Céges autók grafikus feladat

SQL állomány

Az SQL állomány áttekintése

Adatbázis importálása

XAMPP Control Panel

Apache webszerver és MySQL adatbázis szerver elindítása

phpmyadmin felület megnyitása: http://localhost/phpmyadmin/

Új / adatbázis neve: parkolohaz (utf8_general_ci) / Létrehozás

Importálás / Tallózás / Importálás

Az adatbázis áttekintése: táblák szerkezete, tartalma és kapcsolata

Adatbázis kapcsolat beállítása

Visual Studio / Create a new project / WPF Application / Project name: CegesParkolohaz / Create

A WPF Application template megfelelő a projekt létrehozásához, biztosítja az alapvető felépítést:

- MainWindow.xaml: az alapértelmezett fő ablak felhasználói felületének tervezésére szolgál.
- MainWindow.xaml.cs: az alkalmazás betöltését és a főablak indítását tartalmazza.

A MySQL adatbázis kapcsolat beállításához a MySQL.Data NuGet csomagra lesz szükségünk, amely lehetővé teszi a C# alkalmazások számára, hogy kapcsolódjanak MySQL adatbázisokhoz.

MySQL.Data Nuget csomag telepítése:

- NuGet Package Manager: Tools menü / NuGet Package Manager / Package Manager Console (megjelenik egy konzol ablak a Visual Studio alján, ahol parancsok segítségével lehet telepíteni)

- NuGet Package Manager: Tools menü / NuGet Package Manager / Manage NuGet Packages for Solution (új csomagok hozzáadása, csomagok frissítése, eltávolítása)

A konzol ablakban az Install-Package MySql.Data paranccsal telepíthetjük.

Célszerű az adatbázis kapcsolatot külön fájlban létrehozni, mert átláthatóbb és könnyebben karbantartható a kód.

Solution Explorer / projekt nevére kattintás jobb egérgombbal / Add / Class / DatabaseConnection.cs / Add

```
class DatabaseConnection
{
        private MySqlConnection conn;
(A MySQL kapcsolat példánya)
        public DatabaseConnection(string connStr)
(A konstruktor, amely a kapcsolódási stringet kapja paraméterként)
        {
                conn = new MySqlConnection(connStr);
(Létrehozza a MySQL kapcsolatot a megadott string alapján)
        }
        public void OpenConnection()
(A kapcsolat megnyitására szolgáló metódus)
        {
                try
                {
                        conn.Open();
(Megnyitja a kapcsolatot az adatbázissal)
                }
                catch(Exception ex)
(Hiba esetén elkapja a kivételt)
                {
                        MessageBox.Show($"Hiba a kapcsolódás során:
{ex.Message}");
(Hibaüzenetet jelenít meg egy MessageBox-ban)
                }
        }
        public void CloseConnection()
(A kapcsolat bezárására szolgáló metódus)
        {
                if (conn.State == System.Data.ConnectionState.Open)
(Ellenőrzi, hogy a kapcsolat nyitott állapotban van-e)
```

```
                {
                        conn.Close();
(Bezárja a kapcsolatot)
                }
        }
}
```

Használjuk a DataBaseConnection osztályt a MainWindow.xaml.cs fájlban, példányosítsuk az osztályt és nyissuk meg a kapcsolatot.

```
class MainWindow : Window
{
        private DatabaseConnection dbConn;
(DatabaseConnection objektum, amin keresztül kezeljük az adatbázis kapcsolatot)
        public MainWindow()
        {
                InitializeComponent();
                string connStr = "Server=localhost;Database=parkolohaz;User
ID=root;";
(Az adatbázis kapcsolódási sztringje, tartalmazza az adatbázis eléréséhez
szükséges információkat: szerver, adatbázis neve, felhasználónév, jelszó)
                dbConn = new DatabaseConnection(connStr);
(A DatabaseConnection példányosítása a kapcsolódási sztringgel)
                dbConn.OpenConnection();
(Az adatbázis kapcsolat megnyitása, ha a kapcsolat sikeres, innen lehet
kommunikálni az adatbázissal)
        }
}
```

Ha a kapcsolódási sztring jelszót is tartalmaz, akkor a végére a Password=érték kerül.

