

1. Bevezet1

1.1.	Témaválasztás	1
1.2.	Programnyelv C#	1
1.3.	XAML	2
1.4.	Programnyelv SQL	2
1.5.	Elkészítés során használt programok	3
1.5.1.	Microsoft Visual Studio 2019	3
1.5.2.	XAMPP	3
1.5.3.	dbForge Studio for MYSQL	4
1.6.	Tervezés	4
1.7.	Ötlet	5
1.8.	Adatbázis	6

2. Fejlesztői dokumentáció

2.1.	Első Futtatás	8
2.2.	Adatbázis	9
2.3.	Design	9
2.4.	Bejelentkező felület	10
2.5.	Menü rendszer	12
2.6.	Kezdőlap	12
2.7.	Dolgozó Felvétel	13
2.7.1.	Felvétel	13
2.7.2.	Módosítás	14
2.7.3.	Törlés	15
2.7.4.	Keresés	15
2.8.	Munkaórák	15
2.8.1.	Dolgozik-e	15
2.8.2.	Csekkoló	15

2.8.3.	Módosítás	16
2.8.4.	Havi bérék	16
2.8.5.	Havi Zárás	17
2.9.	Naptár	17
2.9.1.	„Ma” gomb	18
2.9.2.	Alap érték	18
2.9.3.	Napi bevétel-kiadás „bevitel” gomb	19
2.10.	Kapcsolat	20
2.11.	Beállítás	20
2.11.1.	Felhasználók módosítása	20
3.	Felhasználói dokumentáció	22
3.1.	Bejelentkező felület	22
3.2.	Kezdőlap	23
3.3.	Dolgozók	24
3.4.	Munkaórák	25
3.5.	Naptár	26
3.6.	Kapcsolatok	27
3.7.	Beállítások	28
4.	Összefoglalás	30
5.	Irodalomjegyzék	32

1. BEVEZETÉS

A megszokottól eltérően, gondoltam kicsit színesebbé teszem ezt a méltó feladatot így 'zárásként'. Próbáltam felépíteni egy olyan Windows alkalmazást, amit a későbbiekben is használni tudok, akár tovább tudom adni. Hosszas tanakodás után, tehát egy munkaügyi nyilvántartóra esett a választásom.

A tudást, amit felhasználtam, iskolai éveim alatt és önfejlesztésből szedtem össze, természetesen nagy segítséget jelentett a talált dokumentációk vagy éppen könyvek, amiket felhasználtam. Mindig is nagy álmom volt, hogy egyszer megírhasam a saját programom, amit nem érzek feleslegesnek és minden erőfeszitésem beleteszem.

1.1. Témaválasztás

Iskola mellett, moziban dolgoztam, ez a munka közel állt hozzám, így próbáltam abból kiindulni mit lenne érdemes csinálni mi az, amit ha végeztem vele, újra elővennék, fejleszteném és még érdekelne is. Azért wpf programra esett a választásom, mert szerettem volna egy normálisan működő alkalmazást, amihez értek és hasznát is venném. A program fő célja az adatok dokumentálása, számolása a mindennapi adminisztráció megkönnyítése. A program felépítése során ütköztem sajnos hibákba és nagyon sokszor újra kellett gondolni pár funkciót, mivel az nem volt hasznos vagy éppen nem működött megfelelően.

1.2. Programnyelv C#

A C# programozási nyelv a Microsoft új fejlesztési környezetével, a 2002-ben megjelent Visual Studio.NET programcsomaggal, annak részeként jelent meg. A Microsoft által a .NET keretrendszer részeként kifejlesztett objektumorientált programozási nyelv. A nyelv hosszú múlttal rendelkezik, elődjének a C++ nyelvet mondhatjuk. A C, C++ nyelv komplexitása, a fejlesztések hosszabb időciklusa azt eredményezte, hogy a C, C++ programozók olyan nyelvet keressenek, amelyik jobb produktivitást eredményez, ugyanakkor megtartja a C, C++

hatékonyságát. Egy C# program tetszőleges számú fordítási egységből (modulból) állhat. Szokás ezen modulok vagy fájlok összességét projektnek nevezni. A program ezen modulokban definiált osztályok összessége. Egy fájl tartalmazza az egyes modulok közti hivatkozásokat (using), osztálytípus-, változó- és függvénydeklarációkat. A nyelv jellemzője, hogy a program több fordítási egységből modulból vagy fájlból áll.

1.3. XAML

XAML, azaz eXtensible Application markup Language, a Microsoft változata az XML-ből grafikus felhasználó felület jellemzésére. Az XAML-ben készíthetünk vezérlő elemeket, amik segítenek a felhasználónak könnyebben értelmezni mit is csinál a program. Rengeteg féle esemény áll rendelkezésünkre, de a legtöbbet a felhasználó egere vagy billentyűzete vált ki.

1.4. Programnyelv SQL

SQL = Structured Query Language (struktúrált lekérdező nyelv). A relációs adatbázis-kezelés szabványos nyelve. Deklaratív nyelvek csoportjába tartozik (nem algoritmikus nyelv) nem tartalmaz algoritmus szerkezeteket (elágazás, ciklus stb.), de algoritmikus nyelvekbe beépíthető (beágyazott SQL). Az SQL halmaz orientált a relációs algebrán alapuló nyelv.

Négy utasítást foglal magába:

1. Adatdefiníciós utasítások
2. Adatmanipulációs utasítások
3. Adatkezelő utasítások
4. Adatvezérlő utasítások

1.5. Elkészítés során használt programok

1.5.1. Microsoft Visual Studio 2019

A Visual studio a Microsoft több programozási nyelvet tartalmazó fejlesztőkörnyezet, ami az évek során egyre több programnyelvvvel bővült. A legfrissebb verziója a 2019-es Visual Studio ami 2019. Április 2-án jelent meg.

Jelenleg a F#, C++, C# és Visual Basic programozási nyelveket, valamint az XML-t támogatja.



1.5.2. XAMPP



Az xampp egy szabad és nyílt forrású platformfüggetlen webszerver-szoftvercsomag, amelynek legfőbb alkotóelemei az Apache webszerver, a MariaDB, valamint a PHP és a Perl programozási nyelvek végrehajtó rendszerei. Ez a szoftvercsomag egy integrált rendszert alkot, amely webes alkalmazások készítését, tesztelését és futtatását célozza, és ehhez egy csomagban minden szükséges összetevőt tartalmaz. A cél az volt, hogy a web-tervezők és programozók internetes kapcsolat nélkül fejleszthetik és tesztelhetik alkalmazásaikat.

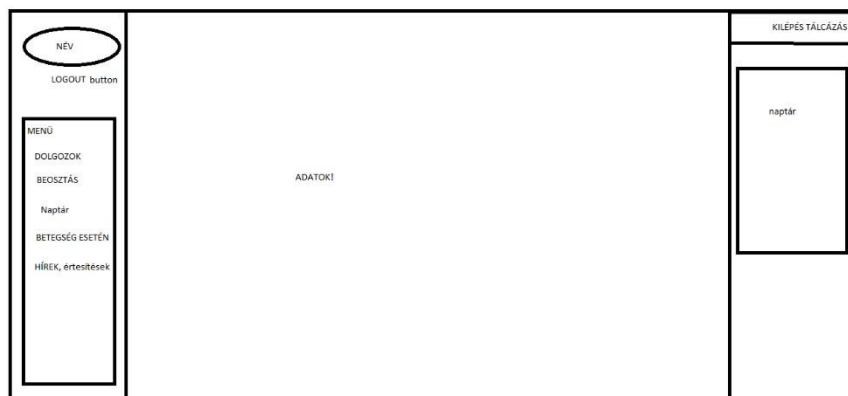
1.5.3. dbForge Studio for MySQL

A dbForge Stuidiot egy strukturált lekérdezési nyelven alapuló relációs adatbázis-kezelő rendszer. Az alkalmazás számos célra használható beleértve az adattárház, az e- kereskedelem és a naplózás. A mySQL leggyakoribb használata azonban egy webes adatbázis kezelés. A programot a Devart nevezetű cég fejleszti, akik adatbázisokkal foglalkoznak.

