Vasvári Pál Gazdasági és Informatikai Szakgimnázium

54 213 05 számú Szoftverfejlesztő szakképesítés

záródolgozata

|  |
| --- |
|  |
| Záródolgozat |
| fogorvosi adminisztrációs rendszer |

|  |
| --- |
| Készítette:  Kovács Ferenc 2/14.DE |
| Szeged  2019 |

Tartalom

[2. Bevezetés 2](#_Toc3991963)

[2.1 Használt programok, felületek és egyéb kiegészítők 3](#_Toc3991964)

[3. Az alkalmazás felépítse 4](#_Toc3991965)

[3.1 Követelmények 4](#_Toc3991966)

[3.2 Az Adatbázis 5](#_Toc3991967)

[3.2.1 Adatbázis felépítése 6](#_Toc3991968)

[3.3 A program felépítése 8](#_Toc3991969)

[3.3.1 FogOrvosiRendelő (WinForm alkalmazás) 8](#_Toc3991970)

[3.3.2 View Réteg 9](#_Toc3991971)

[3.3.2.1 Bejelentkezés (View) 9](#_Toc3991972)

[3.3.2.2 Fő ablak (View) 11](#_Toc3991973)

[3.3.2.3 Új páciens felvétele (View) 13](#_Toc3991974)

[3.3.2.4 Páciens részletes adatai (View) 15](#_Toc3991975)

[3.3.2.5 Páciens kórtörténete / új bejegyzés (View) 16](#_Toc3991976)

[3.3.3 Controller réteg 17](#_Toc3991977)

[3.3.4 Model réteg 17](#_Toc3991978)

[3.3.4.1 LoginModel 17](#_Toc3991979)

[3.3.4.2 MainModel 18](#_Toc3991980)

[3.3.4.3 PatientModel 18](#_Toc3991981)

[3.3.5 WebFogRendelo (Webes alkalmazás) 19](#_Toc3991982)

[3.3.5.1 Index.php (kezdőlap) 19](#_Toc3991983)

[3.3.5.2 Php almappa 21](#_Toc3991984)

[3.4 Üzembe helyezés tesztelés 24](#_Toc3991985)

[4. Felhasználói dokumentáció 24](#_Toc3991986)

[4.1 Ajánlott hardver-, szoftver követelmények 24](#_Toc3991987)

[4.2 A program használata 25](#_Toc3991988)

[4.2.1 Asztali alkalmazás (a fogorvos részére) 25](#_Toc3991989)

[4.2.1.1 Bejelentkezés 25](#_Toc3991990)

[4.2.1.2 Főablak 25](#_Toc3991991)

[4.2.1.3 Páciensek felvétele 26](#_Toc3991992)

[4.2.1.4 Páciens adatainak módosítása 27](#_Toc3991993)

[4.2.1.5 Páciens törlése 28](#_Toc3991994)

[4.2.1.6 Páciens bejegyzései / Kórtörténet 28](#_Toc3991995)

[4.2.2 Webes alkalmazás (a páciensek részére) 29](#_Toc3991996)

[4.2.2.1 Menü / Belépés 29](#_Toc3991997)

[4.2.2.2 Adatok megtekintése 30](#_Toc3991998)

[4.2.2.3 Kórtörténet / Bejegyzések megtekintése 30](#_Toc3991999)

[4.2.2.4 Belépési Adatok módosítása 31](#_Toc3992000)

[5. Összefoglaló 31](#_Toc3992001)

[6. Mellékletek 32](#_Toc3992002)

# Bevezetés

A szakdolgozat témáját az alapján választottam, hogy a tanulmányaim alatt megismert C#, MySQL, PHP és HTML ismereteimet egyaránt betudjam mutatni és tudjak kísérletezni velük, gyakorolni azokat. Ezen ismeretek még bőven fejleszthetőek és fejlesztendőek is, de úgy hiszem mire végzem a szakdolgozat elkészítésével rengeteg új és hasznos dolgot tanulok meg. Egy adminisztráció és előjegyző rendszert készítek fogorvosi rendelők számára. Egy C#-ra épülő asztali alkalmazás, ami a fogorvos számára készül, egy PHP alapú webes alkalmazás a páciensek számára és egy a MySQL adatbázis amelyből az előző kettő dolgozik. Mivel a párom fogorvoslást tanul és abban is dolgozik egy valós problémából indultam ki.

Egy rendelő adminisztrációját, nyilvántartását jelenleg többnyire papíron és egy integrált rendszerben központi rendszerben vezetik, ez utóbbi tartalmazza a legszükségesebb adatokat (név, születési adatok, társadalombiztosítási azonosítót és annak érvényességét). A papír alapú fogorvosi kartoték már tartalmazza a beteg esetleges kórtörténetét és egyéb információkat.

Az általam készített komplex alkalmazás ezt igyekszik egy átláthatóbb felületre vezetni úgy, hogy ezen adatokat a fogorvos könnyedén kiegészíthesse/módosíthassa, hogy a páciensnek otthonról legyen lehetősége megtekinteni a saját adatait, kórtörténetét. A három részből összeálló alkalmazás a következőkből áll össze. A leglényegibb része az adatbázis, ami tartalmazza a páciensek összes adatát és a szükséges információkat, mind a doktor mind a páciensek hozzáférési jogosultságát. Ezzel fog kommunikálni a webes és az asztali alkalmazás is. Az asztali alkalmazás, amin keresztül átláthatóan megtudja a fogorvos tekinteni az összes adatok és képes lesz azokat módosítani, létrehozni és ha szükséges törölni, képes ezen keresztül hozza létre a kórtörténetet azaz a látogatások dokumentációját. És végül a másik végpont a webes alkalmazás, itt lesz lehetőségük a pácienseknek (a rendelőben történő regisztrációt követően) megtekinteni a saját adataikat és ha szükséges javítani is egy részüket. Továbbá ezen a felületen tekintheti meg a saját kórtörténetüket.

## Használt programok, felületek és egyéb kiegészítők

A fejlesztői munka közben a tanulmányaim és otthoni szárnypróbálgatásaim alatt tanult és jelenleg is tanulás alatt lévő programokat fogom alkalmazni. Az adatbázist egy XAMPP v3.2.2 alatt futó Apache/2.4.38, MySQL/4.8.5 phpMyAdmin felület alatt fejlesztettem és teszteltem. Az egyszerű webes felülete, konzolosan visszatekinthető tartalma miatt hamar megbarátkoztam vele és nagy segítségemre volt az SQL nyelv alsajátításában. A program C# nyelven írt fogorvosoknak szánt részéthez a legelterjedtebb és legismertebb IDE-jét (**I**ntegrated **D**evelopment **E**nvironment) magyarul fejlesztői környezetét használom, a Visual Studio 2017-es Community kiadását a (jelenleg) 15.9.4-es verziót. A nyelvet ez alatt a felület alatt tanutam és használtam végig a tanulmányaim alatt, több apró funkcióját megismertem de méretéből adódóan úgy hiszem még mindig tartogat meglepetéseket számomra. Az IDE-hez mint Reference szükségem volt az adatbázis elérése végett a MySql Connector/NET-re. Ez végzi a közvetlen kommunikációt az adatbázis és a C# program között. Illetve a CryptSharp 2.1.0-ás verziójú programra ez végzi a jelszavak kódolását és ellenörzését, a választás azért rá esett mert több kriptográfiai algoritmust is ismer, többek között a később webes felületen használt „Blowfish” -t is.

A project másik felén futó webes alkalmazást PHP nyelven fog íródni, a megjelenést elősegítő HTML és CSS kódokkal és egy kis JavaScript-el utóbbin belül jQuery-könyvtárral dolgozva. Ennek az elkészítésében legfőbb támaszom a NetBeans IDE 8.2-es verziója volt. Korábbi Java-s ismereteim alatt már megbarátkoztam ezzel a programmal és kifejezetten meg is kedveltem a sokszínűsége miatt. Sok hasznos a hatékony programozást elősegítő eleme van. ezen végeztem a komplett webes felület elkészítését. Természetesen a folyamatos tesztelés és kísérletezés végett szükségem volt a webböngészőkre is ezeket változékonyan felváltva használtam (többnyire Google Chrome és Firefox Mozilla, de néha megnyilt az Edge és az IE11 is), úgy gondolom ha az ember webes felületre készít programot szükséges a legelterjedtebb felületeket mind végig próbálni az esetleges hibák kikerülése végett. A PHP esetében nem volt szükség külön kiegészítőre az adatbázissal történő kommunikációhoz az a nyelvbe már előre implementálva van, mint ahogy a fent említett „Blowfish” algoritmust is ismeri.