Tesztelési lehetőségek

Felugró ablakban üzenet, ha a kapcsolódás sikeres:

```
try
{
        dbConn.OpenConnection();
        MessageBox.Show("Kapcsolódás sikeres!");
}
catch (Exception ex)
{
```

```
        MessageBox.Show($"Hiba történt a kapcsolódás során: {ex.Message}");
}
```

Teszt lekérdezés (az adatbázis egyik táblájának a neve):

DatabaseConnection.cs:

```
public List<string> ExecuteQuery(string query)
(A lekérdezést végrehajtó metódus List típusú, paraméterként a lekérdezést kapja meg szövegként)
{
        List<string> results = new List<string>();
(Egy lista, amelyben a lekérdezés eredményeit tároljuk)

        try
        {
                MySqlCommand cmd = new MySqlCommand(query, conn);
(Létrehozunk egy SQL-parancs objektumot a megadott lekérdezéshez és kapcsolatot rendelünk hozzá)
                using (MySqlDataReader reader = cmd.ExecuteReader())
(A "using" kulcsszó gondoskodik arról, hogy az olvasó (reader) megfelelően lezáródjon a használat után)
                {
                        while (reader.Read())
(Amíg van olvasható sor az eredményben, bejárjuk azokat)
                        {
                                results.Add(reader[0].ToString());
(Az aktuális sor első oszlopának értékét szöveggé alakítjuk, majd hozzáadjuk az eredmények listájához)

                        }
                }
        }
        catch (MySqlException ex)
        {
                MessageBox.Show($"SQL-lekérdezési hiba: {ex.Message}");
(Ha SQL-hiba történik, megjelenítünk egy üzenetablakot a hibaüzenettel)
        }
        return results;
(Visszatérünk az eredmények listájával, ez lehet üres is, ha nincs találat)
}
```

MainWindow.xaml.cs:

```
string query = "SHOW TABLES;";
(Egy SQL lekérdezést definiálunk, amely az adatbázis összes tábláját visszaadja)
var result = dbConn.ExecuteQuery(query);
(A lekérdezést végrehajtjuk az ExecuteQuery metódus segítségével, az eredményt
egy `result` nevű változóban tároljuk, amely egy listát tartalmaz)
if (result != null && result.Count > 0)
(Ellenőrizzük, hogy a `result` lista nem null és tartalmaz-e elemeket)
{
        MessageBox.Show($"Első tábla neve: {result[0]}");
(Ha van találat, megjelenítjük az első tábla nevét egy üzenetablakban)
}
else
{
        MessageBox.Show("Nincsenek táblák az adatbázisban.");
(Ha nincs találat - az adatbázisban nincsenek táblák -, egy figyelmeztetést jelenítünk
meg)
}
```

A felugró ablakban megjelenik a tábla neve, ha sikeres a kapcsolat. Ezután töröljük a kódból a tesztelést.

A bérlések listázának megvalósítása

Szükség van egy listára, amelyben az alkalmazottakat tudjuk kiválasztani.

Az adatbázisból le kell kérdezni az alkalmazottak nevét, a DatabaseConnection osztályban hozzunk létre ehhez egy új metódust.

```
public List<string> GetEmployees()
(Az alkalmazottak nevét egy listába töltjük be)
{
        string query = "SELECT nev FROM alkalmazottak;";
(Létrehozzuk a lekérdezés szövegét, a neveket szeretnénk lekérdezni az
alkalmazottak táblából)
        return ExecuteQuery(query);
(A lekérdezést végrehajtjuk az ExecuteQuery metódus segítségével, az eredményt
visszaadjuk)
}
```

A MainWindow.xaml fájlban egy Grid-ben helyezzünk el egy ListBox-ot.

```
<Grid>
```

```
        <ListBox Name="EmployeeListBox"
(Ezen a néven hivatkozhatunk a ListBox-ra a kódban)
                HorizontalAlignment="Left"
                VerticalAlignment="Top"
                Width="200"
                Height="380"
                Margin="10"/>
</Grid>
```

A MainWindow.cs fájlban a ListBox-ot töltsük fel az alkalmazottak neveivel.

```
var employees = dbConn.GetEmployees();
(Meghívjuk a GetEmployees metódust, ami visszaadja az alkalmazottak neveit)
        EmployeeListBox.ItemsSource = employees;
(A ListBox-hoz adjuk a neveket az ItemsSource metódus segítségével)
```

