString()
            + " | " + personDV.Rows[e.RowIndex].Cells[2].Value.ToString() + " | "
+ personDV.Rows[e.RowIndex].Cells[2].Value.ToString();
        personTb.Text = person;
        personIdTb.Text = personDV.Rows[e.RowIndex].Cells[0].Value.ToString();
    }

    public void generateSeats(int seats, int yEnd)
    {
        List<int> takenSeats = getTakenSeats(transportIdTb.Text);
        clearSeats();
        int plus = 0;
        int x = 340;
        int y = 50;
        int seat = 1;
        for (int j = 40; j <= yEnd; j += 40)
        {
            if (j == 120) j += 40;
            for (int i = 1; i <= seats; i++)
            {
                RadioButton rdo = new RadioButton();
                rdo.Name = seat.ToString();
                rdo.Text = seat.ToString();
                rdo.ForeColor = Color.Black;
                rdo.Location = new Point(x + j, y + plus);
                rdo.AutoSize = true;
                if (takenSeats.Contains(seat)) rdo.Enabled = false;
                this.Controls.Add(rdo);
                plus = plus + 20;
                seat++;
            }
            plus = 0;
        }
    }

    public List<int> getTakenSeats(string transportId)
    {
        List<int> seats = new List<int>();
    }

```

```

        string query = "SELECT seatNum FROM Ticket where transport_id = '" +
transportId + "'";
        try
        {
            conn.Open();
            MySqlCommand cmd = new MySqlCommand(query, conn);
            MySqlDataReader reader = cmd.ExecuteReader();
            while (reader.Read())
            {
                seats.Add((int)reader["seatNum"]);
            }
            conn.Close();
        }
        catch (Exception e)
        {
            MessageBox.Show("Query error:" + e.Message);
            conn.Close();
        }

        return seats;
    }

    public void clearSeats()
    {
        foreach (Control item in this.Controls.OfType<RadioButton>().ToList())
        {
            this.Controls.Remove(item);
        }
    }

    public bool checkInfo()
    {
        if (trasnportTb.Text.Length < 1)
        {
            MessageBox.Show("Choose a transport");
            return false;
        }
        if (this.Controls.OfType<RadioButton>().FirstOrDefault(r => r.Checked) ==
null)
        {
            MessageBox.Show("Choose a seat");
            return false;
        }
        if (personTb.Text.Length < 1)
        {
            MessageBox.Show("Choose a person");
            return false;
        }
        return true;
    }

    private void addBt_Click(object sender, EventArgs e)
    {
        if (checkInfo() == true)
        {
            string statusId = (ticketCb.SelectedIndex+1).ToString();
            string seat = this.Controls.OfType<RadioButton>().FirstOrDefault(r =>
r.Checked).Text;

            string query = $"INSERT INTO Ticket (person_id, transport_id, seatNum,
status_id) VALUES" +

```

```

        "@Person, @Transport, @Seat, @Status)";
    try
    {
        conn.Open();
        MySqlCommand cmd = new MySqlCommand(query, conn);
        cmd.Parameters.AddWithValue("@Person", personIdTb.Text);
        cmd.Parameters.AddWithValue("@Transport", transportIdTb.Text);
        cmd.Parameters.AddWithValue("@Seat", seat);
        cmd.Parameters.AddWithValue("@Status", statusId);
        cmd.ExecuteNonQuery();
        conn.Close();
        printTicket();
        MessageBox.Show("Ticket successfully created.");
        RadioButton rb = (RadioButton)Controls[seat];
        rb.Enabled = false;
        rb.Checked = false;
    }
    catch (Exception exception)
    {
        MessageBox.Show("Query error:" + exception.Message);
        conn.Close();
    }
}

private string getTransportPrice(String transportId)
{
    string price = "";
    string query = "SELECT price FROM Transport where id = '" + transportId +
""";
    try
    {
        conn.Open();
        MySqlCommand cmd = new MySqlCommand(query, conn);
        MySqlDataReader reader = cmd.ExecuteReader();
        while (reader.Read())
        {
            price = reader["price"].ToString();
        }
        conn.Close();
    }
    catch (Exception e)
    {
        MessageBox.Show("Query error:" + e.Message);
        conn.Close();
    }
    return price;
}

private void printTicket()
{
    string seat = this.Controls.OfType<RadioButton>().FirstOrDefault(r =>
r.Checked).Text; //suzdavane na dialogovata kutiq za zapis
    SaveFileDialog saveFileDialog1 = new SaveFileDialog();
    string filePath = saveFileDialog1.FileName;
    saveFileDialog1.Filter = "txt files (*.txt)|*.txt";//|All files
(*.*)|*.*";
    saveFileDialog1.FilterIndex = 2;
    saveFileDialog1.FileName = "ticket#route" + transportIdTb.Text + "seat" +
seat;
    saveFileDialog1.RestoreDirectory = true;
    if (saveFileDialog1.ShowDialog() == DialogResult.OK)

```