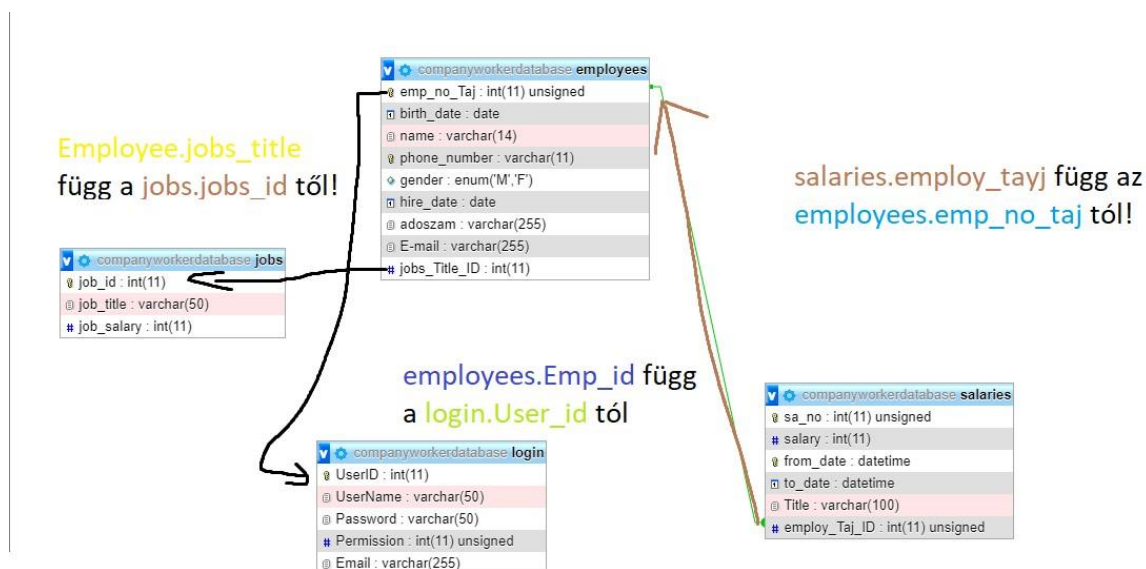


1.6. Tervezés

Ilyen projekteknél érdemes először átgondolni mit és hogyan szeretnénk csinálni. Ötleteinket nem árt ledokumentálni és utána összesíteni hogyan tudnánk azt megvalósítani. A program írása előtt én is így tettem, először kigondoltam a rendszert, majd összeraktam a program 'szívét' a MYSQL adatbázist. Az adatbázis elkészítése után, sokkal egyszerűbb volt elkezdeni a programot. Egy paint képben rajzoltam le, hogy hogyan is képzelem el a programot és azt, hogy nagyjából hogy fog kinézni.



Először átnéztem, hogy miket is vettünk órákon, átgondoltam, mégis nézzen ki a program, mikre vagyok képes a tudásom alapján. Sajnos Entity Frameworkot nem tudtam még használni mert azt később kezdtük el tanórán, mint én a programot, és még nem voltam teljesen biztos a dolgomban. SQL parancsokban viszont már jártas voltam és sokat gyakoroltuk, ezért ezt a módszert választottam. Természetesen a programnál először az alapokat csináltam meg.

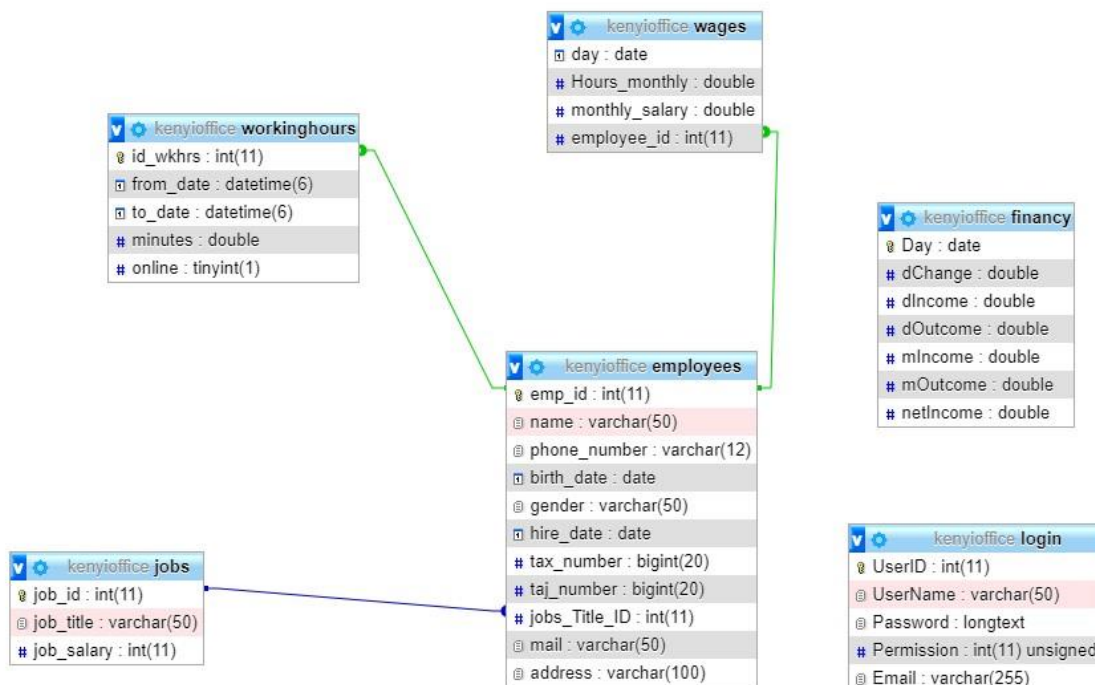


A felhasználói, bejelentkező felületet készítettem el, ezután egyszerűbb volt összerakni a programot, mert láttam, hogy fog kinézni, ezután elkezdtem egy minimális felületet létrehozni mindennek, majd kódolni.

1.7. Ötlet

Az ötlet, a volt munkahelyemről jött, ahol sajnos mindig bajlódni kellett a ki-be jelentkezéssel és a sok jelenlétiív kitöltésével. Ezért mikor mondták, hogy írunk kell egy szakdolgozatot, egyből ez jutott eszembe, hogy milyen jó ötlet lenne egy ilyen programot megírni, amit talán még a cég maga is tudna használni. Akár a saját vállalkozásom is a közeljövőben. A program lényege, hogy ha a dolgozó megérkezik a munkahelyére, csippant egyet majd elkezd a program számolni az óráit, párhuzamosan a fizetéssel, és ebből a nap végére kiszámolja az aznapi bért. Ezt tovább viszi a rendszer, ami azt jelenti, hogy nem csak napi, hanem havi bért is tud nézni az adott dolgozó. A programba tudunk bevinni adatokat, új dolgozókat, és módosításra esetleg törlésre is van lehetőség, viszont csak annak, akinek jogosultsága van a rendszerhez. Mindenki tudja dokumentálni a saját bevételét-kiadását, amit a program egy napi és egy havi nettó értékévé számol, ezzel megkönnyítve a napi és a havi zárást.