A kiválasztott alkalmazott adatainak megjelenítése táblázatban

Adjunk egy DataGrid vezérlőt az xaml fájlhoz és kicsit alakítsuk át az xaml kódot.

```
<Grid>
        <Grid.RowDefinitions>
                <RowDefinition Height="50"/>
                <RowDefinition Height="220"/>
                <RowDefinition Height=”*”/>
        </Grid.RowDefinitions>
(A Grid három sorból áll.)
        <Grid.ColumnDefinitions>
                <ColumnDefinition Width="200"/>
                <ColumnDefinition Width="580"/>
        </Grid.ColumnDefinitions>
(A Grid két oszlopból áll, az oszlopok szélessége pontosan meg van adva.)
        <ListBox Name="EmployeeListBox"
                Grid.RowSpan="3"
                Grid.Row="0"
                Grid.Column="0"
(A ListBox a Grid 0. sorában és 0. oszlopában helyezkedik el, a 0. oszlopban a
három cella össze van vonva.)
                Width="180"
                Height="380"
                Margin="10"
                SelectionChanged=”EmployeeListBox_SelectionChanged”/>
```

(Ha változik a kijelölés a ListBox-ban, az EmployeeListBox_SelectionChanged függvény hívjuk meg.)
<TextBlock Name="EmployeeName"
(A táblázat felett a kiválasztott alkalmazott neve jelenik meg.)
Grid.Row="0"
Grid.Column="1"
(A TextBlock a Grid 0. sorában és az 1. oszlopában helyezkedik el.)
Visibility="Collapsed"
Text="XY bérlései:"
FontSize="24"
VerticalAlignment="Bottom"
Margin="10,0,0,0"/>
<DataGrid Name="EmployeeTable"
(DataGrid a bérlések megjelenítéséhez.)
Grid.Row="1"
Grid.Column="1"
(A DataGrid a Grid 1. sorában és az 1. oszlopában helyezkedik el.)
Width="560"
Height="200"
Margin="10"
AutoGenerateColumns="False"
(Nem generál automatikusan oszlopokat, kézzel állítjuk be őket.)
GridLinesVisibility="All"
(A rácsvonalak mindenhol látszanak.)
HorizontalGridLinesBrush="LightGray"
VerticalGridLinesBrush="LightGray"
(A vízszintes és a függőleges rácsvonalak is halványszürke színűek.)
BorderThickness="1"
(A külső szegény vastagsága.)
BorderBrush="LightGray"
(A külső szegély halványszürke színű.)
HeadersVisibility="Column"
(Csak az oszlopfejléceket mutatja, a sorfejléceket nem, itt nincs szükség rájuk.)
CanUserAddRows="False"
(A felhasználó nem tud új sorokat hozzáadni kézzel.)
IsReadOnly="True">
(A táblázat adatai nem szerkeszthetők.)
<DataGrid.Columns>
(Az oszlopokat manuálisan határozzuk meg.)
<DataGridTextColumn Header="Dátum"
Binding="{Binding Date}"
Width="120"/>
(Az oszlop fejléce Dátum. Adatkötés az adatmodell Date tulajdonságához. Az oszlop szélessége pontosan meg van adva. A Data Binding egy mechanizmus, amely

```
lehetővé teszi, hogy egy UI elem - pl. TextBox, ListBox - automatikusan
szinkronizálódjon egy adatforrással (pl. egy objektum vagy egy lista).
Ezáltal nem kell manuálisan frissíteni az UI-t, amikor az adatok változnak. Az oszlop
minden sora egy olyan objektumból jelenít meg adatot, amelynek van egy Date nevű
tulajdonsága.)
                        <DataGridTextColumn Header="Rendszám"
                                Binding="{Binding LicensePlate}"
                                Width="Auto"/>
                        <DataGridTextColumn Header="Autó"
                                Binding="{Binding CarType}"
                                Width="Auto"/>
                        <DataGridTextColumn Header="Km óra állás"
                                Binding="{Binding Odometer}"
                                Width="Auto"/>
                        <DataGridTextColumn Header="Irány"
                                Binding="{Binding Direction}"
                                Width="Auto"/>
                </DataGrid.Columns>
        </DataGrid>
        <TextBlock Name="NoRentalsMessage"
(Üzenet megjelenítése, ha az adott alkalmazottnak nincs bérlése.)
                Grid.Row="2"
                Grid.Column="1"
(A TextBlock a Grid 2. sorában és 1. oszlopában helyezkedik el.)
                Text="Nem bérelt még autót!"
                Visibility="Collapsed"
(Alapértelmezetten nem látható.)
                FontSize="16"
                FontWeight="Bold"
                Foreground="Red"
                HorizontalAlignment="Center"
                VerticalAlignment="Center"/>
</Grid>
```