```

{
    filePath = saveFileDialog1.FileName;
}

string path = trasnportTb.Text, date = trasnportTb.Text;
path = path.Substring(0, path.IndexOf("|"));
date = date.Substring(date.IndexOf("|")+1);
string[] ticket = {"Билет на: "+personTb.Text, "Разписание: " + path,
"Отпътуване - Пристигане:" + date,
                    "Цена:
"+getTransportPrice(transportIdTb.Text).ToString() + "лв.", "Място: " +
                    seat, "Статус на билета: "+ticketCb.Text};

if (filePath != "")
{
    using (StreamWriter sw = new StreamWriter(filePath, false))
    {
        foreach (string s in ticket)
        {
            sw.WriteLine(s);
        }
        sw.Close();
    }
}
}
private void ticketCb_SelectedIndexChanged(object sender, EventArgs e)
{
    if (ticketCb.SelectedIndex == 0) addBt.Text = "Book";
    if (ticketCb.SelectedIndex == 1) addBt.Text = "Buy";
}
}

```

### TransportForm

## TransportForm

### Transport registration:

Selected transport:

	id	Beginning	Destination
▶	1	Благоевград	Кюстендил
	2	Благоевград	Кюстендил
	3	Русе	Силистра
	5	София	Видин
	6	София	Велико Търново
	9	Монтана	Видин
*			

**Beginning:** 
**Destination:**

**Time departure:**

**Time arrival:**

**Select bus:** 
**Price:**

```

public partial class TransportForm : Form
{
    private MySqlConnection conn;
    DateTime departDateTime;
    DateTime arrivalDateTime;
    HashSet<string> buses;
    HashSet<string> allBuses;
    public TransportForm()
    {
        string connection =
"datasource=127.0.0.1;port=3306;username=root;password=;database=bus_transport";
        conn = new MySqlConnection(connection);

        InitializeComponent();
    }

    private void TransportForm_Load(object sender, EventArgs e)
    {
        displayTransports();
        loadCities(destinationCb);
        loadCities(beginningCb);
        destinationCb.SelectedIndex = 0;
        beginningCb.SelectedIndex = 0;

        departDateTime = departureDate.Value.Date + departureTime.Value.TimeOfDay;
        arrivalDateTime = arrivalDate.Value.Date + arrivalTime.Value.TimeOfDay;
        allBuses = getAllBuses();
        loadBuses();
    }

    private void addBt_Click(object sender, EventArgs e)
    {
        if (checkInfo() == true)
        {
            int beginningCityId = beginningCb.SelectedIndex + 1;
            int destinationCityId = destinationCb.SelectedIndex + 1;
            string busId = busCb.SelectedItem.ToString();
            string query = $"INSERT INTO Transport (beginning, destination,
timeDepart, timeArrive, price, bus_id) VALUES" +
                "(@Beginning, @Destination, @TimeDepart, @TimeArrive, @Price,
@BusId)";
            try
            {
                conn.Open();
                MySqlCommand cmd = new MySqlCommand(query, conn);
                cmd.Parameters.AddWithValue("@Beginning", beginningCityId);
                cmd.Parameters.AddWithValue("@Destination", destinationCityId);
                cmd.Parameters.AddWithValue("@TimeDepart", departDateTime);
                cmd.Parameters.AddWithValue("@TimeArrive", arrivalDateTime);
                cmd.Parameters.AddWithValue("@Price", priceTb.Text);
                cmd.Parameters.AddWithValue("@BusId", busId);
                cmd.ExecuteNonQuery();
                conn.Close();
                MessageBox.Show("Transport successfully added.");
                displayTransports();
                loadBuses();
            }
            catch (Exception exception)
            {
                MessageBox.Show("Query error:" + exception.Message);
            }
        }
    }
}

```



```

        conn.Close();
    }
}

private void editBt_Click(object sender, EventArgs e)
{
    if (idTb.TextLength > 0)
    {
        if (checkInfo() == true)
        {
            int beginningCityId = beginningCb.SelectedIndex + 1;
            int destinationCityId = destinationCb.SelectedIndex + 1;
            string busId = busCb.SelectedItem.ToString();
            string query = $"Update Transport SET beginning = @Beginning,
destination = @Destination, timeDepart = @TimeDepart, timeArrive = @TimeArrive, price
= @Price, bus_id = @BusId WHERE ID= @Id";
            Console.Write(query);
            try
            {
                MySqlCommand cmd = new MySqlCommand(query, conn);
                conn.Open();
                cmd.CommandText = query;
                cmd.Connection = conn;
                cmd.Parameters.AddWithValue("@Beginning", beginningCityId);
                cmd.Parameters.AddWithValue("@Destination",
destinationCityId);
                cmd.Parameters.AddWithValue("@TimeDepart", departDateTime);
                cmd.Parameters.AddWithValue("@TimeArrive", arrivalDateTime);
                cmd.Parameters.AddWithValue("@Price", priceTb.Text);
                cmd.Parameters.AddWithValue("@BusId", busId);
                cmd.Parameters.AddWithValue("@Id", idTb.Text);
                cmd.ExecuteNonQuery();
                conn.Close();
                MessageBox.Show("Transport edited.");
                displayTransports();
                loadBuses();
            }
            catch (Exception exception)
            {
                MessageBox.Show("Query error:" + exception.Message);
                conn.Close();
            }
        }
        else
        {
            MessageBox.Show("Please select a transport");
        }
    }
}