1.8. Adatbázis



Az adatbázis 6 táblából áll. Itt törekedtem arra, hogy ahol szükséges, legyen kapcsolat a táblák között. Ahogy a képen is látható, a 'login' tábla tartalmazza a felhasználókat. Itt kap a felhasználó egy automatikus ID-t, felhasználónevet, jelszót, jogosultságot és e-mailt. Az „Employees” tábla tartalmazza dolgozókat. Itt a munkavállalók egyes adatait tároljuk. Az 'emp_ID' a „Primary key”, ami a megkülönböztetője a dolgozóknak, ez mindig egyedi lehet. A 'Jobs_Title_ID' a másodlagos kulcs a „Jobs” táblához. Itt az „Employee.jobs_Title_ID” függ a jobs_ID- től. A „Workinghours” táblába a dolgozók napi munkaórája található, ebből a táblából kérdezi le vagy tölti fel, a ki- és becekkolt idejét a dolgozónak. Ha befejezték a munkát, akkor ezeket az órákat a „Minutes” sorába menti bele. Az online sor segít a programnak, hogy a dolgozó éppen dolgozik-e vagy sem, ha a személy nem dolgozik, akkor a program innen tudja, hogy el kell indítani a számlálóját, viszont ha dolgozik, akkor pedig leállítja. A 'Wages' táblában találhatóak meg a bérek, dátumra osztva és összesítve a hónap végén. Ide menti le a rendszer a dolgozó bérét, ami egy másodlagos employee_ID-val van az adott személyhez kötve. A 'Financy' táblában a cég pénzügyei vannak számontartva. Itt megtalálható, hogy az adott napon mennyi volt a bevétel/kiadás, mennyi a havi bevétel,

kiadás és a nettó bevétel az adott napra. Ezekbe a sorokba a program folyamatosan frissíti az adatokat. A 'Jobs' táblában tároljuk a munkákat. Ezeket a munkákat nem lehet megváltoztatni, ezeket csak csatolásra használjuk. Minden dolgozóhoz van rendelve egy munkakör. Ebben a táblában szerepelnek a munkához kötött fizetések is. A program a 'Jobb_salary' sort használja a bérek számolásához.

2. FEJLESZTŐI DOKUMENTÁCIÓ

A továbbiakban a program szerkezeti felépítéséről, háttéréről lesz szó. Ugyebár a program a fent említett c#, SQL, XAML nyelvben íródott, itt-ott pár LINQ paranccsal. A lekérdezéseket a program MySQL lekérdezéssel teszi meg.

2.1. Első Futtatás

Első futtatás során a program ellenőrzi SQL parancsokkal, hogy a szükséges adatbázis és táblák léteznek-e. Ha ezeket nem találja a program, elkészíti őket, csinál egy alap felhasználót, amivel be tud lépni az adott illető. Az ellenőrzést sima SQL parancsokkal vittem véghez, amiket a lenti képen meg is tekinthetünk.

```
private void databaseTest()
{
    con = new MySqlConnection(databasetrier);
    con.Open();
    string s0 = "CREATE DATABASE IF NOT EXISTS `KenyiOffice`";
    cmd = new MySqlCommand(s0, con);
    cmd.ExecuteNonQuery();
    cmd.Dispose();
    con.Close();
    con.Dispose();

    con = new MySqlConnection(ConnectionString);
    con.Open();
    using (cmd = new MySqlCommand("CREATE TABLE IF NOT EXISTS `login` (" +
        "`UserID` INT(11) AUTO_INCREMENT, " +
        "`UserName` VARCHAR(50) NOT NULL, " +
        "`Password` LONGTEXT NOT NULL, " +
        "`Permission` INT(11) UNSIGNED NOT NULL, " +
        "`Email` VARCHAR(255) NOT NULL, " +
        "PRIMARY KEY(UserID));", con))
    {
        cmd.ExecuteNonQuery();
        cmd.Dispose();
    }
}
```

Itt a program megadja az adatbázis nevét, ami itt „KenyiOffice” és feltölti a táblákat sorokkal. Ezeknek a soroknak a tulajdonságait beállítja, hogy ahhoz többet ne kelljen hozzányúlani.

2.2. Adatbázis

Az adatbázishoz a program egy connectionString-en keresztül kapcsolódik, ami így néz ki:

```
static string ConnectionString =
    "datasource=127.0.0.1;port=3306;username=root;password=;Initial Catalog=kenyioffice;;
```

Az adatbázis feltöltést MYSQL segítségével vittem véghez. Az SQL kódot egy mysql command stringként töltöttem fel, amit a program a végén végrehajt az adott kapcsolaton. Ha ezt az utasítást végrehajtotta eldobja a command stringet és várja a másik feltételt.

```
using (cmd = new MySqlCommand("insert into jobs (`job_title`, `job_salary`) +
    "Select 'Büfés', 1200 Where not exists(select * from jobs where job_title='Büfés');", con))
{
    cmd.ExecuteNonQuery();
    cmd.Dispose();
}
```

A programírás során az adatbázissal való kommunikáláshoz mindenhol SQL injekciót használtam.

2.3. Design

A dizájnt az XAML Material Design eszköztárával oldottam meg és ezzel díszítettem ki a programot. Leginkább, ahol látszódnak a változások azok nem más, mint a gombok és vannak különböző ikonok, amik megtalálhatóak a bejelentkező vagy a felhasználói felületen. Pár helyen észrevehetőek a különböző effektek, mint például a Textbox alatti vonalak. A programból az összes Windows széria gomb el lett távolítva, helyettük én készítettem új gombokat. Az ablakokon ki van kapcsolva az átméretezés és el vannak tüntetve a Windows stílusok.

```
xmlns:materialDesign="http://materialdesigninxaml.net/winfx/xaml/themes"
xmlns:local="clr-namespace:Login"
mc:Ignorable="d"
Title="LoginScreen" Height="450" Width="800" WindowStartupLocation="CenterScreen" WindowStyle="None" ResizeMode="NoResize">
```

2.4. Bejelentkező felület

A „Login” felület multifunkcionális, mert egyszerre 2 dologra is használható, mint bejelentkező felület vagy egy gomb megnyomásával átvált munkaóra számlálóra. A felhasználók egy Login táblába vannak mentve. A program a gomb lenyomása után egy SQL lekérdezéssel ellenőrzi, hogy a megadott adatok helyesek-e, tartoznak-e valamilyen felhasználóhoz. A hozzájuk tartozó jelszót a program nem menti le, viszont ha leellenőrizte, hogy jó-e a jelszó, akkor eldobja az adatot. Fejlesztési célokból a bejelentkezés úgy lett megírva, hogy lehessen hozzá egyszerűen adni egy sima felületet, amit a dolgozók tudnak használni, de erre még nem került sor.

```
if (reader["Password"].ToString().Equals(DataManager.CreateMD5(txtPassword.Password).ToLower()))
{
    UserInfo.UserName = txtUsername.Text.ToString();
    UserInfo.permission = reader["Permission"].ToString();
    UserInfo.email = reader["Email"].ToString();
    if (UserInfo.permission == "1")
    {
        message = "1";
    }
    else
    {
        message = "0";
    }
}
```

A munkaszámlálóra váltva, a beírt kód alapján kérdezi le, hogy a dolgozó éppen dolgozik-e vagy pihen. Ha a dolgozó nem dolgozik és elindítja a számlálóját, akkor a program beilleszti az időpontot a kezdeti értékbe, az adatbázisba és az online táblát igazzá teszi.

```
public static void StartOrEnd(string ID, DateTime? from, DateTime? to, double? minutes, bool online)
{
    MySqlConnection con = new MySqlConnection(ConnectionString);
    con.Open();
    MySqlCommand cmd = new MySqlCommand("UPDATE workinghours SET from_date=@from, to_date=@to, minutes=@minutes, online=@online" +
        "WHERE id_wkhhrs=@ID", con);
    cmd.Parameters.AddWithValue("@ID", ID);
    cmd.Parameters.AddWithValue("@from", from);
    cmd.Parameters.AddWithValue("@to", to);
    cmd.Parameters.AddWithValue("@minutes", minutes);
    cmd.Parameters.AddWithValue("@online", online);
    cmd.ExecuteNonQuery();
    con.Close();
}
```

Ha a dolgozó éppen dolgozik és ki szeretne csekkolni, akkor a program a kód beírása után ezt észlelti és lezárja a dolgozó számlálóját. Beilleszti az adott időt az adatbázisba, és az online táblát hamissá teszi.