Az EmployeeListBox_SelectionChanged függvény:

```
private void EmployeeListBox_SelectionChanged(object sender,
SelectionChangedEventArgs e)
{
        var rentals =
dbConn.GetEmployeeRentals(EmployeeListBox.SelectedItem.ToString());
(Meghívjuk a GetEmployeeRentals metódust és átadjuk paraméterként a
ListBox-ban kiválasztott elemet.)
```

```
        EmployeeTable.ItemsSource = rentals;
(A metódus által visszaadott listát a táblázathoz adjuk.)
}
```

Hozzuk létre a Rental osztályt: az osztály adattagjai az autó azonosítója, a dátum, a rendszám, a típus, a km óra állása és a ki / behajtás.

```
public int CarId { get; set; }
public string Date {  get; set; }
public string CarLicensePlate { get; set; }
public string CarType { get; set; }
public int Odometer { get; set; }
public string Direction { get; set; }

public Rental(int carId, string date, string carLicensePlate, string carType, int
odometer, string direction)
{
        CarId = carId;
        Date = date;
        CarLicensePlate = carLicensePlate;
        CarType = carType;
        Odometer = odometer;
        Direction = direction;
}
```

Az ExecuteQuery metódust töröljük, és a GetEmployees metódus módosítása:

```
List<string> results = new List<string>();
string query = "SELECT nev FROM alkalmazottak;";
try
{
        MySqlCommand cmd = new MySqlCommand(query, conn);
        using (MySqlDataReader reader = cmd.ExecuteReader())
        {
                while (reader.Read())
                {
                        results.Add(reader[0].ToString());
                }
        }
}
catch (MySqlException ex)
{
        MessageBox.Show($"SQL-lekérdezési hiba: {ex.Message}");
```

```
}
return results;
```

A GetEmployeeRentals metódus:

```
public List<Rental> GetEmployeeRentals(string employee)
(A metódus paraméterként az alkalmazott nevét kapja, és egy Rental típusú listát ad
vissza.)
{
	string employeeId = "";
(Az alkalmazott neve alapján az alkalmazott azonosítóját kell megállapítani.)
	string query = $"SELECT azon FROM alkalmazottak WHERE nev =
'{employee}'";
	try
	{
		MySqlCommand cmd = new MySqlCommand(query, conn);
		using (MySqlDataReader reader = cmd.ExecuteReader())
		{
			reader.Read();
			employeeId = reader[0].ToString();
		}
	}
	catch (MySqlException ex)
	{
		MessageBox.Show($"SQL-lekérdezési hiba: {ex.Message}");
	}
	List<Rental> rentals = new List<Rental>();
(Rental típusú lista létrehozása.)
	query = $"SELECT autoAzon, datum, kmAllas, bekiHajtas FROM parkolo
WHERE alkalmazottAzon = '{employeeId}'";
(Az autó azonosítójára a rendszám és típus lekérdezése miatt szükség lesz,
valamint a parkolo táblából a dátumot, a km óra állását és a ki / behajtást tudjuk
lekérdezni.)
	string direction = "";
(Az irányt szövegesen kell megjeleníteni a táblázatban.)
	try
	{
		MySqlCommand cmd = new MySqlCommand(query, conn);
		using (MySqlDataReader reader = cmd.ExecuteReader())
		{
			while (reader.Read())
			{
				if (Convert.ToInt32(reader[3]) == 0)
```

```
                                {
                                        direction = "ki";
                                }
                                else
                                {
                                        direction = "be";
                                }
(Ha a harmadik érték 0, akkor az irány ki, ha 1, akkor be.)
                                Rental rental = new Rental(Convert.ToInt32(reader[0]),
reader[1].ToString(), "rendszam", "tipus", Convert.ToInt32(reader[2]), direction);
                        rentals.Add(rental);
(Az objektum létrehozása: az azonosító és a dátum a lekérdezésből, a rendszámot
és a típust még nem tudjuk, a km óra állása és az irány a lekérdezésből.)
                                }
                        }
                        foreach (var rental in rentals)
(Bejárjuk a listát.)
                        {
                                rental.CarLicensePlate = GetCarLicensePlate(rental.CarId);
(A rendszámot a GetCarLicensePlate metódussal kérdezzük le, átadjuk az autó
azonosítóját paraméterként.)
                                rental.CarType = GetCarType(rental.CarId);
(A típust a GetCarType metódussal kérdezzük le, átadjuk az autó azonosítóját
paraméterként.)
                        }
                }
                catch (MySqlException ex)
                {
                        MessageBox.Show($"SQL-lekérdezési hiba: {ex.Message}");
                }
                return rentals;
(A metódus visszatérési értéke a lista az objektumokkal.)
}
```