private void deleteBt_Click(object sender, EventArgs e)
{
    if (idTb.TextLength > 0)
    {
        deleteTickets();
        string query = "DELETE FROM Transport WHERE ID=" + idTb.Text;
        MySqlCommand command = new MySqlCommand(query, conn);
        conn.Open();
        command.CommandText = query;
        command.Connection = conn;
        command.ExecuteNonQuery();
    }
}

```

```

        MessageBox.Show("Transport deleted.");
        conn.Close();
        displayTransports();
        loadBuses();
    }
    else
    {
        MessageBox.Show("Please select a transport");
    }
}

private void deleteTickets()
{
    string query = "DELETE FROM Ticket WHERE transport_id=" + idTb.Text;
    MySqlCommand command = new MySqlCommand(query, conn);
    conn.Open();
    command.CommandText = query;
    command.Connection = conn;
    command.ExecuteNonQuery();
    conn.Close();
}

private HashSet<string> getAllBuses()
{
    HashSet<string> b = new HashSet<string>();

    string query = "SELECT b.id FROM Bus b";
    try
    {
        conn.Open();
        MySqlCommand cmd = new MySqlCommand(query, conn);
        MySqlDataReader reader = cmd.ExecuteReader();
        while (reader.Read())
        {
            b.Add(reader["id"].ToString());
        }
        conn.Close();
    }
    catch (Exception e)
    {
        MessageBox.Show("Query error:" + e.Message);
        conn.Close();
    }

    return b;
}

private void loadCities(ComboBox cb)
{
    string query = "SELECT id,name FROM City";
    try
    {
        conn.Open();
        MySqlCommand cmd = new MySqlCommand(query, conn);
        MySqlDataReader reader = cmd.ExecuteReader();
        while (reader.Read())
        {
            cb.Items.Add(reader["name"].ToString());
            cb.ValueMember = reader["id"].ToString();
            cb.DisplayMember = reader["name"].ToString();
        }
        conn.Close();
    }
}

```

```

    }
    catch (Exception e)
    {
        MessageBox.Show("Query error:" + e.Message);
        conn.Close();
    }
}

private void loadBuses()
{
    busCb.Items.Clear();
    getUsedBuses();
    allBuses = getAllBuses();
    allBuses.ExceptWith(buses);
    foreach (string s in allBuses)
    {
        busCb.Items.Add(s);
    }
    busCb.SelectedIndex = 0;
}

private void displayTransports()
{
    string query = "SELECT t.id, c.name as \"Beginning\", cc.name as \"Destination\", t.timeDepart, t.timeArrive, price, bus_id FROM Transport t join City c on t.beginning = c.id join City cc on t.destination = cc.id";
    try
    {
        conn.Open();
        MySqlDataAdapter adapter = new MySqlDataAdapter(query, conn);
        DataTable dt = new DataTable();
        adapter.Fill(dt);
        usersDV.DataSource = dt;
        conn.Close();
    }
    catch (Exception e)
    {
        MessageBox.Show("Query error:" + e.Message);
        conn.Close();
    }
}

private bool checkInfo()
{
    if (priceTb.Text.Length < 1)
    {
        MessageBox.Show("Invalid price");
        return false;
    }
    if (destinationCb.SelectedIndex == beginningCb.SelectedIndex)
    {
        MessageBox.Show("Can not travel to the same city");
        return false;
    }
    if (departDateTime.CompareTo(arrivalDateTime) >= 0)
    {
        MessageBox.Show("Departure can not be after arrival.");
        return false;
    }
    return true;
}

```

```

private void priceTb_KeyPress(object sender, KeyPressEventArgs e)
{
    if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar) &&
        (e.KeyChar != '.'))
    {
        e.Handled = true;
    }

    // only allow one decimal point
    if ((e.KeyChar == '.') && ((sender as TextBox).Text.IndexOf('.') > -1))
    {
        e.Handled = true;
    }
}

private void departureDate_ValueChanged(object sender, EventArgs e)
{
    departDateTime = departureDate.Value.Date + departureTime.Value.TimeOfDay;
    loadBuses();
}

private void departureTime_ValueChanged(object sender, EventArgs e)
{
    departDateTime = departureDate.Value.Date + departureTime.Value.TimeOfDay;
    loadBuses();
}

private void arrivalDate_ValueChanged(object sender, EventArgs e)
{
    arrivalDateTime = arrivalDate.Value.Date + arrivalTime.Value.TimeOfDay;
    loadBuses();
}

private void arrivalTime_ValueChanged(object sender, EventArgs e)
{
    arrivalDateTime = arrivalDate.Value.Date + arrivalTime.Value.TimeOfDay;
    loadBuses();
}