2.5. Menü rendszer

Ha a felhasználó sikeresen bejelentkezett a programba, egy menü fogja várni, amivel különböző lapokhoz tud hozzáférni. A menüpontok <ListView> -al lettek létrehozva, amit ha szeretnék be is tudunk csukni, és akkor csak a menüpont ikonjait látjuk. Ezekre a menüpontokra linkeltem rá <UserControl> lapokat, amiket a főoldal 'gyerekeiként' adtam hozzá. Így az adott felület nem változik, viszont a menüknek kijelölt terület igen.

```
public void ListViewMenu_SelectionChanged(object sender, SelectionChangedEventArgs e)
{
    int index = ListViewMenu.SelectedIndex;
    MoveCursorMenu(index);

    switch (index)
    {
        case 0:
            GridPrincipal.Children.Clear();
            GridPrincipal.Children.Add(new Kezdolap());
            break;
        case 1:
            GridPrincipal.Children.Clear();
            GridPrincipal.Children.Add(new DolgozokData());
            break;
        case 2:
            GridPrincipal.Children.Clear();
            GridPrincipal.Children.Add(new Munkaóra());
            break;
    }
}
```

2.6. Kezdőlap

A program futása során az első ablak, ami meg fog jelenni az a kezdőlap lesz. A kezdőlapon elhelyezte pár adatot, mint például: éppen dolgozó emberek, cégnél dolgozó emberek száma, havi bevétel. Valamint még egy digitális óra, ami mutatja a pontos időt.

2.7. Dolgozó Felvétel

2.7.1. Felvétel

A „Dolgozófelvétel” menüpontban lehet új dolgozót felvenni, esetleg módosítani vagy kitörölni. Ezeknek a programkódját külön egy DataManager osztályban tárolom és azokat statikussá téve, elérhető teszem az összes osztály számára. A dolgozó felvételt egy SQL injekcióval oldottam meg. Ha megnyomásra kerül a „Felvétel” gomb, egy új ablak jelenik meg, ahol be lehet vinni a felvenni kívánt dolgozó adatait. Itt a dolgozók nemét külön <combobox>ból lehet kiválasztani, amit a program a 'Menüpont'-ra kattintva tölt fel a megfelelő elemekkel. A munkakört egy külön adatbázis táblából szedi le a rendszer, amit az injekciónál a munkakör száma alapján indexeli be az „Employee” táblába. Ahova lehetett raktam ellenőrzéseket, viszont ahol csak számokat tudtam megadni, ott egy metódust használtam, ami csak számokat enged begépelni a textbox-ba.

Dolgozó adatbázishoz adása SQL injekcióval:

```
public static void Insert(string Nev, string telefonszám, DateTime
Szuletes, string gender, DateTime felvétel ,long adó_szá, long
tajszám, string mail, string munka, string cim)
```

```
{
    MySqlConnection con = new MySqlConnection(ConnectionString); var
    munkaID = Job_ID(munka);
    con.Open();
    MySqlCommand cmd = new MySqlCommand("INSERT INTO `employees`
    (`name`, `phone_number`, `birth_date`, `gender`, `hire_date`,
    `tax_number`, `taj_number`, `mail`, `jobs_Title_ID`,
    `address`) VALUES (@name, @phone_number, @birth_date, @gender,
    @hire_date, @tax_number, @taj_number, @mail, @jobs_Title_ID,
    @cim);", con); cmd.Parameters.AddWithValue("@name",
    Nev);
    cmd.Parameters.AddWithValue("@phone_number", telefonszám);
    cmd.Parameters.AddWithValue("@birth_date", Szuletes);
    cmd.Parameters.AddWithValue("@gender", gender);
    cmd.Parameters.AddWithValue("@hire_date", felvétel);
    cmd.Parameters.AddWithValue("@tax_number", adó_szá);
    cmd.Parameters.AddWithValue("@taj_number", tajszám);
    cmd.Parameters.AddWithValue("@mail", mail);
    cmd.Parameters.AddWithValue("@jobs_Title_ID", munkaID);
    cmd.Parameters.AddWithValue("@cim", cim);
    cmd.ExecuteNonQuery();
    InsertHR(Nev); con.Close();
}
```

Természetesen mielőtt feltöltenénk az adatbázisba az adatokat, előtte a program végez pár ellenőrzést, hogy elkerüljük az összeomlást.

2.7.2. Módosítás

A módosításokat egy előzetes adatbázis lekérdezéssel kezdi a program, majd azokat beilleszti az adott oldalra, amiket a felhasználó módosítani tud. A módosítani kívánt dolgozó adatait egy külön osztályból (DolgozoData.cs) olvassuk ki, ahol ezeket külön propertykbe kérdezhetünk le.

2.7.3. Törlés

A törlés egyszerűen egy metódusba írva SQL injekcióval oldottam meg, ahol a dolgozó ID-ja alapján töröl a program. A törléshez ki kell választani a datagridből a törölni kívánt személyt és a program utána lekérdezi az ID-ját és végrehajtja a metódust.

2.7.4. Keresés

Az oldalon továbbá elérhető egy kereső is, ahol munkakörökre és a nevekre van mód rá keresni. A munkákat egy SQL lekérdezéssel egy listába mentettem, amiket egy <ComboBox>ból érek el. A névre 'Textboxból' lehet keresni. A kereső gomb lenyomásával a program csinál egy egyedi SQL lekérdezést, ami az adott adatokra keres rá az adatbázisban és vetíti vissza a programnak.

SQL lekérdezés a kereséssel:

```
string sql = "SELECT e.emp_id, e.birth_date, e.name, e.phone_number,  
e.gender, e.hire_date, e.tax_number, e.taj_number, j.job_title, e.mail,  
e.address FROM employees e INNER JOIN jobs j ON j.job_id = e.jobs_Title_ID  
Where name LIKE CONCAT('%', @name, '%') AND j.job_title LIKE CONCAT('%',  
@job_title);";
```

2.8. Munkaórák

2.8.1. Dolgozik-e

A „Dolgozik-e” gomb arra szolgál, hogy le lehessen kérdezni, hogy a személyek dolgoznak vagy pihennek. A gomb 'click' eseménye egy lekérdezést hajt végre, amit a gomb neve alapján egy változó segítségével kérdez le. A Contentjét a gombnyomás után változtatja a program, amit az SQL lekérdezés során is használ változóként.

2.8.2. Csekkoló

A csekkológomb megnyomására felugrik egy ablak, ahova be tudja írni a felhasználó a dolgozó kódját és el tudja indítani, vagy lezárni az óraszámológóját. A gomb lenyomása után a program egy új felületet hoz létre, ahol a program megvizsgálja, hogy az adott személy dolgozik-e vagy épp pihen és aszerint indítja, vagy zárja a számlálóját. A program mind a két esetben feltölti az

értéket az adatbázisba és azt onnan is szedi le. Ha az adott illető befejezte a munkát, a program kiszámolja az aznapi óráit percekben megadva majd azt kerekítve, és felmenti a kapott értéket az adatbázisba.

A programkód a kezdeti időből vonja ki a kicsekkolt értéket és azt formázza string-é: string

```
Minutes = (myDateTime - from_working).TotalMinutes.ToString("f4");
```

Ha ezzel kész van, a stringet egy double-re konvertálva frissíti az adatbázisban a személy értékeit.

SQL adat frissítés, ahol a dolgozó ID-je megegyezik a kiválasztott személyével:

```
MySqlConnection con = new MySqlConnection(ConnectionString);
con.Open();
MySQLCommand cmd = new MySQLCommand("UPDATE workinghours SET from_date=@from,"
    "to_date=@to, minutes=@minutes, online=@online WHERE id_wkhrs=@ID", con);
cmd.Parameters.AddWithValue("@ID", ID);
cmd.Parameters.AddWithValue("@from", from);
cmd.Parameters.AddWithValue("@to", to);
cmd.Parameters.AddWithValue("@minutes", minutes);
cmd.Parameters.AddWithValue("@online", online);
cmd.ExecuteNonQuery();
con.Close();
```

2.8.3. Módosítás

A módosítás gombra kattintva egy új felület fog megjelenni, ahol át tudjuk írni a kiválasztott személy értékeit és tudjuk állítani, hogy épp dolgozik e vagy nem. Az új felületen a módosít gombra kattintva a program frissíti az adott személy értékeit, ha azokat mi módosítottuk.