A GetCarLicensePlate metódus:

```
public string GetCarLicensePlate(int carId)
{
        string carLicensePlate = "";
        string query = $"SELECT rendszam FROM autok WHERE azon = '{carId}'";
        try
        {
                MySqlCommand cmd = new MySqlCommand(query, conn);
```

```
                using (MySqlDataReader reader = cmd.ExecuteReader())
                {
                        reader.Read();
                        carLicensePlate = reader[0].ToString();
                }
        }
        catch (MySqlException ex)
        {
                MessageBox.Show($"SQL-lekérdezési hiba: {ex.Message}");
        }
        return carLicensePlate;
}
```

A GetCarType metódus:

```
public string GetCarType(int carId)
{
        string carType = "";
        string query = $"SELECT tipus FROM autok WHERE azon = '{carId}'";
        try
        {
                MySqlCommand cmd = new MySqlCommand(query, conn);
                using (MySqlDataReader reader = cmd.ExecuteReader())
                {
                        reader.Read();
                        carType = reader[0].ToString();
                }
        }
        catch (MySqlException ex)
        {
                MessageBox.Show($"SQL-lekérdezési hiba: {ex.Message}");
        }
        return carType;
}
```

Ha valamelyik alkalmazott nem bérelt még autót, akkor egy felirat jelenik meg:

```
if (rentals.Count == 0)
(Ha lista üres, akkor az adott alkalmazottnak nincsenek bérlései, ezért láthatóvá
tesszük a szöveget.)
{
        NoRentalsMessage.Visibility = Visibility.Visible;
}
```

```
else
(Egyébként eltüntetjük a szöveget.)
{
        NoRentalsMessage.Visibility= Visibility.Collapsed;
}
```

A táblázat felett jelenjen meg az XY bérlései szöveg:

```
EmployeeName.Text = EmployeeListBox.SelectedItem.ToString() + " bérlései:";
EmployeeName.Visibility = Visibility.Visible;

if (rentals.Count == 0)
{
        NoRentalsMessage.Visibility = Visibility.Visible;
}
else
{
        NoRentalsMessage.Visibility = Visibility.Collapsed;
}
```

Az Új bérlés felvétele funkció megvalósítása

A főablak alján helyezzünk el egy gombot, ami megnyitja az új bérlés felvételét tartalmazó ablakot. A Grid 4 sorból álljon, a ListBox esetében 4 cellát kell összevonni, a gomb a Grid 3. sorába és 1. oszlopába kerül.

```
<Button Content="Új bérlés felvétele"
        Click="OpenNewRentalWindow"
        Grid.Row="3"
        Grid.Column="1"
        Width="120"
        Height="30"/>
```

A Click eseményhez tartozó kód:

```
private void OpenNewRentalWindow(object sender, RoutedEventArgs e)
{
        NewRentalWindow newRentalWindow = new NewRentalWindow();
        newRentalWindow.Show();
}
```

Készítsük el az új ablakot, amin az űrlapot el tudjuk helyezni.

Kattintsunk jobb egérgombbal a projekt nevére a Solution Explorer-ben. Válasszuk az Add / New Item opciót. Válasszuk a Window (WPF) lehetőséget, a név legyen NewRentalWindow.xaml.

Ezután az új ablak a főablakból megnyitható.

Készítsük el az űrlapot az új ablakban.