private void getUsedBuses()
{
    buses = new HashSet<string>();
    string query = "SELECT bus_id FROM Transport where (timeDepart BETWEEN
@departDate AND @arriveDate) or (timeArrive BETWEEN @departDate AND @arriveDate)";
    try
    {
        conn.Open();
        MySqlCommand cmd = new MySqlCommand(query, conn);
        cmd.Parameters.AddWithValue("@departDate", departDateTime);
        cmd.Parameters.AddWithValue("@arriveDate", arrivalDateTime);
        MySqlDataReader reader = cmd.ExecuteReader();
        while (reader.Read())
        {
            buses.Add(reader["bus_id"].ToString());
        }
        conn.Close();
    }
    catch (Exception e)
    {
        MessageBox.Show("Query error:" + e.Message);
        conn.Close();
    }
}

```

```

    }

    private void usersDV_RowHeaderMouseClick(object sender,
DataGridViewCellEventArgs e)
    {
        idTb.Text = usersDV.Rows[e.RowIndex].Cells[0].Value.ToString();
        beginningCb.SelectedItem =
usersDV.Rows[e.RowIndex].Cells[1].Value.ToString();
        destinationCb.SelectedItem =
usersDV.Rows[e.RowIndex].Cells[2].Value.ToString();
        priceTb.Text = usersDV.Rows[e.RowIndex].Cells[5].Value.ToString();
        busCb.SelectedItem = usersDV.Rows[e.RowIndex].Cells[6].Value.ToString();

        string role = usersDV.Rows[e.RowIndex].Cells[5].Value.ToString();
    }
}

```

### ***UserDashboardForm***

**Home**

**Your tickets:** Show active Show All

Благоевград->Кюстендил(5.1.2022 12:12:51 PM), seat:6, price:22лв.  
Благоевград->Кюстендил(5.1.2022 12:12:51 PM), seat:40, price:22лв.  
Благоевград->Кюстендил(5.1.2022 12:12:51 PM), seat:47, price:22лв.  
София->Видин(5.18.2022 11:43:59 AM), seat:22, price:21лв.  
София->Велико Търново(5.18.2022 11:43:59 AM), seat:7, price:21лв.

```

public partial class UserDashboardForm : Form
{
    private MySqlConnection conn;
    private string username;
    public UserDashboardForm(string username)

```

```

{
    string connection =
"datasource=127.0.0.1;port=3306;username=root;password=;database=bus_transport";
    conn = new MySqlConnection(connection);
    this.username = username;
    InitializeComponent();
}

private void UserDashboardForm_Load(object sender, EventArgs e)
{
    displayActiveTickets();
}

private void displayActiveTickets()
{
    string query = "SELECT t.id as \"Id\", c.name as \"Beginning\", cc.name as
\"Destination\", t.timeDepart as \"Departure\", ti.seatNum as \"Seat\", t.price as
\"Price\"" +
        " FROM Ticket ti join Transport t on ti.transport_id = t.id
join City c on t.beginning = c.id join City cc on t.destination = cc.id" +
        " where ti.person_id = " + getIdByUsername(username) + " and
t.timeDepart >= CURRENT_TIMESTAMP";
    try
    {
        conn.Open();
        MySqlCommand cmd = new MySqlCommand(query, conn);
        MySqlDataReader reader = cmd.ExecuteReader();
        while (reader.Read())
        {
            listBox1.Items.Add(reader["Beginning"] + "->" +
reader["Destination"] + "(" + reader["Departure"] + "), seat:" + reader["Seat"] + ",
price:" + reader["Price"] + "лв.");
        }
        conn.Close();
    }
    catch (Exception e)
    {
        MessageBox.Show("Query error:" + e.Message);
        conn.Close();
    }
}

private void displayAllTickets()
{
    string query = "SELECT t.id as \"Id\", c.name as \"Beginning\", cc.name as
\"Destination\", t.timeDepart as \"Departure\", ti.seatNum as \"Seat\", t.price as
\"Price\"" +
        " FROM Ticket ti join Transport t on ti.transport_id = t.id join City
c on t.beginning = c.id join City cc on t.destination = cc.id" +
        " where ti.person_id = "+ getIdByUsername(username);
    try
    {
        conn.Open();
        MySqlCommand cmd = new MySqlCommand(query, conn);
        MySqlDataReader reader = cmd.ExecuteReader();
        while (reader.Read())
        {
            listBox1.Items.Add(reader["Beginning"] + "->" +
reader["Destination"] + "(" + reader["Departure"] + "), seat:" + reader["Seat"] + ",
price:" + reader["Price"] + "лв.");
        }
        conn.Close();
    }
}

```