2.8.4. Havi bérek

A havi bérek gombra kattintva egy új datagrid felület fog megjelenni, ahol a hónap végi zárásokat, illetve összesítéseket tudjuk megtekinteni. Itt a program nem számol semmit, csak lekérdezi az adatokat az adatbázisból.

2.8.5. Havi Zárás

A havi zárást a kiválasztott személyt lezárja az adott hónapban. Itt a program lenullázza a személy óráit és kicsekkolja őt/őket. A havi zárás veszi az összes percet, azt elosztja 60-nal, és beszorozza a személy munkakörének órabérével. Ha kész a számolásokkal, utána ezeket feltölti az adott hó végi dátummal, amit ha szeretnénk vissza tudunk nézni bármikor.

A személy perceit órává alakítja:

```
double hours = (double)data.minutes / 60;
```

Felszorozza az órákat a munkakör órabérével:

```
double monthlySalary = data.Hour_pay * hours;
```

Ezt a funkciót csak akkor lehet elérni, ha az adott hónap végén járunk és nem lehet lezárni a béreket idő előtt.

2.9. Naptár

A naptár menüpont alatt a cég kiadásai és bevételei találhatóak. Itt megadhatunk napi bevételt és napi kiadást, amit a program kiszámol és különböző változókba rak. A napokat az éppen választott napból veszi és kérdezi le.

```
private void DateChanger(object sender, SelectionChangedEventArgs e)
{
    if (calendar123.SelectedDate.HasValue)
    {
        time = calendar123.SelectedDate.Value;
        current = DataManager.finance(time);
        if (current.Count == 0)
        {
            btnCheck();
            tb1.Text = "Üres az adatbázis!";
            tb2.Text = "Üres az adatbázis!";
            tb3.Text = "Üres az adatbázis!";
            tb4.Text = "Üres az adatbázis!";
            tb6.Text = "Üres az adatbázis!";
        }
        else
        {
            refresher();
            btnCheck();
            calculator();
        }
    }
}
```

Az oldalon megtalálható még a napi váltó, amit a nap elején a kasszásnak át kell számolni, ez az az összeg, amit este a záró kasszás ott hagyott, ennek az összegnek mindig ugyanannyinak kell lennie. A havi bevétel, ami a hónap minden napját összevonja és kiszámolja az eddigi havi nettó bevételét a cégnek, ezen felül még kiszámolja a napi nettó bevételt is. A menüpont „szíve” a naptár, ami változás esetén ad vissza értéket és keres rá az adatbázisból. Itt visszamenőleg lehet keresni, akár hónapra vagy évre. Az adatbázis az adott hónapokat nem tárolja, így ha a kijelölt napon nincsen érték azt generálni kell. A program az adott hónapot tölti fel értékekkel ezzel, hogy csak utólag töltjük fel az adatbázis helyet és időt nyer a programnak, mert nem kell végig iterálgatni az összes adatbázis értéken. Az adatbázisba az adott hónap végét adja meg kezdő értékek. A napi váltót a program a fájlok közül olvassa ki, és adja be értéknek az adatbázisba. A naptár bármelyik értékét választja ki a felhasználó az adott napot mindig bekarikázza, de a „ma” gombra nyomva is visszaugrik.

2.9.1. „Ma” gomb

A gomb megnyomására a program a naptáron kijelöli az adott napi dátumot, ezt egy sötétszürke körbe jelzi a felhasználónak. Természetesen, az adatokat is átállítja az adott értékre.

2.9.2. Alap érték

Az alapérték gomb megnyomásával a program feltölti az adatbázist az adott hónap napjaival.

A napok az adott hónapban egy beépített függvény segítségével tudtam lekérni, ehhez a függvényhez szükségem volt egy változóra, amiben a kiválasztott dátumot tartom:

```
time = calendar123.SelectedDate.Value;
current = DataManager.financy(time);
```

Ezután megnéztem, hogy hány napos az adott hónap:

```
calendar123.SelectedDate = time;
current = DataManager.financy(time);
days = DateTime.DaysInMonth(time.Year, time.Month);
```

Ezek után egy for ciklussal végig iterálgatok az összes napon és feltöltöm azokat alap adatokkal:

```

for (int i = 1; i <= days; i++)
{
    string add = $"{time.Year}-{time.Month}-{i}";
    DateTime counter = DateTime.Parse(add);
    DataManager.basicValues(counter, DailyChange, 0, 0, 0, 0, 0);
}

```

2.9.3. Napi bevétel-kiadás „bevitel” gomb

Ezekkel a gombokkal tudjuk állítani a választott értéket. Ha megnyomjuk a gombot, azok contentje megváltozik „OK” -ra majd, ha megadtuk az értéket a mellette lévő textbox-ba a program hozzáadja az adott értéket az adatbázisban lévő számhoz. Természetesen itt védelem szempontjából csak számokat lehet megadni, hogy a programot ne lehessen kifagyasztani.

```

private void btn2_Click(object sender, RoutedEventArgs e)
{
    if (btn2.Content.ToString() == "bevitel")
    {
        tb2.Visibility = Visibility.Hidden;
        tbw2.Visibility = Visibility.Visible;
        btn2.Content = "OK";
    }
    else
    {
        btn2.Content = "bevitel";
        tb2.Visibility = Visibility.Visible;
        tbw2.Visibility = Visibility.Hidden;
        if (tbw2.Text == "")
            dIncome += 0;
        else
            dIncome += double.Parse(tbw2.Text);

        DataManager.moneyDebit(dIncome, dOutcome, mIncome, mOutcome, netIncome, time);
        calculator();
        refresher();
        tb2.Text = dIncome.ToString();
        tbw2.Clear();
    }
}

```

2.10. Kapcsolat

A ‘Kapcsolat’ menüpont alatt találjuk meg a dolgozók elérhetőségeit, (lásd; telefonszám, email, lakcím). Ezeket az adatokat a program egy datagridbe tölti fel. A keresés a

Datamanager.Dolgozólistáka függvényt használja, amit a dolgozóknál is használtunk. Itt viszont az adatok egy részét használjuk csak fel.

2.11. Beállítás

A beállítás menüpontban tudjuk megnézni a felhasználó nevét és hogy milyen jogosultsága van. Ezeket az adatokat egy változóban tároljuk, amit a program a bejelentkezés után tölt fel még az elején. Napi váltót tudjuk módosítani, amit a program egy txt fájlban tart a forrásfájlok között. Ezt a fájlt, ha nem találja a program egy újat generál egy alap értékkel.

2.11.1. Felhasználók módosítása

Jelszavunkat meg tudjuk változtatni a 'jelszó módosítás' gombra kattintva. Miután megnyomtuk a gombot a felhasználó törlése felület eltűnik és <stackpanel> jelenik meg a helyén, ahova az adott adatokat kell bevinni. Természetesen ezeknek az adatoknak van ellenőrzése is. Például a program megnézi, hogy a profil, abibe be vagy jelentkezve tényleg jól adtad-e meg a régi jelszavad vagy az új jelszó elég erős-e, mondjuk vannak e benne számok, ugyanaz e a két új jelszó.

```
private string AzEllenőr()
{
    if (string.IsNullOrEmpty(passNow.Text) || string.IsNullOrEmpty(newPass.Password) || string.IsNullOrEmpty(newPassAgain.Password))
    {
        return "Kötelező megadni adatokat!";
    }
    if (newPass.Password != newPassAgain.Password)
    {
        return "Nem egyeznek meg a jelszavak!";
    }
    if (!newPass.Password.Any(char.IsDigit))
    {
        return "Jelszó gyenge, tegyen bele számokat!";
    }
    if (newPass.Password.Length < 4)
    {
        return "Nem elég hosszú a jelszó";
    }
    if (NewBool && !Email.Text.Contains('@') || !Email.Text.Contains('.'))
    {
        return "Nem adta meg helyesen az email címét!";
    }
    return "";
}
```

Új felhasználó bevitel esetén az eljárás majdnem ugyan az, itt meg kell adnod egy felhasználónevet, ami még nincs az adatbázisban (természetesen ezt a program ellenőrzi), a jelszó megadása ugyan úgy ellenőrizve van és a végén meg kell még adni egy e-mail címet, ahol a program ellenőrzi, hogy a „.” és a „@” benne van-e a megadott értékben.