Az új ablak szélessége legyen 400 képpont, a magassága legyen 500 képpont.

```
<Grid>
        <Grid.RowDefinitions>
                <RowDefinition Height="*"/>
                <RowDefinition Height="*"/>
                <RowDefinition Height="*"/>
                <RowDefinition Height="*"/>
                <RowDefinition Height="*"/>
                <RowDefinition Height="*"/>
                <RowDefinition Height="*"/>
                <RowDefinition Height="*"/>
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
                <ColumnDefinition Width="*"/>
                <ColumnDefinition Width="2*"/>
        </Grid.ColumnDefinitions>
        <TextBlock Text="Új bérlés felvitele"
                Grid.ColumnSpan="2"
                Grid.Row="0"
                Grid.Column="0"
                FontSize="24"
                HorizontalAlignment="Center"
                VerticalAlignment="Center"/>
        <TextBlock Text="Név:"
                Grid.Row="1"
                Grid.Column="0"
                FontSize="16"
                HorizontalAlignment="Right"
                VerticalAlignment="Center"/>
        <ComboBox Name="cmbName"
                Grid.Row="1"
                Grid.Column="1"
```

```
            HorizontalAlignment="Left"
            VerticalAlignment="Center"
            Width="200"
            Height="25"
            Margin="10,0,0,0"/>
    <TextBlock Text="Autó:"
            Grid.Row="2"
            Grid.Column="0"
            FontSize="16"
            HorizontalAlignment="Right"
            VerticalAlignment="Center"/>
    <ComboBox Name="cmbCar"
            Grid.Row="2"
            Grid.Column="1"
            HorizontalAlignment="Left"
            VerticalAlignment="Center"
            Width="200"
            Height="25"
            Margin="10,0,0,0"/>
    <TextBlock Text="Dátum:"
            Grid.Row="3"
            Grid.Column="0"
            FontSize="16"
            HorizontalAlignment="Right"
            VerticalAlignment="Center"/>
    <DatePicker Name="dpRentalDate"
            Grid.Row="3"
            Grid.Column="1"
            HorizontalAlignment="Left"
            VerticalAlignment="Center"
            Width="200"
            Height="25"
            Margin="10,0,0,0"/>
    <TextBlock Text="Időpont:"
            Grid.Row="4"
            Grid.Column="0"
            FontSize="16"
            HorizontalAlignment="Right"
            VerticalAlignment="Center"/>
    <TextBox Name="tbRentalTime"
            Grid.Row="4"
            Grid.Column="1"
            HorizontalAlignment="Left"
            VerticalAlignment="Center"
```

```
            Width="200"
            Height="25"
            Margin="10,0,0,0"/>
    <TextBlock Text="Km óra állása:"
            Grid.Row="5"
            Grid.Column="0"
            FontSize="18"
            HorizontalAlignment="Right"
            VerticalAlignment="Center"/>
    <TextBox Name="tbKm"
            Grid.Row="5"
            Grid.Column="1"
            HorizontalAlignment="Left"
            VerticalAlignment="Center"
            Width="200"
            Height="25"
            Margin="10,0,0,0"/>
    <StackPanel Orientation="Horizontal"
            Grid.ColumnSpan="2"
            Grid.Row="6"
            Grid.Column="0"
            HorizontalAlignment="Center"
            VerticalAlignment="Center">
            <RadioButton Name="rbOut"
                    Content="Kihajtás"
                    IsChecked=”True”
                    GroupName="Direction"
                    FontSize="16"
                    HorizontalAlignment="Center"
                    VerticalAlignment="Center"
                    Margin="0,0,20,0"/>
            <RadioButton Name="rbIn"
                    Content="Behajtás"
                    GroupName="Direction"
                    FontSize="16"
                    HorizontalAlignment="Center"
                    VerticalAlignment="Center"
                    Margin="10,0,0,0"/>
    </StackPanel>
    <Button Name="btnSave"
            Grid.ColumnSpan="2"
            Grid.Row="7"
            Grid.Column="0"
            Content="Felvétel"
```

```
                HorizontalAlignment="Center"
                VerticalAlignment="Center"
                Width="120"
                Height="30"
                Click="btnSave_Click"/>
</Grid>
```

Időválasztót próbáltam keresni, de csak olyan külső könyvtárat találtam, amiben dátum- és időválasztó van, viszont, dátumválasztó van az űrlapon, így az időpont egyszerű szövegdoboz lesz. A km óra állása esetében szintén egyszerű szövegdobozt használunk.

A nevet és az autót (rendszám) legördulő listából lehet kiválasztani.

NewRentalWindow.xaml.cs:

```
private DatabaseConnection dbConn;

string connStr = "Server=localhost;Database=parkolohaz;User ID=root;";
dbConn = new DatabaseConnection(connStr);
try
{
        dbConn.OpenConnection();
}
catch (Exception ex)
{
        MessageBox.Show($"Hiba történt a kapcsolódás során: {ex.Message}");
}
var employees = dbConn.GetEmployees();
(A főablakban is ezt a metódust használtuk az alkalmazottak nevének
lekérdezéséhez, itt is használhatjuk.)
cmbName.ItemsSource = employees;
(A legördülő lista adatainak forrása a metódus által visszaadott lista.)
cmbName.SelectedIndex = 0;
(A legördülő lista alapértelmezetten nem mutatja egyik elemet sem, de beállítható,
hogy melyik elem legyen kiválasztva.)
var cars = dbConn.GetCars();
(Az alkalmazottak neveit lekérdező metódushoz hasonlóan működő metódus, ami a
rendszámokat kérdezi le.)
cmbCar.ItemsSource = cars;
(A legördülő lista adatainak forrása a metódus által visszaadott lista.)
cmbCar.SelectedIndex = 0;
```

(A legördülő lista alapértelmezetten nem mutatja egyik elemet sem, de beállítható, hogy melyik elem legyen kiválasztva.)