# Az alkalmazás felépítse

Az alkalmazás megvalósítása több részből kell hogy álljon:

1. Követelmények meghatározása, információ gyűjtés a felhasználóktól (esetdiagrammok), hogy mire van igény, mit szeretnének a programon belül látni, csinálni.
2. Felhasználói felület felépítése. Egyszerű és logikus vázlata a felhasználói felületeknek. Az igényeket figyelembevéve.
3. Az programhoz szükséges adatbázis/adatbázisok megtervezése és kivitelezése.
4. A program elkészítése az előre meghatározott nyelveken a használni kívánt környezetekhez.
5. Az alkalmazás üzembehelyezése, tesztelése, esetleges javítása.

## Követelmények

Az alkalmazásnak az alábbi követelményeknek kell majd megfelelnie. Mivel két külön oldalról (webes és asztali alkalmazás) két külön rétegnek (a rendelő kezelő személyzete és a paciensek) kell egy részben közös adatbázist használni, így MySQL adatbázisban fogom tárolni az adataim nagy részét. Így mint két fél egyidejűleg képes lesz használni az alkalmazást.

Biztonsági szempontból mindkét oldalon jelszavas védelmet alkalmazunk. A hozzáférés jogosultságának ellenőrzését követően mindenki a jogkörének megfelelően képesnek kell lenni adatokat megjeleníteni, létrehozni, törölni vagy esetleg módosítani. Egy egyszerű belső rendszer segítségével képesek lesznek üzenetet váltani egymással. Ezen felül rendelkezniük kell naptárral amin az asztali fogorvosi oldalon minden lefixált és igényelt előjegyzés látható lesz illetve a webes (páciens) oldalán a saját időpontjai megtekinthetőek lesznek.

Az 1. Kép az eset diagrammot tartalmazza szerepkörök és azok feladatai szerint.

. Kép



## Az Adatbázis

A relációs adatmodell egy olyan adatmodell, amelynek legfontosabb eleme a matematikai reláció fogalma.[[1]](#footnote-1) A reláció adatbázis legnagyobb előnye számunkra hogy a csak az éppen szükséges információkat keressük/jelenítjük meg a komplex adatbázis megtekintése és programon belül történő végig járása nélkül. A tanulmányaim alatt tanult MySQL-t mint logikus felépítésű és véleményem szerinti egyszerű kezelésű relációs adatbázis-kezelőt fogok használni.

A programhoz több különböző táblából fog összeállni az adatbázis, hogy elkerüljük a jogosulatlan hozzáféréseket, illetve módisításokat (pl. a páciensnek a saját belépéshez szükséges nevét és jelszavát jogában áll módisítani, személyes adatait nem).

### Adatbázis felépítése

Mivel az adatbázishoz nem lenne szerencsés ha bármi illetéktelen személy hozzáférhetne, a legfontosabb az magának a fogorvosnak, sőt akár több fogorvosnak, illetve az asszisztensek hozzáférése.

Így szükség van egy táblára az orvosi személyzet bejelentkezési adataival:

1. **Tábla:** *staff*
2. id
3. username
4. password
5. name

Az id egy automatikusan nullától induló inkrementált (auto increment[[2]](#footnote-2)) szám (az első elem ebből kifolyólag az 1) nem szerkeszthető, a username magyarul felhasználóinév egy unique azaz egyedi rekord, nem létezhet két egyforma. A password az orvos jelszavát tartalmazza. A name meg magának a fogorvosnak illetve az asszisztensnek neve.

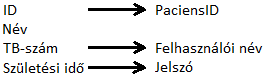
A következő fontos réteg az maguk a páciensek és azoknak minden adata, amikre szükség lehet a későbbiekben. Itt úgy gondolom a legkézenfekvőbb ha négy külön táblában dolgozom aszerint hogy kinek mihez lesz jogköre látni illetve szerkeszteni az adott rekordokat. Az első ezekből a páciens legszükségesebb adatai ami alapján meg lehet őket különböztetni

**2. Tábla:** *patient*

1. id
2. name
3. birthdate
4. tb

Az id megint csak egy auto increment szám, a TB-szám viszont unique mivel két külön személy nem rendelkezhet egy társadalombiztosítási. A name a páciens nevét, míg a birthdate a születési idejét tartalmazza. Ezen adatokat csakis az orvosi személyzet módosíthatja.

Második a webes felülethez történő hozzáférést biztosító adatok tárolására szolgáló tábla.

**3. Tábla:** *patient\_web*

1. patient id
2. username
3. password

A páciens id az előző táblából örökölt id lesz. A társadalombiztosítási azonosítóból jön létre a username (felhasználói név) és a születési idő adja magát a kezdeti password-öt (jelszavat). Ezek alapértelemezett létre fognak jönni az előző táblázatból, a fent említett formában.

Természetesen ezeket az adatokat a webes felületre történő első bejelentkezés alkalmával módosíthatja sőt erősen ajánlott módosítani a páciensnek. Az első és a módosított jelszót is egy hash függvénnyel kódoljuk. Ehhez a táblához létrejöttétől kezdve már csak a páciens fog tudni hozzáférni, senki másnak nem lesz jogosultsága szerkeszteni azt.

A negyedik tábla fogja tartalmazni a páciens további adatait, mindent ami alapján kapcsolatban maradni, értesíteni vagy a kezelés szempontjából lényeges de tömör adatként szükségünk lehet rá.

1. **Tábla:** *patient\_sec*
2. páciens id
3. birthplace
4. address
5. phone
6. email
7. mother\_name
8. comment
9. birth\_name

A patient\_id megint csak az előbb látott módon öröklődik a második táblából, így biztosítva az összeköttetést egymással, ezek azok az adatok amelyeket az id kivételével bármely fél képes módosítani. A birthplace a születési helyet, az address a teljes lakcímet (irányítószám, település, utca, házszám), phone a telefonszámot, míg a mother\_name az anyja születési nevét kell hogy tartalmazza. A birth\_name (születési név) és az email-t nem kötelező megadni, ezeken túl a comment mezőben a pácienshez tartozó megjegyzendő információ írható (pl: gyógyszer allergia) de ennek a kitöltés sem kötelező.

Az ötödik és egyben utolsó tábla, a látogatásokat fogja adminisztrálni

1. **Tábla:** *patient\_visits*
2. id
3. patient\_id
4. staff\_id
5. date
6. text

Az id a már fent megismert auto increment érték, a patient\_id és a staff\_id a patient és staff táblából öröklött szám, a date a bejegyzés készítésénél elmentett dátum. A text meg a bejegyzést szövegét fogja tárolni.

Az adatbázis teljes felépítés a 2. mellékleten összefoglalva.

## A program felépítése

### FogOrvosiRendelő (WinForm alkalmazás)

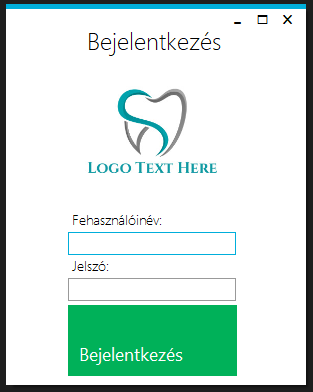
A program felépítésénél MVC Model-View-Controller tervezési mintát fogom alkalmazni. A módszer lényege hogy elkülönítse egymástól a felhasználók által látott és közvetlen használt View-t (Nézet) a Model-től ami maga a szoftver magja, itt zajlanak le a legelemibb műveletek, ez fér hozzá magához az adatbázishoz. És a Controller az ezt a két réteget összeköti és kommunikál mindkettővel, meghívja a megfelelő függvényeket, és értesíti a View-t. A szoftver működése során több különböző ablakot fog kezelni épp ezért több osztály lesz mindegyik rétegben. A megjelenítő réteg GUI (**G**raphical **U**ser **I**nterface)-ját a MetroFramework letisztult külsejű fogja biztosítani leváltva a véleményem szerint elavult WinForm kinézetű eredeti GUI-ját.

Ezt a fent említett három réteget fogom az alábbiakban részletezni, az osztályok és a közöttük lévő kapcsolatokra is kitérve. A legelső elem ami indulásnál fogadja az orvost a View réteg LoginForm-ja.

Csak ezt a bejelentkezési felületet fogom ilyen részletesen képekkel illusztrálva részletezni.