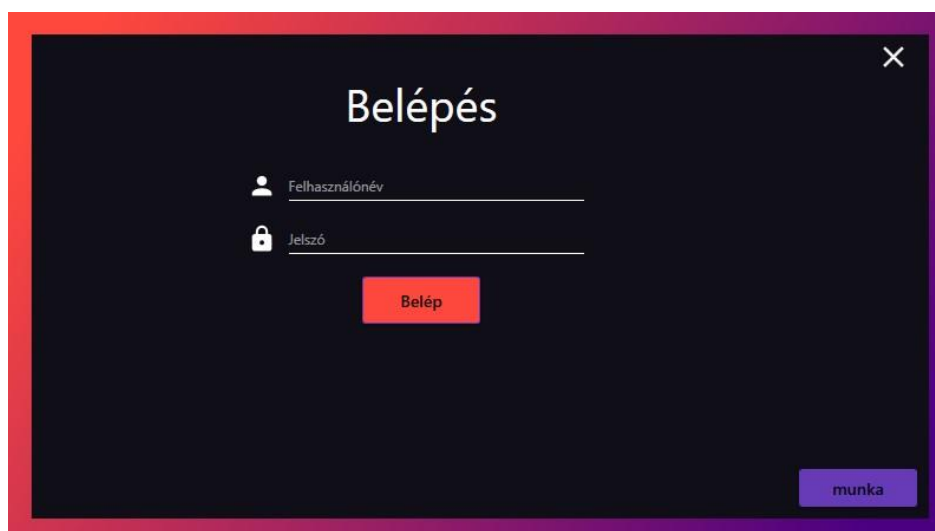
Felhasználó törlése esetén a program egy <combobox>-ba tárolja el a felhasználókat, ahonnan ki tudja választani a felhasználó, hogy melyiket szeretné törölni. Természetesen az adott felhasználót, amibe be van jelentkezve nem tudja kitörölni, de az össze többit igen.

Ezt egy for ciklussal feltöltjük egyesével alap adatokkal feltöltve:

```
private void newUser_Click(object sender, RoutedEventArgs e)
{
    if (newUser.Content.ToString() == "Új felhasználó")
    {
        newUser.Content = "Ok";
        passModify.Content = "Mégsem";
        cbUsers.Visibility = Visibility.Collapsed;
        userDelete.Visibility = Visibility.Collapsed;
        texts.Visibility = Visibility.Visible;
        Email.Visibility = Visibility.Visible;
        MaterialDesignThemes.Wpf.HintAssist.SetHint(passNow, "Felhasználó név:");
        MaterialDesignThemes.Wpf.HintAssist.SetHint(newPass, "Jelszó:");
        MaterialDesignThemes.Wpf.HintAssist.SetHint(newPassAgain, "Jelszó újra:");
        MaterialDesignThemes.Wpf.HintAssist.SetHint(Email, "Email:");
    }
}
```

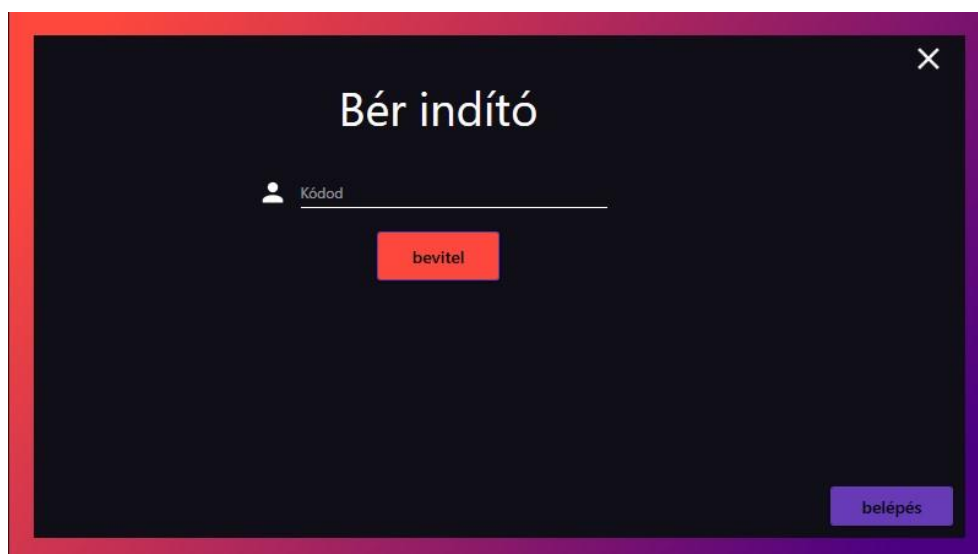
3. FELHASZNÁLÓI DOKUMENTÁCIÓ

3.1. Bejelentkező felület



A program elindítása után, ez az ablak fog megjelenni a felhasználó előtt. Itt több opciónk is van, hogy mit szeretnénk csinálni. Beléphetünk a felhasználói fiókunkba, amit úgy tehetünk meg ha megadjuk a helyes felhasználónevet és jelszót, majd megnyomjuk a belépés gombot.

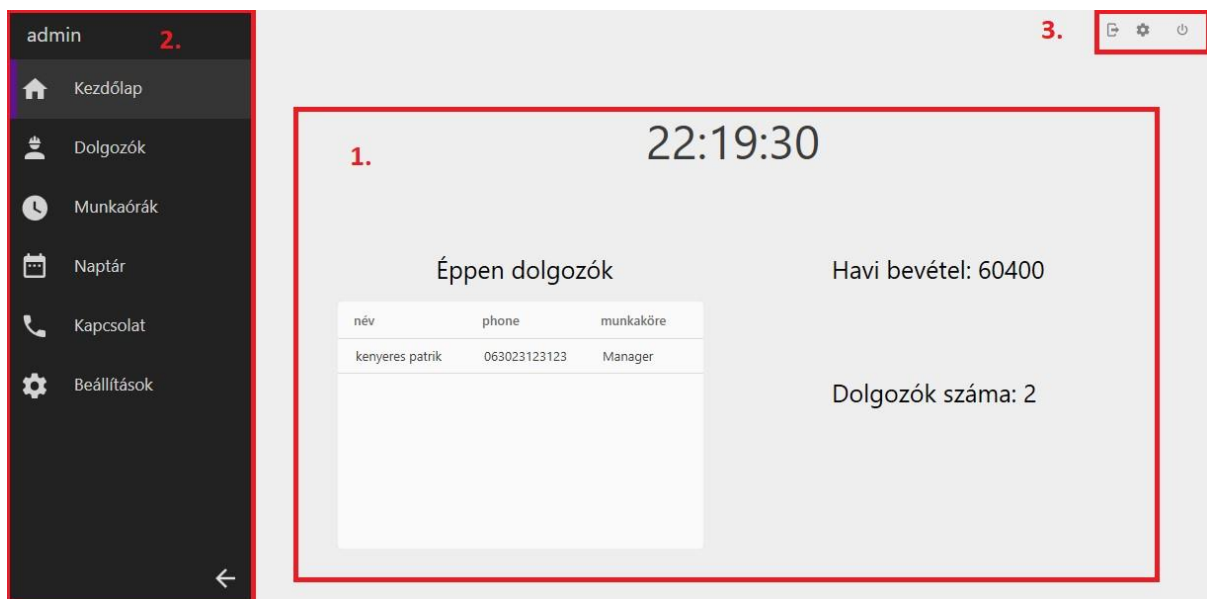
Másik lehetőség pedig, hogy ha megnyomjuk a 'Munka' gombot, ezután egy új felület fog megjelenni:



Ide a dolgozónak generált kódot kell megadni. Ezt a kódot a rá jogosult ember tudja megtekinteni és közölni az adott dolgozóval. Ezeket a kódokat belépés után a munkaóra menüpontban tudja megtekinteni az arra jogosult személy. Ha a kód helyes, a program kiírja, hogy a dolgozónak mikortól vagy meddig van/volt a munkaideje.