DatabaseConnection.cs:

```
public List<string> GetCars()
{
	List<string> results = new List<string>();
	string query = "SELECT rendszam FROM autok;";
	try
	{
		MySqlCommand cmd = new MySqlCommand(query, conn);
		using (MySqlDataReader reader = cmd.ExecuteReader())
		{
			while (reader.Read())
			{
				results.Add(reader[0].ToString());
			}
		}
	}
	catch (MySqlException ex)
	{
		MessageBox.Show($"SQL-lekérdezési hiba: {ex.Message}");
	}
	return results;
}
```

Ellenőrzések az űrlap elküldésekor:

```
private void btnSave_Click(object sender, RoutedEventArgs e)
{
	if (dpRentalDate.SelectedDate == null)
(Ha nincs dátum kiválasztva a dátum választóban, akkor hibaüzenetet jelenítünk meg.)
	{
		MessageBox.Show("Kérem, válasszon ki egy dátumot!", "Hiba",
MessageBoxButton.OK, MessageBoxImage.Warning);
(Az üzenetablak szövege, címe, a megjelenítendő gomb és ikon.)
		dpRentalDate.Focus();
(Az üzenetablak bezárása után erre a mezőre állítjuk a fókuszt.)
		return;
(Mivel hiba történt, a metódus nem lehet tovább, mert akkor a hibák ellenére eljut a végére.)
```

```
        }
        if (string.IsNullOrWhiteSpace(tbRentalTime.Text))
        {
(Ha az időpont szövegdoboz üres vagy csak whitespace karaktereket tartalmaz,
akkor hibaüzenetet jelenítünk meg.)
                MessageBox.Show("Kérem adjon meg egy időpontot!", "Hiba",
MessageBoxButton.OK, MessageBoxImage.Warning);
        tbRentalTime.Focus();
        return;
        }
        if(!Regex.IsMatch(tbRentalTime.Text,
@"^([01]?\d|2[0-3]):([0-5]?\d):([0-5]?\d)$"))
(Ha az időpont szövegdoboz szövege nem felel meg a reguláris kifejezésnek -
óó:pp:mm -, akkor hibaüzenetet jelenítünk meg.)
        {
                MessageBox.Show("Kérem, adjon meg érvényes időpontot (óó:pp:mm
formátumban)!", "Hiba", MessageBoxButton.OK, MessageBoxImage.Warning);
        tbRentalTime.Focus();
        return;
        }
        if (string.IsNullOrWhiteSpace(tbKm.Text))
        {
                MessageBox.Show("Kérem adjon meg egy km óra állást!", "Hiba",
MessageBoxButton.OK, MessageBoxImage.Warning);
        tbKm.Focus();
        return;
        }
        if (!int.TryParse(tbKm.Text, out int kmOraAllas) || kmOraAllas <= 0)
        {
(Ha a km szövegdoboz nem alakítható int típusúvá, vagy igen, de nem nagyobb,
mint 0, akkor hibaüzenetet jelenítünk meg.)
                MessageBox.Show("A kilométeróra állásnak pozitív egész számnak
kell lennie!", "Hiba", MessageBoxButton.OK, MessageBoxImage.Warning);
        tbKm.Focus();
        return;
        }
        dbConn.NewRentalSave(cmbName.Text, cmbCar.Text, dpRentalDate.Text,
tbRentalTime.Text, Convert.ToInt32(tbKm.Text), Convert.ToInt32(rbIn.IsChecked));
(Ha túljutottunk az ellenőrzéseken, akkor meghívjuk a NewRentalSave metódust,
átadjuk neki az űrlapon bekért adatokat és végrehajtja a lekérdezést. A km óra állást
számként. Ha a behajtás rádiógomb be van jelölve, akkor az IsChecked metódus
true-t vagyis 1-et ad vissza és 1-el jelöljük az adatbázisban a behajtást. Ha nincs
bejelölve, akkor a kihajtás van bejelölve, viszont akkor az IsChecked metódus false-t
```

```
vagyis 0-t ad vissza, ami megfelel a kihajtásnak. Így nem szükséges mindkét
rádiógombbal kapcsolatban adatot átadni a metódusnak, hiszen kizárják egymást.)
        this.Close();
(Az adatok felvitele után az űrlap ablakot bezárjuk.)
}
```