```

    }
    catch (Exception e)
    {
        MessageBox.Show("Query error:" + e.Message);
        conn.Close();
    }
}

private void showActiveBt_Click(object sender, EventArgs e)
{
    listBox1.Items.Clear();
    displayActiveTickets();
}

private void showAllBt_Click(object sender, EventArgs e)
{
    listBox1.Items.Clear();
    displayAllTickets();
}

private string getIdByUsername(string username)
{
    string query = "SELECT id FROM User where username='" + username + "'";
    string id = "";
    try
    {
        conn.Open();
        MySqlCommand cmd = new MySqlCommand(query, conn);
        MySqlDataReader reader = cmd.ExecuteReader();
        while (reader.Read())
        {
            id = reader["id"].ToString();
        }
        conn.Close();
    }
    catch (Exception e)
    {
        MessageBox.Show("Query error:" + e.Message);
        conn.Close();
    }
    return id;
}

private void label2_Click(object sender, EventArgs e)
{
}
}

```

## UserForm

### User registration:

Name:

Username:

Email:

Password

Employee(2) ▼

Selected user:

	id	username	password
	1	user1	password1
	2	nana123	12345678
	7	vujnata	123456789
	9	puntacana	12345678
	13	a	a
▶	14	employee1	123456789
	17	user3	12345
	24	e	e
	25	u	u
*			

```

public partial class UserForm : Form
{
    private MySqlConnection conn;
    public UserForm()
    {
        string connection =
"datasource=127.0.0.1;port=3306;username=root;password=;database=bus_transport";
        conn = new MySqlConnection(connection);

        InitializeComponent();
    }

    private void UserForm_Load(object sender, EventArgs e)
    {
        roleCb.Items.Add("User(3)");
        roleCb.Items.Add("Employee(2)");
        roleCb.Items.Add("Administrator(1)");
        roleCb.SelectedIndex = 0;
        DisplayUser();
    }

    private void addBt_Click(object sender, EventArgs e)
    {
        addUser();
    }

    private void editBt_Click(object sender, EventArgs e)
    {
        editUser();
    }
}

```



```

private void DisplayUser()
{
    string query = "SELECT * FROM User";
    try
    {
        conn.Open();
        MySqlDataAdapter adapter = new MySqlDataAdapter(query, conn);
        DataTable dt = new DataTable();
        adapter.Fill(dt);
        usersDV.DataSource = dt;
        conn.Close();
    }
    catch (Exception e)
    {
        MessageBox.Show("Query error:" + e.Message);
    }
}

private bool checkIfUserExists(string username)
{
    try
    {
        conn.Open();
        using (var sqlCommand = new MySqlCommand("SELECT * FROM User WHERE
username = '" + username + "'", conn))
        {
            MySqlDataReader reader = sqlCommand.ExecuteReader();
            if (reader.HasRows)
            {
                reader.Close();
                reader.Dispose();
                conn.Close();
                return false;
            }
            else
            {
                reader.Close();
                reader.Dispose();
                conn.Close();
                return true;
            }
        }
    }
    catch (Exception e)
    {
        MessageBox.Show("Query error:" + e.Message);
    }
    return false;
}

private void addUser()
{
    int role=3;
    if (roleCb.SelectedIndex == 1) role = 2;
    if (roleCb.SelectedIndex == 2) role = 1;
    string query = $"INSERT INTO User (id, name, email, username, password,
role_id) VALUES" +
        "(NULL, @Name, @Email, @Username, @Password, @Role)";

```

```

        if (nameTb.Text != null && emailTb.Text != null && usernameTb.Text != null
&& passTb != null && nameTb.Text.Length > 0
        && emailTb.Text.Length > 0 && usernameTb.Text.Length > 0 &&
passTb.Text.Length > 0)
        {
            if (checkIfUserExists(usernameTb.Text) != false)
            {
                try
                {
                    conn.Open();
                    MySqlCommand cmd = new MySqlCommand(query, conn);
                    cmd.Parameters.AddWithValue("@Name", nameTb.Text);
                    cmd.Parameters.AddWithValue("@Email", emailTb.Text);
                    cmd.Parameters.AddWithValue("@Username", usernameTb.Text);
                    cmd.Parameters.AddWithValue("@Password", passTb.Text);
                    cmd.Parameters.AddWithValue("@Role", role);
                    cmd.ExecuteNonQuery();
                    conn.Close();
                    MessageBox.Show("User successfully added.");
                    DisplayUser();
                }
                catch (Exception e)
                {
                    MessageBox.Show("Query error:" + e.Message);
                }
            }
            else
            {
                MessageBox.Show("Username already exists.");
            }
        }
        else
        {
            MessageBox.Show("Please fill all tabs.");
        }
    }

    private void editUser()
    {
        if (idTb.Text.Length > 0) {
            int role = 3;
            if (roleCb.SelectedIndex == 1) role = 2;
            if (roleCb.SelectedIndex == 2) role = 1;
            string query = $"Update User SET name = @Name, username = @Username,
email = @Email, password = @Password, role_id = @Role WHERE ID= @Id";
            Console.Write(query);
            try
            {
                MySqlCommand command = new MySqlCommand(query, conn);
                conn.Open();
                command.CommandText = query;
                command.Connection = conn;
                command.Parameters.AddWithValue("@Name", nameTb.Text);
                command.Parameters.AddWithValue("@Email", emailTb.Text);
                command.Parameters.AddWithValue("@Username", usernameTb.Text);
                command.Parameters.AddWithValue("@Password", passTb.Text);
                command.Parameters.AddWithValue("@Role", role);
                command.Parameters.AddWithValue("@Id", idTb.Text);
                command.ExecuteNonQuery(); //zaduljitelni, inache nqma da se