### View Réteg

#### Bejelentkezés (View)

A program indításánál lefutó első parancs:

1. ábra

* Application.Run(new LoginForm());

|  |
| --- |
| controller.loginController(userName, passWord, loginF);  │ ↓ │ │  │ felhasználóinév └→ jelszó └→tovább viszi a  │ megnyitott loginForm  │  │  │ Metódus meghívása:” controller” az osztályt, a  └→ „loginController” a konkrét metódust hívja meg az  osztályon belül.  2. ábra |

A célja az hogy illetéktelen személy ne férjen hozzá ezekhez a bizalmas adatokhoz és ne manipulálhassa azokat. Maga a form semmi mást nem csinál minthogy a „Bejelentkezés” gombra, pontosabban a MetroForm által kínált (tile) „csempére” kattintva textbox[[3]](#footnote-3)-okból kinyeri a szöveget és továbbítja azt a controllernek. *1.ábra*

|  |
| --- |
| public void loginController(string userName, string passWord, LoginForm loginF)  {  if(check(userName)&&check(passWord))  model.loginModel(userName, passWord, loginF);  }  public bool check(string text)  {  if (text == "")  {  DialogResult dialog = MessageBox.Show("Felhasználóinév és jelszó megadása kötelező!","HIÁNY!",MessageBoxButtons.OK,MessageBoxIcon.Warning);  return false;  }  return true;  }  3. ábra |

Miután meghívásra került controller osztály adott metódusa *2.ábra*, az csinál egy gyors ellenőrzést megnézi hogy a mezőkből kinyert és továbbított értékek nem üresek-e, ha mégis azok lennének akkor nem is meg mélyebbre a program és hibaüzenet keretében visszatér az Bejelentkezési felületre. Abban az esetben ha nem üresek egy réteggel lejjebb kerül innen már a model rétegbe.

Az itt megírt metódus *3.ábra* szintén a fentről kapott paraméterekkel dolgozik de még mielőtt azokkal bármit is kezdene, létrehoz egy MySQL kapcsolatot. Ehhez egy külön a programhoz létrehozott MySqlConn osztályt használok ez az osztály kezel és hajt végre ténylegesen minden sql parancsot és ez adja az adatbázis elérhetőségét is.

Itt a LoginModel rétegben *4.ábra* ezt az osztályt példányosítjuk és használjuk a metódusait. Először is csatlakozunk az adatbázishoz, azt meg is nyitjuk és a megnyitott adatbázisnak elküldünk egy sql query-t (utasítást). Ami kivételes esetben nem egy komplett utasítást tartalmaz most hanem egy előre az adatbázisban létrehozott procedure-t (eljárást) készít elő, ezt feltöltjük külön-külön értékadással. Majd mivel az eljárás ismeretében tudom hogy ez egy számadatot adhat csak vissza egyből egy int típusú változóba futtatom le.

|  |
| --- |
| public void loginModel(string userName, string passWord, Form loginF)  {  MySqlComm mysql = new MySqlComm();  MySqlConnectionDatabase connDb = new MySqlConnectionDatabase();  mysql = connDb.connection();  mysql.open();  string query = "stafflogin";  cmd = mysql.getConnect(query);  cmd.CommandType = CommandType.StoredProcedure;  cmd.Parameters.AddWithValue("username", userName);  cmd.Parameters.AddWithValue("password", passWord);  int count = int.Parse(cmd.ExecuteScalar().ToString());  4. ábra |

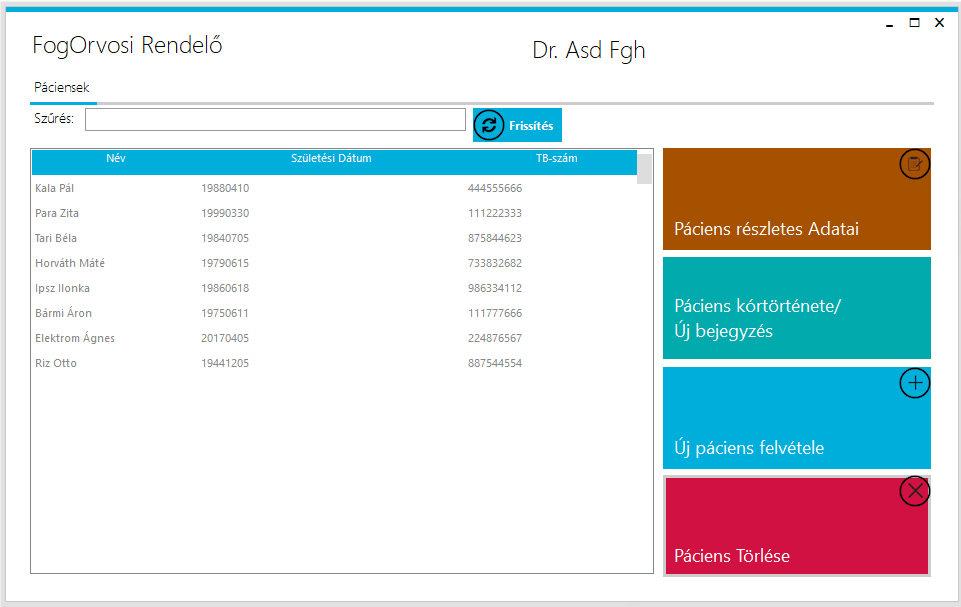
Az így kapott értéket átküldöm egy if *5.ábra* elágazáson ellenőrizve hogy az egyenlő-e eggyel, mivel ha az eljárás során a megadott nevet és jelszót vagy megtalálhatja egyszer vagy egyszersem tehát nullaszor. Ha nulla (vagy bármi okból kifolyólag más ami nem egy) abban az esetben megjeleník egy hiba üzenet hogy „Hibás felhasználói név vagy jelszó” (mivel nem található ilyen név jelszó páros az adatbázisban). Azonban ha egy lesz az eredmény az if feltétel igaznak bizonyul és az az ág lép életbe és csinál még egy

|  |
| --- |
| if (count == 1)  {  string getID = "SELECT staff.id FROM staff WHERE staff.username=@username;";  cmd = mysql.getConnect(getID);  cmd.Parameters.AddWithValue("@username", userName);  int id = int.Parse(cmd.ExecuteScalar().ToString());  MainForm mainF= new MainForm(id);  mainF.Show();  loginF.Hide();  mysql.close();  }  else  {  MessageBox.Show("Hibás felhasználóinév vagy jelszó!");  mysql.close();  }  }  *5.ábra* |

lekérdezést ami kinyeri a felhasználó névhez tartozó azonosítót.

(Ez a későbbiekben a belépett személy azonosításához lesz szükségszerű.) Ezek után a kinyert értékkel továbbadva megnyitja a MainForm-ot és bezárja a LoginForm-ot. Majd az adatbázis kapcsolat bezárul.

#### Fő ablak (View)

Miután a bejelentkezés sikeresen megtörtént ez a felület lesz talán a legtöbbet látott része a programnak. Ezen keresztül történik a meglévő páciensek kiválasztása, azok szerkesztés, törlése, bejegyzések hozzáadása és az új páciensek felvétele.

A form konstruktoraként példányosítom a MainFormhoz tartozó MainController osztályt (három külön controller osztállyal dolgozom egy a login egy a main és egy paciensekhez tartozó formok számára). Ezután létrehozom a kapott azonosítót az egész main osztályban és végül két metódust autómatikusan lefuttatok az egyik a bejelentkező személy nevének kiírására szolgál a form neve mellett felül egy label-en belül, azt az elöbb meghatározott azonosítóból nyeri ki. A másik metódus a táblázatot tölti be.

Ezt a központ szerepet betöltő táblázat a form nagyrészét le is fedi és több és az össze többi funkcióra kihatással van. Ezt egy a MetroFramework-höz készített ListView-t használok, csupán a páciens legszükségesebb adatai szerepelnek, csupán a név a születési dátum és a tb-szám. Ezt is a későbbiekben is csak szokásosnak említendő a loginnál használt MySqlConn osztály által a model-ben használt metódus által töltődik be a szükséges rétegeken áthaladva természetesen. Annyi eltérés van talán a később használtakhoz képest hogy míg legtöbbször egy vagy több string-et nyerünk ki az adatbázisból string-ként is adunk tovább a felsőbb szinteknek, ennél a model réteg egy ListView-t ad vissza mivel ismeretlen akár nagyobb mennyiségű adat is lehet nem csupán egy érték, itt majd hogy nem egy sor hanem egy egyész adattáblát ad vissza ami visszatérve be is töltődik a táblázatunkban. Ennek a táblázatnak a leglényegibb része a kijelölés ez határozza meg a későbbiekben hogy a többi gombra kattintva tudjanak azok értéket is tovább vinni. Ehhez először arra volt szükség hogy a táblázatban letiltsuk hogy több sort is kijelölhessünk egyszerre, illetve egy „SelectedIndexChange” esemény ami hatására egy egyszerű if feltétellel le ellenőrzi a metódus hogy a lista hossza tartalmaz e még elemet kiürült teljesen ne próbálja meg kijelölni a legelső elemet automatikusan(mivel nincs mit).