3.2. Kezdőlap

A kezdőlapon a felhasználó kap egy pontos órát, hogy éppen kik dolgoznak, a havi nettó bérét a cégnek és az összes dolgozó számát.





A menüben megtalálható a felhasználó neve, amivel bejelentkezett az alkalmazásba. A felhasználónév alatt láthatóak a menüpontok, mindegyik külön funkcióval az adott oldalra irányítja a felhasználót. A menük alatt egy nyíl látható, amit ha megnyomunk összezsugorodik a menü és csak az ikonok láthatóak.

A shortcut panel, ami lehetővé teszi, hogy a felhasználó ki tudjon jelentkezni a felhasználójából, ezt a kis ajtó jelre kattintva tud megtenni. A középső fogaskerék ikon a beállításokhoz vezeti a felhasználót. A power gomb pedig a program futását állítja meg, ezzel a program teljesen bezáródik.

3.3. Dolgozók

A dolgozók menüpontba a felhasználó meg tudja nézni a dolgozók adatait, és rá tud keresni adott dolgozókra, munkakör és név szerint.

A 'felvétel gombra' nyomva fel tudja venni a felhasználó a dolgozókat.

Név:	<input type="text"/>
születési év:	<input type="text"/> 
telefonszám:	<input type="text"/>
Neme:	<input type="text"/> ▼
felvétel napja:	<input type="text"/> 
Adószám:	<input type="text"/>
Taj szám:	<input type="text"/>
Email:	<input type="text"/>
Munkaköre:	<input type="text"/> ▼
Lakcíme	<input type="text"/>
<div>Mégsem Felvesz</div>	

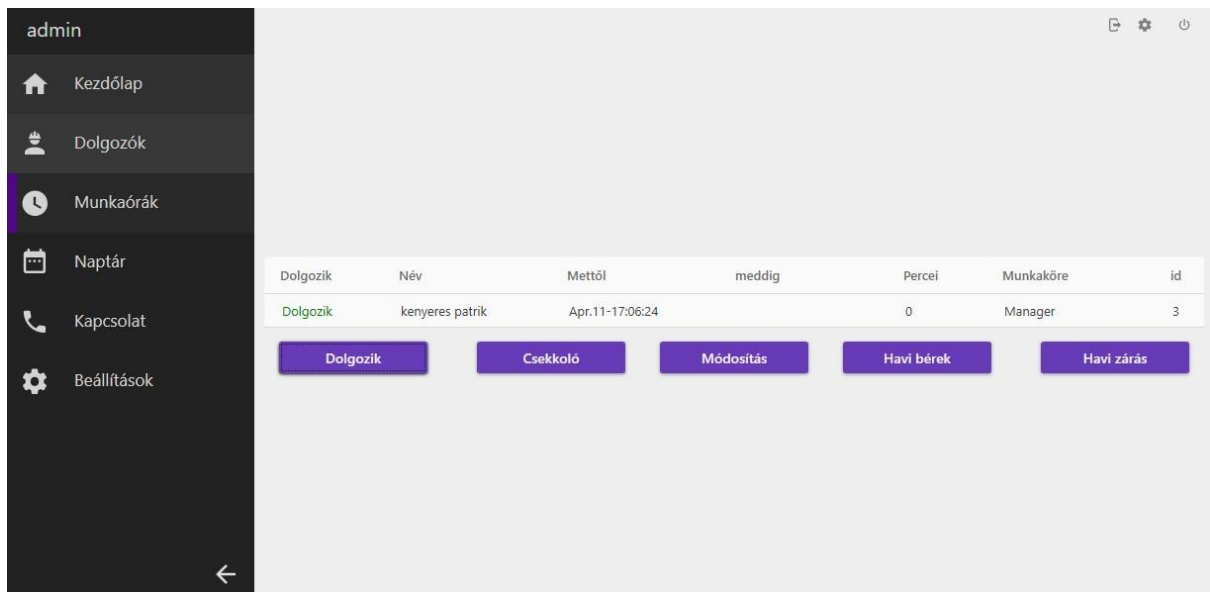
A felvevő felületen, ha minden adat megfelelő a „felvesz” gombbal hozzá tudja adni a dolgozók listához az illetőt.

A módosítás gomb megnyomása előtt ki kell jelölni azt adott munkavállalót, akinek az adatait módosítani szeretnénk és a felugró ablakon átírhatjuk az adatait, majd a felvesz gombbal tudja az adatokat módosítani.

A törlés gomb megnyomása előtt ki kell jelölnünk az adott illetőt, akit törölni szeretnénk a rendszerből, ezt egy megerősítő ablak követi, amit ha le igenezünk kitörli a személyt.

3.4. Munkaórák

A munkaórák menüpontban meg lehet nézni, hogy ki dolgozik és ki nem. Ezt a funkciót a 'dolgozik/pihen' gombbal tudja megtenni. A gomb megnyomása után ez átíródik.



A csekkoló gomb a dolgozót tudja be-ki jelenteni. Itt is, mint a bejelentkező felületen a dolgozó id-jét kell megadni.

A módosítás gomb megnyomása előtt ki kell jelölni a módosítani kívánt személyt, majd a felugró ablakba át lehet írni az értékeit.

Havi bérek gomb megnyomásával a bérjegyzéket lehet megnyitni, hónapokra osztva.

Havi zárás gombbal le tudja zárni a kiválasztott személy bérét, de ezt csak akkor tudja megtenni, ha az adott hónap végénél járunk, előbb nem lehet lezárni.

3.5. Naptár

The screenshot shows the 'Naptár' (Calendar) interface in the 'admin' section. The sidebar on the left contains the following menu items: Kezdőlap, Dolgozók, Munkaórák, Naptár (selected), Kapcsolat, and Beállítások. The main content area displays financial data for the current day (April 8, 2021). The data is organized as follows:

- Napi váltó: 40000
- Napi bevétel: 57400 (with a 'bevitel' button next to it)
- Napi kiadás: 2000 (with a 'bevitel' button next to it)
- Havi nettó bevétel: 60400
- Napi nettó bevétel: 55400

At the top right, there are buttons for 'Alap érték' and 'Ma'. On the right side, there is a calendar widget for April 2021, showing the current date as 'Thu, Apr 8'.

A „Naptár” menüpontba a cég bevételét és kiadásait lehet megtekinteni és módosítani. A napi váltó a kasszában lévő pénz tartja számon, ezt tudja változtatni a beállítás menüpontban. A 'bevitel' gomb megnyomása után a kijelölt helyre be kell írni a kívánt összeget/összegeket. A havi nettó bevétel az aktuális hónap nettó, azaz a kiadások levonása utáni összeget adja meg. A napi nettó bevétel az aznapi bevétel nettó értékét adja vissza. A „ma” gomb megnyomására a naptár a mai dátumra ugrik. Az „alap érték” gombbal a hónap minden napjának adunk egy alap értéket.

3.6. Kapcsolatok

admin

Kezdőlap
Dolgozók
Munkaórák
Naptár
Kapcsolat
Beállítások

Név: _____
Munkakör: _____

KERESÉS

Név	telefonszám	email	lakcím
Kenyi	06204787262	kenyeres@gmail.com	asd út 32
kenyeres patrik	063023123123	asd@gmail.com	ibolya utca 3/a

A kapcsolatok menüpontba a dolgozók elérhetőségeit, lakcímét lehet megtekinteni. Itt is, mint a dolgozók menüpontnál lehet keresni adott munkakörre és a dolgozónak a nevére is. Ezután a keresés gomb lenyomásával a program megkeresi a kívánt személyt/személyeket.