A NewRentalSave metódus:

```
public void NewRentalSave(string name, string car, string date, string time, int km, int
inOut)
(A metódus paraméterként megkapja az alkalmazott nevét, az autó rendszámát, a
dátumot, az időt, a km óra állást és be / kihajtást.)
{
        date = date.Replace(". ", "-");
        date = date.Remove(date.Length - 1);
        date = "'" + date + " " + time + "'";
(A dátumot a dátum választó mezőből úgy kapjuk meg, hogy az év és a hónap,
valamint a hónap és a nap között pont és szóköz van, ezt lecseréljük kötőjelre. A
nap utáni pontot eltávolítjuk. Mivel az adatbázisban a dátum és az idő a dátum
mezőben van tárolva, egy szóközzel összefűzzük őket. Valamint az egészet
aposztrófok közé tesszük. Ez a szöveg megfelel annak a formátumnak amit az
adatbázis vár.)
        string query = $"INSERT INTO parkolo (datum, autoAzon, alkalmazottAzon,
kmAllas, bekiHajtas) VALUE ({date}, {GetCarId(car)}, {GetEmployeeId(name)}, {km},
{inOut});";
(A lekérdezésből kihagyjuk a parkolás azonosítóját, mivel az automatikusan
növekszik. A dátum a date változóban van. Az autó azonosító miatt meghívjuk a
GetCarId metódust, átadjuk neki a rendszámot és megkapjuk az azonosítót. Az
alkalmazott azonosító miatt meghívjuk a GetEmployeeId metódust, átadjuk neki a
nevet és megkapjuk az azonosítót. A km óra állás a km változóban van. A be /
kihajtás az inOut változóban van.)
        try
        {
                using(MySqlCommand cmd = new MySqlCommand(query, conn))
                {
                        cmd.ExecuteNonQuery();
(Ez a metódus nem ad vissza adatot, INSERT / UPDATE / DELETE esetén
használatos.)
                }
        }
        catch (MySqlException ex)
        {
                MessageBox.Show($"SQL-lekérdezési hiba: {ex.Message}");
```

```
        }
}
```

A GetEmployeeId metódus:

```
public int GetEmployeeId(string employeeName)
(Paraméterként az alkalmazott nevét kapja a metódus és az azonosítóját adja
vissza.)
{
        int employeeId = 0;
        string query = $"SELECT azon FROM alkalmazottak WHERE nev =
'{employeeName}'";
        try
        {
                MySqlCommand cmd = new MySqlCommand(query, conn);
                using (MySqlDataReader reader = cmd.ExecuteReader())
                {
                        reader.Read();
                        employeeId = Convert.ToInt32(reader[0]);
                }
        }
        catch (MySqlException ex)
        {
                MessageBox.Show($"SQL-lekérdezési hiba: {ex.Message}");
        }
        return employeeId;
}
```

A GetCarId metódus:

```
public int GetCarId(string carLicensePlate)
(Paraméterként az autó rendszámát kapja a metódus és az azonosítóját adja
vissza.)
{
        int carId = 0;
        string query = $"SELECT azon FROM autok WHERE rendszam =
'{carLicensePlate}'";
        try
        {
                MySqlCommand cmd = new MySqlCommand(query, conn);
                using (MySqlDataReader reader = cmd.ExecuteReader())
                {
                        reader.Read();
```

```
                    carId = Convert.ToInt32(reader[0]);
                }
            }
            catch (MySqlException ex)
            {
                MessageBox.Show($"SQL-lekérdezési hiba: {ex.Message}");
            }

            return carId;
}
```

A főablak bezárásakor zárjuk le az adatbázis kapcsolatot. Az OnClosed metódus akkor hívódik meg, ha az ablakot bezárjuk, ezt felülírjuk.

```
protected override void OnClosed(EventArgs e)
{
        dbConn.CloseConnection();
(Az adatbázis kapcsolat lezárása.)
        base.OnClosed(e);
}
```

A base kulcsszó arra utal, hogy a származtatott osztály (tehát a főablak osztálya) az ősosztály (általában az Window osztály) metódusát szeretné meghívni.