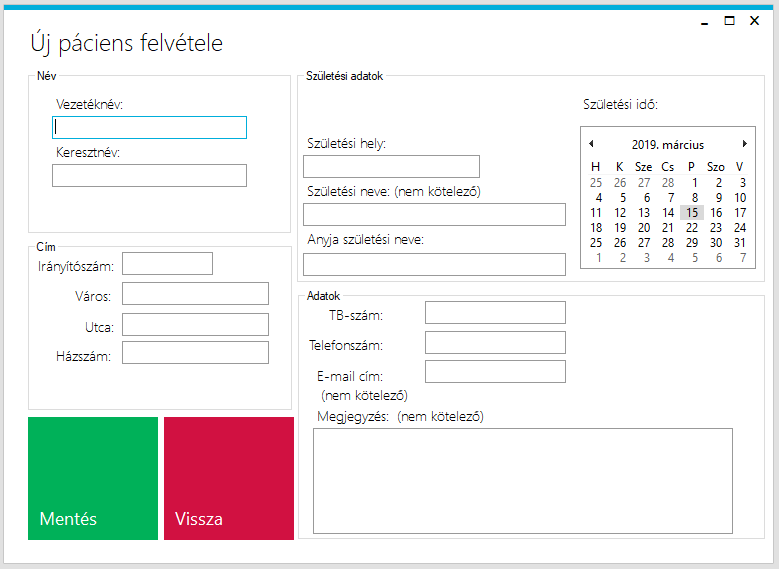
A ListView fölött egy élő szűrő található, amint bármilyen szöveget számot elkezdünk beírni a fent említett táblázatban található értékeket átszűri és csak azokat mutatja akiknek valamelyik értéke megegyezik a begépelt szöveggel/számmal. Így módon akár név akár születési idő vagy tb-szám alapján is gyorsan meglehet találni bármelyik regisztrált pácienst. Ennek a megvalósítására a textbox-ok egy hasznos eseményét (Event[[4]](#footnote-4)) a TextChanged, ez lehetővé teszi hogy amint bármilyen változás történik a textbox-ban, akár beírás akár törlés, azonnal lefuttatok egy metódust ami létrehoz egy újabb ListView (az előzőt a metódus legelején kiürítjük) de itt már a SQL lekérdezés része az is ellenőrizze a mezőbe begépelt érték szerepel e az adatsor bármelyik tagjában. Csak ezeket az egyezést mutató sorokat fogja visszaadni a táblázatnak.

A mező mellett található Frissítés törli ezt a szűkítést és a teljes listát tölt be ismételten.

Ezek mellett további négy gomb található még a mainform-on ezekből egyedül a Törlés gomb nem nyit meg egy újabb formot. Ez csupán miután ellenőrzi hogy van e kiválasztott elem, egy szinttel lejjebb megjelenít egy messagebox-ot hogy valóban törölni szeretné e a kiválasztott (a nevet hozza a táblázatból magával) pácienst az adatbázisból. nem esetén visszatér az egyszerűen a főablakhoz. igen esetén viszont lent a model rétegben végrehajt egy a fent a kiválasztott paciens tb száma segítségével beazonosítható DELETE utasítást az SQL adatbázisban. Miután megtörtént történik egy adatbázis frissítés.

A tovább három gomb „Páciens részletes adatai”; „Páciens kórtörténete/Új bejegyzés”; „Új páciens felvétele” egy-egy új formot hoz létre.

#### Új páciens felvétele (View)



Közvetlen mainform-ból nyíló NewPatientForm. Ez a felület anélkül is lefut hogy kapott volna bármit is az előző osztálytól, mivel itt új pácienst hozunk létre, nem egy meglévőt módosítunk. Ez a form tartalmazza a legtöbb textbox-ot mivel minden értéket nevet kettő, míg a címet négy értékre tagolva viszi tovább.

Az osztály konstruktorának nincs is sok extra dolga, csupán példányosítja a hozzátartozó PatientController-t (később a többi páciensekkel foglalkozó osztály is ezt fogja használni átjárónak)., ezután meg a getFromForm() metódus meghívásával. Egy alapértéket adok az összes adattagnak amivel a form dolgozni fog.

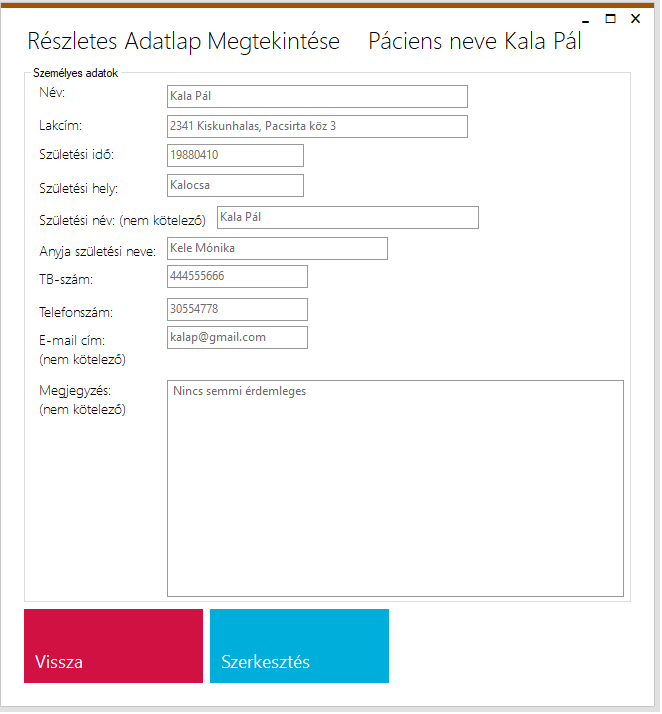
Egy adat kivételével minden adat megadása külön textbox által megy végbe, egyedül a születési idő nem, ezt egy MonthCalendar[[5]](#footnote-5)-ból lehet kiválasztani, a naptár kijelölési tartománya le van szűkítve egy mezőre és csak egy értéktartományon belül lehet kijelölni (nem lehet 1900 elé és 2040 felé sem haladni). A mezők egy részének kitöltése nem kötelező ezek külön jelezve is vannak. A form-on két gomb szerepel egy Vissza és egy Mentés. A Vissza gomb lefuttat egy metódust ami ellenőrzi hogy a textbox-ok üresek e, amennyiben igen bezárja ezt a formot, ha mégsem üres legalább egy mező akkor megjelenít egy messagebox-ot hogy el nem mentett adatokat tartalmaz a form valóban bezárja-e azt. Az így kapott eredményt egy if feltételben DialogResult[[6]](#footnote-6) alapján vagy bezárja az formot vagy ott marad a további adatbevitelre részen.

A Mentés gombra kattintva lefut egy ellenőrzés hogy minden szükséges textbox ki lett e töltve és a speciális igényű mezők például ahová nem kerülhet betű például telefonszám egy speciális ellenőrzés (trim[[7]](#footnote-7)) után, amennyiben minden szükséges mező megfelelően lett kitöltve. A PatientModel osztályba már három részre darabolva aszerint hogy mely adattáblába kerül mentésre a tartalma. Először az elsődleges adatok kerülnek mentésre, a név, a születésidátum és tb-szám.

Ezután jön a másik kettő, a másodlagos adatok tartalmazó. Itt mivel a két tábla kapcsolódik egy beágyazott (nested[[8]](#footnote-8)) SQL lekérdezés zajlik le. Ennek az az oka hogy a patient\_sec adattáblának a patient\_id tagját ami összeköti a két táblát az elsőnek létrehozott patient táblában jön létre mint auto increment szám. Ezt a számot nem ismerjük mert az előbb metódusban jött csak létre, ezért beágyazott lekérdezés keretében a tb-szám alapján megkeresi az adatbázis megfelelő sorát és kiszedi a hozzátartozó id-t, amit újabb lekérdezés nélkül el is tudja menteni.