                MessageBox.Show("User edited");
                conn.Close();
            }
        }
    }

```

```

        DisplayUser();
    }
    catch (Exception e)
    {
        MessageBox.Show("Query error:" + e.Message);
    }
}
else
{
    MessageBox.Show("Please select an user");
}
}

private void deleteUser()
{
    if (idTb.TextLength > 0)
    {
        deleteTickets();
        string query = "DELETE FROM User WHERE ID=" + idTb.Text;
        MySqlCommand command = new MySqlCommand(query, conn);
        conn.Open();
        command.CommandText = query;
        command.Connection = conn;
        command.ExecuteNonQuery(); //zaduljitelni, inache nqma da se izpulni
zaqvkata
        MessageBox.Show("User deleted.");
        conn.Close();
        DisplayUser();
    }
    else
    {
        MessageBox.Show("Please select an user");
    }
}

private void deleteTickets()
{
    string query = "DELETE FROM Ticket WHERE person_id=" + idTb.Text;
    MySqlCommand command = new MySqlCommand(query, conn);
    conn.Open();
    command.CommandText = query;
    command.Connection = conn;
    command.ExecuteNonQuery();
    conn.Close();
}

private void usersDV_RowHeaderMouseClick(object sender,
DataGridViewCellEventArgs e)
{
    idTb.Text = usersDV.Rows[e.RowIndex].Cells[0].Value.ToString();
    usernameTb.Text = usersDV.Rows[e.RowIndex].Cells[1].Value.ToString();
    passTb.Text = usersDV.Rows[e.RowIndex].Cells[2].Value.ToString();
    nameTb.Text = usersDV.Rows[e.RowIndex].Cells[3].Value.ToString();
    emailTb.Text = usersDV.Rows[e.RowIndex].Cells[4].Value.ToString();

    string role = usersDV.Rows[e.RowIndex].Cells[5].Value.ToString();

    if (role.Equals("1")) roleCb.SelectedIndex = 2;
    if (role.Equals("2")) roleCb.SelectedIndex = 1;
    if (role.Equals("3")) roleCb.SelectedIndex = 0;
}

```

```

private void deleteBt_Click(object sender, EventArgs e)
{
    deleteUser();
}
}

```

### ***UserTicketForm***

Search by: Beginning

	id	Beginning	Destination	timeDepart
	5	София	Видин	5/18/2022 11:43:59 AM
▶	6	София	Велико Търново	5/18/2022 11:43:59 AM
*				

Pick a seat:

☐ 1
 ☐ 2
 ☐ 3
 ☐ 4
 ☐ 5
 ☒ 6
 ☐ 7
 ☐ 8

☐ 9
 ☐ 10
 ☐ 11
 ☐ 12
 ☐ 13
 ☐ 14
 ☐ 15
 ☐ 16

☐ 17
 ☐ 18
 ☐ 19
 ☐ 20
 ☐ 21
 ☐ 22
 ☐ 23
 ☐ 24

6 София - Велико Търново | 5/18/2022 11:43:59 AM - 5/18/2022 5:43:59 PM

```

public partial class UserTicketForm : Form
{
    private MySqlConnection conn;
    private string username;
    public UserTicketForm(string username)
    {
        this.username = username;
        string connection =
"datasource=127.0.0.1;port=3306;username=root;password=;database=bus_transport";
        conn = new MySqlConnection(connection);

        InitializeComponent();
    }

    private void UserTicketForm_Load(object sender, EventArgs e)
    {
        searchByCb.Items.Add("Beginning");
        searchByCb.Items.Add("Destination");
        searchByCb.SelectedIndex = 0;
        loadTransports();
    }
}

```

```

private void addBt_Click(object sender, EventArgs e)
{
    if (checkInfo() == true)
    {
        string seat = this.Controls.OfType<RadioButton>().FirstOrDefault(r =>
r.Checked).Text;
        string userId = getUserId();
        string query = $"INSERT INTO Ticket (person_id, transport_id, seatNum,
status_id) VALUES" +
            "(@Person, @Transport, @Seat, @Status)";

        HashSet<string> transports = getUsedTransports();
        if (transports.Contains(transportIdTb.Text))
        {
            MessageBox.Show("You have already booked a ticket for this
transport.");
        }
        else
        {
            try
            {
                conn.Open();
                MySqlCommand cmd = new MySqlCommand(query, conn);
                cmd.Parameters.AddWithValue("@Person", userId);
                cmd.Parameters.AddWithValue("@Transport", transportIdTb.Text);
                cmd.Parameters.AddWithValue("@Seat", seat);
                cmd.Parameters.AddWithValue("@Status", "1");
                cmd.ExecuteNonQuery();
                conn.Close();
                //printTicket();
                MessageBox.Show("Ticket successfully reserved.");
                RadioButton rb = (RadioButton)Controls[seat];
                rb.Enabled = false;
                rb.Checked = false;
                loadTransports();