3.7. Beállítások

A 'beállítás' menüpontot több helyről is el tudja érni. Egyszer elérhető a jobb fenti kis gyorsgomb tárból, másodszor pedig, ha a menüpontból kiválasztjuk a beállítások menüpontot.

admin

Kezdőlap
Dolgozók
Munkaórák
Naptár
Kapcsolat
Beállítások

Felhasználóneve: admin
Jogosultsága: Admin
Napi váltó 40000

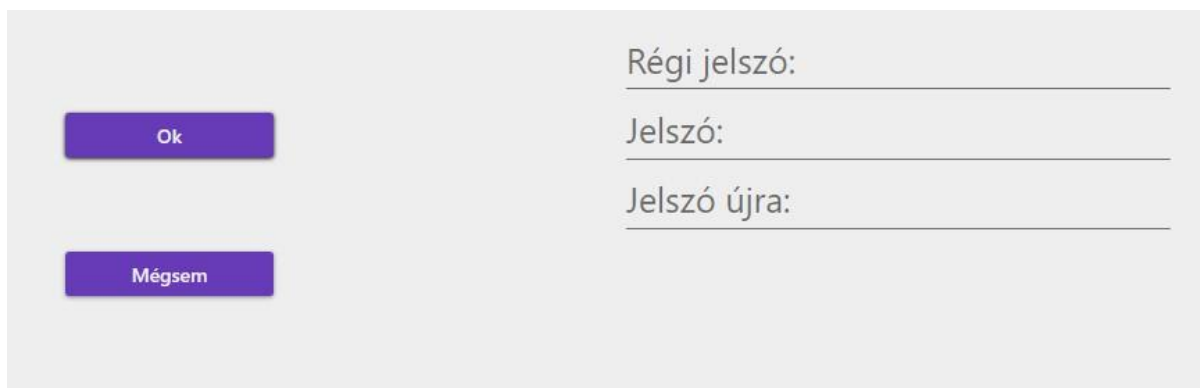
Módosít

Jelszó módosít

Új felhasználó

Felhasználó törlés

A menüpont alatt meg tudja változtatni a jelszavát, tud új felhasználót hozzáadni a rendszerhez vagy tud törölni felhasználót onnan. A jelszavakat, ha kiválasztjuk az egyik funkciót el fog tűnni a felhasználó törlése felület és egy új adatbevitelhez készített beviteli mező lesz a helyén.



The screenshot shows a light gray dialog box for changing a password. On the left side, there are two purple buttons: 'Ok' at the top and 'Mégsem' (Cancel) at the bottom. On the right side, there are three text input fields with labels: 'Régi jelszó:' (Old password), 'Jelszó:' (Password), and 'Jelszó újra:' (Password again). Each label is followed by a horizontal line representing the input field.

A jelszó módosítása gomb megnyomásával hasonló dolog fog történni, mint az új felhasználó létrehozásánál. A jobb oldalon lévő felületre be kell írni az előző jelszavát és az új választott jelszavát, amire meg szeretné változtatni az előzőt. Viszont, ha az új felhasználó funkciót választja, akkor meg kell adni a kívánt felhasználónevet, arra figyelni kell, hogy olyan felhasználónevet ne adjon meg amit már a rendszer tartalmaz, de ha ilyen előfordul a program szólni fog. A jelszavak megadásánál figyeljen, hogy legyen benne nagybetű és szám és persze ne legyen túl rövid.

A felhasználó törlésénél válassza ki a legördülő listából melyik felhasználót szeretné kitörölni és a program eldönti, hogy a kiválasztott értéket lehet-e törölni vagy esetleg valamilyen feltételnek nem felel meg.

4. ÖSSZEFOGLALÁS

Próbáltam arra figyelni, hogy egy jól használható programot írjak, aminek van értelme és célja is. Mégpedig az, hogy a munkáltatónak kevesebb feladata legyen a mindennapi életben, és a tárolandó adatok, megtalálhatóak legyenek egy helyen, és ráadásul összesítve is van. Ha én, mint munkáltató kapnám 'kézhez' ezt a programot, minden elfogultság nélkül, meg lennék elégedve a végeredménnyel.

Fontosnak tartottam, hogy a program továbbfejleszthető legyen, bár a kigondolt koncepciót nagyrészt siketült véghez vinnem, viszont nem minden cégnek ugyanaz az elvárása egy ilyen dokumentáló program során. Ezzel tisztában voltam és úgy is alakítottam ki, hogy ez könnyen fejleszthető legyen. Amit kigondoltam emellé, esetleg egy online felületet létre lehetne hozni a dolgozók részére, így azok meg tudják nézni, mennyit dolgoztak az adott hónapban, emellett lehetőség lenne rá, hogy az aktuális fizetésüket is nyomon kövessék. A dizájn során figyeltem arra, hogy a mai kornak megfelelő legyen és próbáltam a színekkel játszodozni, nem hétköznapi, de mégis jól átlátható felületeket létrehozni. Úgy gondolom sikerült létrehoznom egy modern, a kornak megfelelő programot.

A tesztelést többféle módon is megtettem. Más számítógépeket is igénybe vettem ehhez, hogy kiderüljön működik-e minden. Először az adatbázist töröltem és megnéztem mit tesz a program, ha nem találja. Ezután átneveztem az adatbázist és regeneráltattam újra a programmal. Kipróbáltam más gépen is futtatni, szerencsére ott is sikerült. Munkám során, igyekeztem minden helyre rossz adatot adni, hogy esetleg valahol kifagy-e a program, ha nem jó típusú adatot adok be. A programot próbáltam minél több hibalehetőségnek kitenni. Próbálgattam kifagyasztani gombokkal, itt végül találtam is egy hibát, amit úgy tudtam kihozni, hogy ha pozícionálni szerettem volna az ablakot és bal klikk helyett, jobbal húztam oda, vagy csak rányomtam, a program kifagyott. Valamit találok többféle típuskonverziós hibával. Sajnos rájöttem, hogy ha nullával kezdődő adatot adok meg valahova, és azt az adatbázisban számként mentem, azt kitörli a sor elejéről. Ezért több helyen inkább string típusként mentettem el az adatokat és utána konvertáltam át őket számmá.

Ha valamit másként kéne csinálnom az az SQL injekció lenne. Szerintem eléggé amatőr munka ennyi SQL injekcióval dolgozni, de sajnos még nem volt meg a kellő tudásom Entity Frameworkkel dolgozni. Ezt a jövőben orvosolni fogom, hogy az elvárásaimnak teljesen eleget tegyek. Végző soron még a bevétel funkciót csináltam volna egy kicsit máshogy. Utólag rájöttem, hogy sokkal jobb lenne, ha tudnánk valami kis megjegyzést is tenni a bevétel/kiadás mellé, hogy a felhasználó tudja, hogy mikor mire költött vagy épp honnan van a bevétel.

5. IRODALOMJEGYZÉK

Illés Zoltán – Programozás C# nyelven

Jedlik Oktatási Stúdió Bt.

XAMPP 2021.03.5.

https://www.inf.u-szeged.hu/~gnemeth/adatbgyak/exe/MySQL_XAMPP_JDBC/a_xampp_keretrendszer_s_hasznalata.html

Microsoft Visual Studio 2021.03.5.

https://hu.wikipedia.org/wiki/Microsoft_Visual_Studio

phpMyAdmin 2021.03.5.

https://www.inf.u-szeged.hu/~gnemeth/adatbgyak/exe/MySQL_XAMPP_JDBC/phpmyadmin_a_mysql_kezelshez.html

<https://wordpress.org/support/article/phpmyadmin/>

dbForge Studio 2021.03.5.

<https://www.devarit.com/dbforge/mysql/studio/>