Harmadik darabka a páciens webes bejelentkezéshez szükséges felhasználói nevet és jelszavat hozza létre. Itt ugyancsak egy nested SQL lekérdezés zajlik le az előző okból kifolyólag. Alapértelmezett felhasználói névnek, a páciens saját tb-számát míg jelszónak a születésidátumát, biztonsági okokból ezt előbb átküldjük egy trim metóduson hogy csak a számok maradjanak benne és ezt meg kriptográf metóduson Blowfish[[9]](#footnote-9) algoritmussal hogy a jelszó kellően védett legyen. Később a webes felületen bejelentkezéskor is a megadott jelszó átmegy egy hasonlón és a két kódolt jelszó speciális összehasonlításán dől el be tud-e jelentkezni a paciens. Ezt a nevet és jelszavat természetesen azon a felületen lesz lehetősége a páciensnek módosítania. Ehhez a névhez és jelszóhoz a létrehozást követően nincs hozzáférése a kezelőszemélyzetnek.

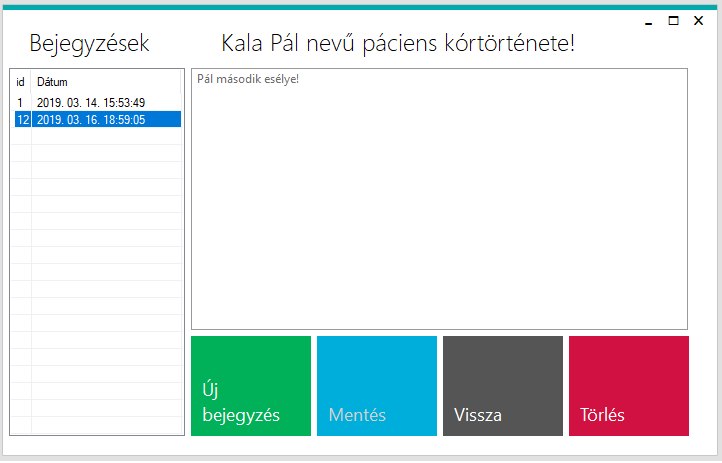
#### Páciens részletes adatai (View)



A most következő form szintúgy a mainform-ból lesz elérhető, Ez a PatientDetailsForm. Ez viszont az előzővel ellentétben csak abba az esetben nyílik meg a főablakból ha van kiválasztott elem táblázatból. Ha van akkor onnan hozza is magával a kiválasztott személy második részelemét SubItem[2], mivel az ablak elsődleges célja a kiválasztott páciens adatainak megjelenítése, és ha azok megvannak szükség szerinti módosítása. Az osztály konstruktora miután példányosítja az előbb is használt PatientController osztályt elkezdi feltölteni egy metódussal az összes textbox-ot, ehhez a fentről kapott tb-számot használja. Mivel ezen az ablakon az adatok megtekintésén túl a módosításra is sor kerülhet lényegi hogy a kitöltéshez szükséges adatokon kívül az páciens id-jét is kinyerjük mivel csak így biztosítható probléma mentesen hogy az eddig azonosításra kapott tb-számot is módosíthassuk. Ezeken túl a konstruktor végre hajt még egy metódust ami a textbox-ok szerkeszthetőségét szabályozza, a metódus mindössze annyit csinál hogy ha a szerkesztés mód aktív akkor lezárja azt, ellenkező esetben meg aktiválja azt, ezt az állapotot egy bool[[10]](#footnote-10) típusú változóban tárolom. A cél hogy nyitásnál nem legyenek szerkeszthetőek az adatok. A form-hoz két gomb tartozik egy a visszalépésért, egy másik meg a szerkesztésért és mentésért. A vissza gomb funkciója egyszerű, bezárja a formot és visszatér az főablakhoz.

A másik, szerkesztés/mentés gomb funkciója már kettős. Egyrészt megnézi hogy a szerkeszthetőség állapotát tároló bool változó milyen állapotban van és a metódus e szerint dönt hogy mentse-e el a változásokat, ez abban az esetben történik meg ha a szerkeszthetőség állapota „true”. Ez esetben a textboxok-ból minden értéket kiment azok szintenként lejutnak és felülíródnak az adatbázisban a PatientModel réteg segítségével. Ellenkező esetben ha szerkeszthetőség „false” állapotban volt, a textbox-ok szerkeszthető állapotba kerülnek. Ahhoz hogy ez az állapot a programot használók számára egyértelmű legyen, a gomb színe és szövege is az állapothoz igazul. Szerkeszthető módban „Mentés” szerepel zöld háttérrel. Megtekintő módban meg a „Szerkesztés” kék háttérrel.

#### Páciens kórtörténete / új bejegyzés (View)

PatientVisitsForm mint ahogy a fentebb részletezettnél ez is a főablakból nyílik és csak akkor ha van kiválasztott páciens a táblázatból hozza magával a tb-számát és magával hozza továbbá a bejelentkezett személy azonosítószámát is, ennek szükségességét lentebb részletezem. A form kontruktora a szokásos módon példányosítja a PatientContorllert a kommunikációhoz, az ablak tetejére kiírja egy label-be a fentebb kiválasztott paáciens nevét, és bal oldalra egy ListView-ba feltölti az ehhez a pácienshez tartozó eddigi bejegyzések listáját, a kórtörténetet. Ezek a bejegyzések 4 információt tartalmazn, egy egyéni azonosítószámot, egy időbélyegzőt az elkészülés időpontja szerint (ez az adat alkotja a lista fő gerincét) továbbá tárolja hogy mely pácienshez írodtak, ez szükséges hogy ebben a listbe csak a hozzátartozó bejegyzések kerüljenek, illetve a webes felületen is így tekintheti mindenki meg csakis a saját bejegyzéseit és még tartalmazza a bejelentkezett személy azonosítóját is, hogy követhető legyen ki hozta létre a bejegyzést.

A főablakhoz hasonlóan itt iss vizsgálni kell a „SelectedIndexChange” eseményt hogy ne omoljon össze a program abban az esetben ha nem maradna kiválasztandó elem és ne keresse üres lista esetén a legelsőt. Az ablak másik felét egy nagy méretű többsoros textbox alkotja, ez két funkcióval bír, alapesetben csupán megtekintésre szolgál, ide töltődik be a kiválasztott bejegyzés, míg új bejegyzés írásánál ide lehet a kívánt szöveget begépelni. Ezeken túl a form még négy gombbal rendelkezik, egy az új bejegyzések írását engedélyezi, egy elmenti azt, egy visszalép a fő ablakra és eggyel meg törölni lehet a bejegyzéseket.

A vissza gomb az eddig megszokott módon egy form bezárást tartalmaz hogy visszatérjen a program az előző ablakhoz. A Törlés gombbal amennyiben van ki kiválasztva elem a listáról, feldob egy dialog box-ot és az eredményétől függően eldönti hogy az adott elemet szintenként lejjebb adogatva törölje e az adatbázisból, ha a törlés megtörtént a baloldali lsita azonnal frissül. Az „Új bejegyzés” gomb az alapértelmezetten zárt központi nagy szövegdobozt szerkeszthetővé teszi és amennyiben van benne betöltött szöveg az kiüríti. Továbbá az alapértelmezetten letiltott mentés gombot is elérhetővé teszi.

A mentés gomb meg az előbbi párja, amennyiben a textbox kitöltésre került (ezt le ellenőrzi egy szinttel lejjebb egy trim metódussal ami eltávolítja a fölös whitespace-eket és megnézi a string nem egyértékű e egy üres string-el. Ha nem egyenlő már megy is le a model rétegbe és elmentésre kerül az üzent, a készítője, a páciens akihez tartozik és az éppen aktuális idő.

### Controller réteg

A controller rétegről nem nagyon tudok sokkal többet mondani mint amit már az előbb taglalt részelemekben amik egyben a View réteget alkotnak. Ott a legtöbb része említve volt, a feladata nem több mint a Model és a View közötti kommunikáció és adatok ellenőrzése volt. Legtöbb metódusa rövid tömör, adat továbbvitelre szolgált, az áthaladó adatok ellenőrzése és dialog box-ok megjelenítése volt a feladata. A bejövő adatok hosszát több szinten, részben magukon a form-okon a textbox-ok tulajdonságán, részben meg a controller rétegben kerültek ellenőrzésre.

### Model réteg

A három model réteg, lényegében a program magja, ezek hajták végre az elemi közvetlen az adatbázisra kihatással lévő módosításokat. Ez az a réteg amihez kívülről más módon nem szabad hozzáférni, ide már csak olyan és az az információ, adat juthat el amit átengedtünk a program többi részén. Minden model osztály bármely adatbázishoz kapcsolatos metódusát egy try-catch[[11]](#footnote-11) parancson át fut le, megakadályozva a program hirtelen összeomlását és a hiba esetén annak feltárását esetleges orvosolását.