            }
            catch (Exception exception)
            {
                MessageBox.Show("Query error:" + exception.Message);
                conn.Close();
            }
        }
    }
}

private HashSet<string> getUsedTransports()
{
    HashSet<string> transports = new HashSet<string>();

    string query = "SELECT transport_id FROM Ticket where person_id = " +
getUserId();
    try
    {
        conn.Open();
        MySqlCommand cmd = new MySqlCommand(query, conn);
        MySqlDataReader reader = cmd.ExecuteReader();
        while (reader.Read())
        {
            transports.Add(reader["transport_id"].ToString());
        }
    }
}

```

```

        conn.Close();
    }
    catch (Exception e)
    {
        MessageBox.Show("Query error:" + e.Message);
        conn.Close();
    }

    return transports;
}

private void loadTransports()
{
    string query = "SELECT t.id, c.name as \"Beginning\", cc.name as \"Destination\", t.timeDepart, t.timeArrive, price, bus_id FROM Transport t" +
        " join City c on t.beginning = c.id join City cc on t.destination = cc.id where t.timeDepart >= CURRENT_TIMESTAMP";

    try
    {
        conn.Open();
        MySqlDataAdapter adapter = new MySqlDataAdapter(query, conn);
        DataTable dt = new DataTable();
        adapter.Fill(dt);
        transportDV.DataSource = dt;
        conn.Close();
    }
    catch (Exception e)
    {
        MessageBox.Show("Query error:" + e.Message);
        conn.Close();
    }
}

public int searchBus(string busId)
{
    int i = 0;
    string query = "SELECT busType_id FROM Bus where id = '" + busId + "'";
    try
    {
        conn.Open();
        MySqlCommand cmd = new MySqlCommand(query, conn);
        MySqlDataReader reader = cmd.ExecuteReader();
        while (reader.Read())
        {
            i = (int)reader["busType_id"];
        }
        conn.Close();
    }
    catch (Exception e)
    {
        MessageBox.Show("Query error:" + e.Message);
        conn.Close();
    }
    return i;
}

public void generateSeats(int seats, int yEnd)
{

```

```

List<int> takenSeats = getTakenSeats(transportIdTb.Text);
clearSeats();
int plus = 0;
int x = 400;
int y = 50;
int seat = 1;
for (int j = 40; j <= yEnd; j += 40)
{
    if (j == 120) j += 40;
    for (int i = 1; i <= seats; i++)
    {
        RadioButton rdo = new RadioButton();
        rdo.Name = seat.ToString();
        rdo.Text = seat.ToString();
        rdo.ForeColor = Color.Black;
        rdo.Location = new Point(x + j, y + plus);
        rdo.AutoSize = true;
        if (takenSeats.Contains(seat)) rdo.Enabled = false;
        this.Controls.Add(rdo);
        plus = plus + 20;
        seat++;
    }
    plus = 0;
}

public List<int> getTakenSeats(string transportId)
{
    List<int> seats = new List<int>();

    string query = "SELECT seatNum FROM Ticket where transport_id = '" +
transportId + "'";
    try
    {
        conn.Open();
        MySqlCommand cmd = new MySqlCommand(query, conn);
        MySqlDataReader reader = cmd.ExecuteReader();
        while (reader.Read())
        {
            seats.Add((int)reader["seatNum"]);
        }
        conn.Close();
    }
    catch (Exception e)
    {
        MessageBox.Show("Query error:" + e.Message);
        conn.Close();
    }

    return seats;
}

public void clearSeats()
{
    foreach (Control item in this.Controls.OfType<RadioButton>().ToList())
    {
        this.Controls.Remove(item);
    }
}

public bool checkInfo()
{

```

```

        if (trasnportTb.Text.Length < 1)
        {
            MessageBox.Show("Choose a transport");
            return false;
        }
        if (this.Controls.OfType<RadioButton>().FirstOrDefault(r => r.Checked) ==
null)
        {
            MessageBox.Show("Choose a seat");
            return false;
        }
        return true;
    }

    private string getUserId()
    {
        string i = "0";
        string query = "SELECT id FROM User where username = '" + username + "'";
        try
        {
            conn.Open();
            MySqlCommand cmd = new MySqlCommand(query, conn);
            MySqlDataReader reader = cmd.ExecuteReader();
            while (reader.Read())
            {
                i = reader["id"].ToString();
            }
            conn.Close();
        }
        catch (Exception e)
        {
            MessageBox.Show("Query error:" + e.Message);
            conn.Close();
        }
        return i;
    }