#### LoginModel

A három model közül a legrövidebb, elkülönítésétéhez csupán azért ragaszkodtam mert ez egy olyan része a programnak ami egy részt rendkívül fontos, mivel jogosulatlan személy ne férhessen hozzá a programhoz, másrészt meg mert egyedi és ez az osztály normál körülmények között csak egyszer kerül meghívásra a programba történő belépéskor. Az után már későbbiekben amig a felhasználó bevan jelentkezve használata nem indokolt.

Maga az osztály is csupán egy nagyobbacska metódusból áll amit fentebb a LoginForm bemutatásánál részleteztem is.

#### MainModel

A második model a Main a program fő ablakának a végrehajtásaiért felel. Ebben már lényesen több metódus szerepel. A legfontosabb talán maga az elsődleges táblázatnak feltöltése, ez kardinális része hiszen e-nélkül a program egy részének (Új paciens létrehozása) kivételével minden részelem ebből dolgozik ebből nyer ki egy a futtatáshoz szükséges adatot. Ez a loadListViewPatient mást nem is csinál csupán létrehozza az adatokat magának az elsődleges táblázatnak. Ezen túl csupán három másik metódust tartalmaz. Egy a páciens törlésért, egy az élő keresésből adódó elsődleges táblázat újra feltöltésért, ez szinte teljesen ugyanazt végzi el mint az első annyi plusszal hogy egy „LIKE” operátort is használ a lekérdezésben. Az utolsó negyedik metódus csupán a bejelentkezett személy nevét írja ki az ablak tetejére, apró de hasznos funkció hogy ha véletlenül bejelentkezve maradna valaki, az utána használó észrevegye hogy nem az ő adataival van bejelentkezve.

#### PatientModel

A PatientModel, talán a legtöbb sort tartalmazó osztály 24 különböző metódust tartalmaz. Ezeknek egy része csupán azért fele hogy más metódusok által kinyert adatot vissza lehessen egyesével is juttatni a feljebb lévőszintekre. De van közöttük ami a MainModel-hez hasonlóan egy listát ad vissza ez esetben a bejegyzéseket a kórtörténethez. Itt kerülnek több metódusra osztva az adatbázis feltöltésére szolgáló metódusok. Míg a többi model osztályban csak lekérdezések és egy törlés szerepel itt több, a bővítésével és módosításával foglalkozik. Ezeknél kardinális hogy csak is a helyes adatok jussanak el erre a szintre több lépcsős ellenőrzés után. Természetesen ebben az osztályban is lényeges hogy szinte minden metódus egy try-catch folyamaton essen át, a hibák feltárása, kezelése és szükségtelen leállások végett.

És még ebben az osztályban szerepel a sokszor használt trimText metódus is ami beérkezett szöveget mégegyszer le ellenőrizve megszabadítja a fölös írásjelek és szóközöktől.

### WebFogRendelo (Webes alkalmazás)

A project másik fele PHP alapokra épülő weblap, ami tartalmaz néhány JavaScript kódot is, a megjelenítésért HTML kód és CSS formázás teszi egységessé és letisztultá. Az oldal funkciója az előzetesen legalább egyszer már a rendelőben járt ezáltal az adatbázisba felvett páciensek kiszolgálását látja. Miután ez az adatfelvétel a rendelőben megtörtént automatikusan generálódik egy belépési név-jelszó páros amivel képesek lesznek belépni a lentebb említendő funkcióit használni az oldalnak.

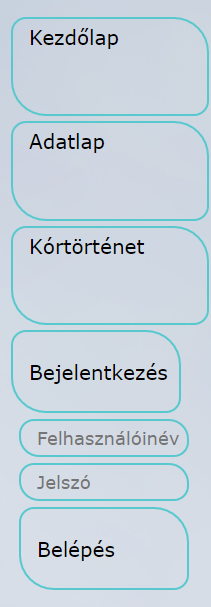
Természetesen ezen információk nélkül is megtekinthető az oldal de így a legtöbb funkciója használhatatlan lesz, tájékozódásra és informálódásra így is lesz lehetőség, a kezdőlap segítségével.

Az egész weblap egy egyszerűbb szerkezeti felépítéssel rendelkezik a gyökérmappa csupán magának az index.php-nak ad otthont, minden más megjelenítendő oldal ezen túl egy „php” nevű mappán belül található, a kiegészítő képeket mint például a háttér egy „pics” mappában, az adatbázishoz történő hozzáférést biztosító fájlok egy „config” a csak HTML kódot tartalmazó fájlok a „html”, scriptek a „js” míg a formázásért felelő stílus fájl a „css” mappába került. Ebbe a logikus szerkezetbe kerül bele minden egye fájl a projektben. A weblap szerkezeti felépítése a 3. és 4. számú mellékletben megtekinthető.

#### Index.php (kezdőlap)

A weblap legelemibb része maga az index.php. Igyekeztem a lehető legkevesebb kódot ezen az oldalon tárolni, mivel ez leginkább egy keret oldal, bárhová is kattintunk minden egyes menüpont és funkció végsősoron erre az oldalra fog rátöltődni, ezen fog megjelenni. Az egész oldal mindössze kevesebb mint 50 sornyi php-kódot fog tartalmazni, azok is csak egyszerű lekérdezések és kiírások. A nyitótag[[12]](#footnote-12) után elindítjuk a munkamenetet „session\_start()[[13]](#footnote-13)” egyből betöltjük a html lap nyitó tag és az utána következő head is ami definiálja az oldal alap tulajdonságait mint, nyelv, nézet, oldalnév és végül megadja a különböző külső források elérhetőségét mint a scripteket tartalmazó fájlok és a stílus lap elérhetőségét. Ezeket a session\_start() után felsoroltakat kivétel nélkül a head.html fogja tárolni a nyitó body tag-el egyetemben. Ezt követi egy apró if- feltételbe szabott lekérdezés hogy létezik-e már a $\_SESSION[[14]](#footnote-14)[’id’], nevű globális változó, amennyiben ez létezik az azt jelenti hogy jelenleg is egy bejelentkezett felhasználó ül a böngésző elött és ez esetben azt a navigációs sávot tölti be oldalra amelyik a bejelentkezett személyeknek szól. Ellenkező esetben ha az if feltétel false ágra fut még nem történt meg a bejelentkezés így az a menü jelenik meg ami a login form-ot is tartalmazza.

Ezt követően még két darab if-el történő feltétel vizsgálat történik, az első megnézni próbálkoztak-e jogosulatlanul hozzáférni bármely al-oldalhoz, ha igen megjelenik egy ehhez kapcsolódó figyelmeztető szöveg hogy jelentkezzen be előbb. Míg a másik vizsgálat azt ellenőrzi hogy történt-e jogszerű hozzáférési kísérlet az al-oldalakhoz, ha igen, az adott aloldalhoz tartozó adatok betöltésre kerülnek a kezdőlap felületére, ellenkező esetben a home.php kezdőlap tartalma töltődik be.

Végül ezeket zárva még egy fájl meghívással „file\_get\_contents(’fájl neve’)” betöltődik a html oldal kódok záró tag-ei.

A navigációs menü egy olyan eleme a weblapnak ami úgy gondolom külön bekezdést érdemel nem bejelentkezett állapotban egy egyszerü 5 hivatkozást tartalmaz amelyek egy adott php elemre mutatnak. Míg nem bejelentkezett állapotban, három hivatkozáson túl először tartalmaz egy gombot ami kattintás hatására lenyit egy három input-os formot ez szolgáltatja a bejelentkezést magát. az egész menü egy egységes grafikát követ, magát a menüpontok eltünését és megjelenését egy jQuery script és CSS kód segítségével valósítottam meg. A belépés gombra kattintva kitöltött input mezők esetén POST metódus jön létre ami átadja az adatait a login.php-nak de ezt egy kicsit lejjebb részletezem.

#### Php almappa

Ez az almappa tartalmazza a legtöbb fájlt szám szerint hét és ez adja vissza az összes menüpont értékét. Ezeken fogok az alábbiakban végig menni.

##### home.php

Mint minden egyes php-fájlom ez is a „session\_start()” paranccsal indul hogy hozzákapcsolódjon a munkamenethez. Ez a lap csupán magának a kezdőoldalnak a fő szövegét tartalmazza, egy bemutató egy nyitvatartás egy elérhetőség.

##### login.php

|  |
| --- |
| $sql="SELECT password "  . "FROM patient\_web "  . "WHERE username=?;";  $stmt =$connection->prepare($sql);  $stmt -> bind\_param("s", $username);  $stmt -> execute();  $stmt -> bind\_result($hash\_pass);  $stmt -> fetch();  $stmt->close(); |

A login.php lesz felelős a felhasználói bejelentkezésért. A munkamenet indítás után betölti az adatbázis elérhetőségét tartalmazó connect.php fájl. Az említett fájl csupán az adatbázis csatlakozáshoz szükséges adatokat tartalmazza, így ha máshová kerül az adatbázis ennek az egy fájlnak a módosításával az egész programon belül az elérhetőség. A bejelentkezést szolgáló metódus egy nagy if ágba került bele ami biztosítja hogy csak a bejelentkezés gombra kattintás után induljon el az folyamat. Rögvest a folyamat elején kimenti a munkafolyamatban tárolt előzőleg beírt bejelentkezési nevet és jelszavat ezeket változókban tárolom. Majd írok egy string változót benne a bejelentkezéshez szükséges SQL lekérdezéshez.

A PHP szerencsére alapértelemzetten támogatja a MySQL rendszert így nem szükséges hozzá semmilyen összekötő konnektort használni elég az alap erre írt parancsokkal dolgoznom. A lekérdezéseket végig az egész webes alkalmazásban „mysqli\_stmt[[15]](#footnote-15)” osztály segítségével fogom végrehajtani. A művelet minden esetben a következőképp fog végrehajtódni. Az $sql-ben a kérdőjelek minden esetben egy később feltöltődö értéket határoznak meg, így az adott kódba több minden is behelyettesíthető igény szerint. A statement megkapja a csatlakozáshoz szükséges adatokat ($connection) és felkészíti magát az a lekérdezéshez (prepare($sql)). Ezután jönnek a kérdőjelek feltöltése a bind\_param() függvénnyel itt megkell adni kötött sorrendben hogy milyen típusú adatot viszünk be (i=int, d=double, s=string, b=blob), majd vesszővel elválasztva az ezek szerinti értékeket. Természetesen ha nem szükséges semmilyen paraméter bevitele az előző sor kimaradhat, ezután jön a lekérdezés végrehajtása az execute(); paranccsal. Majd bind\_result($eredmény) függvénnyel megint csak kötött sorrendben megadható milyen változóba kerüljön a kinyert adat. És jöhet a fetch() ellenőrzi ennek a sikerességét és végrehajtja azt végül a close() függvénnyel bezárul a kapcsolat. Jelen esetben mivel ez egy bejelentkezés és a jelszó titkosítva van,

|  |
| --- |
| if (password\_verify($password,$hash\_pass)) |

első körben szükséges a névhez tartozó jelszó kódolt kinyerése, hogy azután az és a beírt

jelszó speciális if feltételbe zárt összehasonlítása után eldöl hitelesek e a belépési adatok, ha nem vissza kerül az alapoldalra és lehet újból próbálkozni. Ha viszont sikeres jöhet még egy gyors lekérdezés ami visszaadja a névhez tartozó id-t (azonosító számot) ez az értéket egy globális változóba tároljuk. A későbbiekben a páciensről minden adatot ez alapján lehet majd kinyerni. Ha ez megvan a lap visszairányít a index.php-ra (ez minden php lap közös tulajdonsága).

##### logout.php

Az oldal tartalma mindössze pár sor. Megnyitja a meglévő munkafolyamatot majd le is rombolja az végérvényesen a session\_destroy() paranccsal majd vissza irányit a kezdőlapra.

##### patient.php

Ez az oldal lesz felelős a páciens adatlapjáért annak tartalmi, esztétikai részéért és ezen lesz majd lehetősége bizonyos adatainak a módosítására. Innentől kezdve az oldalak egy közös tulajdonsága lesz hogy szinte az egész fájl tartalma egy nagy if ágba kerül a szerint hogy van e bejelentkezett személy, tehát hogy van e $\_SESSION[’id’] vagy nincs. Amennyiben igaz a feltétel a fentebb már taglalt módon végrehajtódik először létrehozunk egy változót ezt fogjuk az oldal végén visszaadni mint string ez fogja tartalmazni az új megjelenítésre szánt részt a kezdőlapon. Ezt egy form és egy div nyitással kezdek, a div az esztétikáért, míg a form azért hogy a benne megjelenítendő adatok közül egy néhányat legyen lehetőség a későbbiekben módosítani.

|  |
| --- |
| $address = $row['address'];20.ábra |

Majd jön egy SQL lekérdezés, annyi különbséggel hogy a kötött sorrendű kivétel helyett itt egy „while[[16]](#footnote-16)” ciklusba kerül a fetch\_assoc() tagolással ami egy asszociatív tömböt eredményez. Ez a sok különböző adattag miatt szükséges, így elkerülhető hogy csak sorszámmal hivatkozzak valamire, ehelyett az adatbázisban meghatározott névvel kell azonosítanom. 20. árba

Ez után mivel tudom ez a ciklus csak egyszer meg végbe, nem ismétlődik, létrehozok egy változóban egy HTML táblázatot gondosan azonosítókkal ellátva, a fent kinyert értékekkel feltöltve, úgy hogy ott ahol olyan adat van amit lehetőség lesz módosítani azt egy input mezőbe kerül. A táblázat végére meg egy „submit” mező a módosított adatok mentésére, végül a táblázat, a div, és a form lezárása. Ezután ezt az egész változót feltesszük a globális változóba és visszaküldjük a kezdőlapra, és ott már ezzel a tartalommal együtt jelenik meg.

##### saveTable.php

Ez a fent taglalt patient.php társ oldala, ez hívódik meg abban az esetben ha submit-elve lett volna az ott látható táblázat, azaz ha az ott bekövetkezett változásokat valaki menteni szerette volna. A lap abból áll hogy megnézi megtörtént-e a POST az előző oldalon és ha igen akkor az onnan hozott változókat elmenti egy saját változóba, majd egy SQL lekérdezéssel UPDATE-eli az adatbázist a kinyert adatok behelyettesítésével. Ezután visszatér a patient.php-re és az már a frissített adatokat jeleníti meg.

##### visits.php

A lap célja a rendelőbe lévő látogatások alkalmával készült bejegyzések megtekintése, így lehetősége van minden páciensnek a sajátkórtörténetének visszatekintésére, időpont és a kezelő fogorvos nevének kíséretében. Ez a metódus is a már megszokott lekérdezés útján zajlik és itt is egy while cikluson át listázzuk ki az értékeket minden egyes alkalommal amikor végrehajtódik a ciklusmag, létrejön egy p[[17]](#footnote-17)-tag ami kiírja a lekérdezés által kinyert adatokból a készítés dátumát, az orvos nevét aki írta és a bejegyzés szövegét. A végén ezt az egészet visszaadja az indexnek, az meg megjeleníti.

##### change\_webdata.php

És az utolsó php lapon a pácienst webes belépési adatait van lehetősége módosítani. A megszokott módon egy if ágon belül végbemegy a lekérdezés most a felhasználói név kinyerése céljából. Majd két külön form-ot csinálok, az elsőbe kerül be a kiszedett felhasználói név ez részben emlékeztetőnek hogy miről szeretne változtatni másrész meg, mert csak meglévőtől eltérő felhasználói nevet lehet létrehozni, ezt az új nevet két input mezőbe is be kell a félre írás elkerülése végett. Amikor ez POST-ra kerül áthalad egy sor if ágon ellenőrizve hogy a beírt adatok egyformák-e, nem csak üres string-et tartalmaznak, illetve hogy nem egyforma e a jelenlegi felhasználói névvel. Ha ezeken mind túl jut akkor végbemegy SQL UPDATE lekérdezés. A második form is hasonlóképpen jár el annyi különbséggel hogy ott a jelenlegi jelszót is a felhasználónak kell megadni. És miután ugyanúgy a fent említett ellenőrzéseken átment előbb lezajlik egy lekérdezés annak vizsgálatára hogy a megadott jelenlegi jelszó valóban a jelenlegi jelszó-e, ha ezen is átjut, az új jelszó titkosításra kerül és egy újabb SQL UPDATE-el frissítésre kerül.

Minden lehetséges esetben létrejön egy bejegyzés is ami kiírásra kerül aszerint hogy sikerült-e a név vagy jelszó módosítás, illetve ha nem, akkor milyen hibával került visszadobásra.