    private void transportDV_RowHeaderMouseClick_1(object sender,
DataGridViewCellEventArgs e)
    {
        string transport;
        transport = transportDV.Rows[e.RowIndex].Cells[1].Value.ToString()
            + " - " + transportDV.Rows[e.RowIndex].Cells[2].Value.ToString() + " |
" + transportDV.Rows[e.RowIndex].Cells[3].Value.ToString()
            + " - " + transportDV.Rows[e.RowIndex].Cells[4].Value.ToString();
        trasnportTb.Text = transport;
        int type =
searchBus(transportDV.Rows[e.RowIndex].Cells[6].Value.ToString());
        transportIdTb.Text =
transportDV.Rows[e.RowIndex].Cells[0].Value.ToString();

        if (type == 1) generateSeats(8, 120);
        if (type == 2) generateSeats(10, 200);
        if (type == 3) generateSeats(14, 200);
    }

    private void searchTransport()
    {
        string query="";
        if (searchByCb.SelectedIndex == 0)
        {

```



```

        query = "SELECT t.id, c.name as \"Beginning\", cc.name as
\"Destination\", t.timeDepart, t.timeArrive, price, bus_id FROM Transport t" +
        " join City c on t.beginning = c.id join City cc on
t.destination = cc.id where c.name = '"+searchTb.Text+ "' and t.timeDepart >=
CURRENT_TIMESTAMP";
    }
    else
    {
        query = "SELECT t.id, c.name as \"Beginning\", cc.name as
\"Destination\", t.timeDepart, t.timeArrive, price, bus_id FROM Transport t"+
        " join City c on t.beginning = c.id join City cc on
t.destination = cc.id where cc.name = '" + searchTb.Text + "' and t.timeDepart >=
CURRENT_TIMESTAMP";
    }

    try
    {
        conn.Open();
        MySqlDataAdapter adapter = new MySqlDataAdapter(query, conn);
        DataTable dt = new DataTable();
        adapter.Fill(dt);
        transportDV.DataSource = dt;
        conn.Close();
    }
    catch (Exception e)
    {
        MessageBox.Show("Query error:" + e.Message);
        conn.Close();
    }
}

private void button1_Click(object sender, EventArgs e)
{
    searchTransport();
}

private void showAllBt_Click(object sender, EventArgs e)
{
    loadTransports();
}
}

```

### ***Unit Tests***

```

public class UnitTest1
{
    //Bus Testing
    [TestMethod]
    public void Test_IfBusDoesntExist()
    {
        BusTransportation.BusForm form = new BusTransportation.BusForm();

        string plate = "CC9696CA";
        bool actual = form.checkIfBusDoesntExists(plate);
        Assert.AreEqual(false, actual);

        plate = "asdasdasdas";
        actual = form.checkIfBusDoesntExists(plate);
    }
}

```

```

        Assert.AreEqual(true, actual);
    }

    //Edit Ticket Testing
    [TestMethod]
    public void Test_getIdByUsername()
    {
        BusTransportation.EditTicket form = new BusTransportation.EditTicket();

        string user = "a";
        string actual = form.getIdByUsername(user);
        Assert.AreEqual("13", actual);
    }

    [TestMethod]
    public void Test_getUser()
    {
        BusTransportation.EditTicket form = new BusTransportation.EditTicket();

        string userId = "13";
        string actual = form.getUser(userId);
        Assert.AreEqual("13: a", actual);
    }

    [TestMethod]
    public void Test_getTransport()
    {
        BusTransportation.EditTicket form = new BusTransportation.EditTicket();

        string transportId = "1";
        string actual = form.getTransport(transportId);
        Assert.AreEqual("1: Благоевград->Жюстендил(5/1/2022 12:12:51 PM)",
actual);
    }

    [TestMethod]
    public void Test_searchBus()
    {
        BusTransportation.EditTicket form = new BusTransportation.EditTicket();

        string transportId = "1";
        int actual = form.searchBus(transportId);
        Assert.AreEqual(1, actual); //maluk
    }

    [TestMethod]
    public void Test_getTransportPrice()
    {
        BusTransportation.EditTicket form = new BusTransportation.EditTicket();

        string transportId = "1";
        string actual = form.getTransportPrice(transportId);
        Assert.AreEqual("22", actual);
    }

    //Register testing

    [TestMethod]
    public void Test_isValidEmail()
    {
        BusTransportation.Form1 form = new BusTransportation.Form1();

```

```

        string email = "testemail@abv.bg";
        bool actual = form.isValidEmail(email);
        Assert.AreEqual(true, actual);

        email = "incorrectemail";
        actual = form.isValidEmail(email);
        Assert.AreEqual(false, actual);
    }

    [TestMethod]
    public void Test_checkIfUserDoesntExist()
    {
        BusTransportation.Form1 form = new BusTransportation.Form1();

        string user = "a";
        bool actual = form.checkIfUserDoesntExist(user);
        Assert.AreEqual(false, actual);    //it exists

        user = "doesntexist";
        actual = form.checkIfUserDoesntExist(user);
        Assert.AreEqual(true, actual);
    }
}

```