## Üzembe helyezés, tesztelés

# Felhasználói dokumentáció

***Bevezető:***

Üdvözölük önt a FOR (FogOrvosiRendelő) Adminisztrációs alkalmazás és weblap Felhasználói útmutatójában, az alábbiakban ismertetjük a program használatához szükséges Hardveres és Szoftveres követelményeket. Bemutatjuk a program használatát és funkcióit, megmutatjuk miként könnyítheti meg a program által a jelenlegi munkáját. Kérjük a program használata elött ezen útmutatót olvassa végig és használat közben is amennyiben kérdése merülne fel.

## Ajánlott hardver-, szoftver követelmények

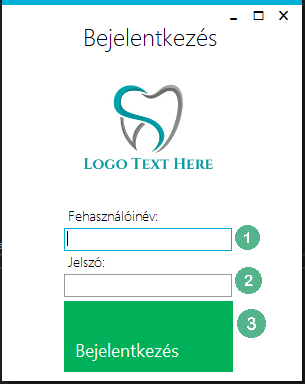
|  |  |
| --- | --- |
| Minimális rendszer követelmény:  CPU: 1 GHz  RAM: 512 MB  LAN: Internet elérés | Minimális szoftver követelmények:  OS: Windows XP, Vista, 7, 8 vagy 10 operációs rendszer  .NET Framework 3.5  Bármilyen internetes böngészőprogram |

## A program használata

### Asztali alkalmazás (a fogorvos részére)

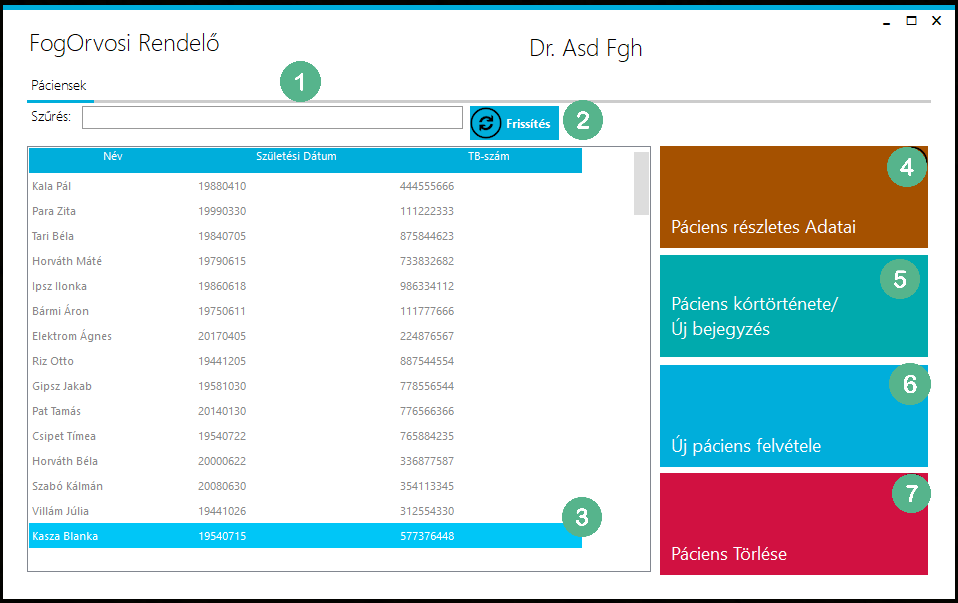
A FOR asztali alkalmazás kifejezetten a rendelők számára készült, célja a praxis hatékonyabbá és átláthatóbba tétele mind a rendelőben dolgozók mind a klientúra számára. A program képes a teljes klientúra adatainak, kórtörténetének és a látogatások tárolására. Segítségével egy jobb teljesebb képet kaphat és adhat az ön betegének és ezt később ő maga is megtekintheti otthon egy weblapon keresztül.

#### Bejelentkezés

Bejelentkezési felület. A program használatához minden esetben bejelentkezés szükséges! Bejelentkezési adatait, őrizze meg és ne ossza meg másokkal!

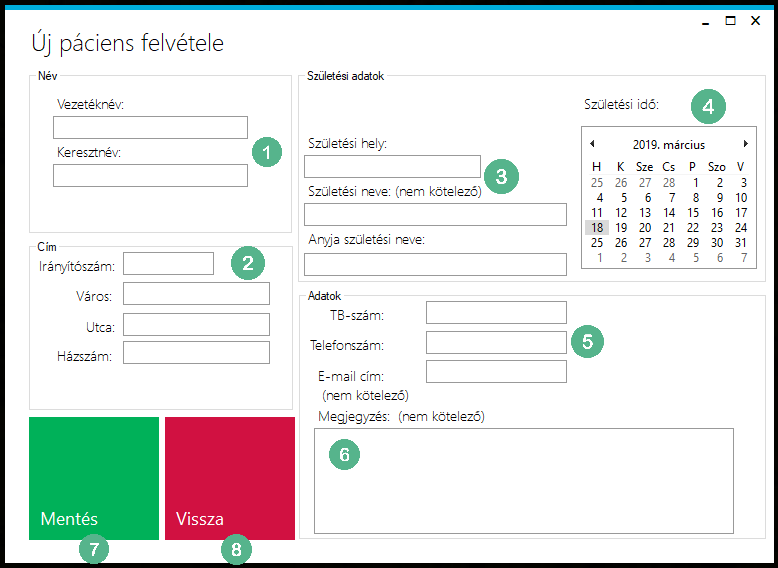
|  |  |  |
| --- | --- | --- |
| 1 | Felhasználóinév | Az ön egyéni azonosítója |
| 2 | Jelszó | Az azonosítóhoz tartozó jelszava |
| 3 | Bejelentkezés gomb | Helyes adatok kitöltése esetén a program elindul |

#### Főablak

A FogOrvosiRendelő adminisztrációs rendszer főablaka, ezt fogja a legtöbb esetben látni és használni. Az ablak közepén megjelenő táblázatban megtekintheti az ön összes regisztrált páciensét. Ezen keresztül van lehetőség a páciensek felvételére, módosítására és törlésére. Itt hozhat létre új bejegyzéseket és itt kereshet név, születési idő és TB-szám szerint az ön valamennyi regisztrált páciensei között.

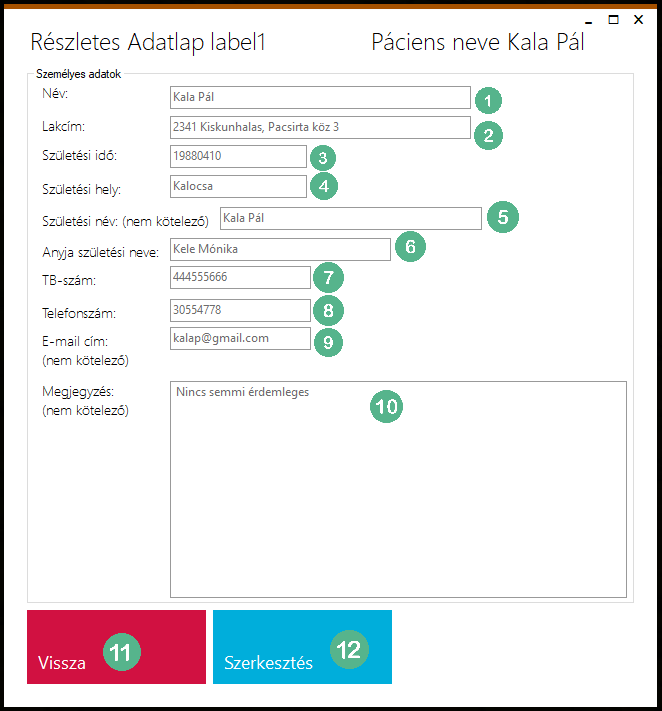
|  |  |  |
| --- | --- | --- |
| 1 | Keresés/szűrés | Élő kereső/ szűrő mező, csak írja be a keresett személy valamely alap adatát (név, szül. idő, tb-szám) és a táblázatban már csak az ön által keresett személy fog megjelenni. |
| 2 | Frissítés | Törli az ön eddigi szűrését és ismét az egész klientúrát mutatja, frissítve a legújabb adatokkal. |
| 3 | Kiválasztott személy | Ha ön bármelyik sorra rákattint onnantól az lesz a kiválasztott páciens akit az oldalt lévő gombok segítéségével kezelhet |
| 4 | Páciens részeletes Adatai | A 3. pont szerint kiválasztott személy részletes adatai tekintheti meg, itt lehetősége lesz módosítani is azokat. |
| 5 | Pácines Kórtörténete /  Új bejegyzések | A 3. pont szerint kiválasztott személy részletes kórtörténetének megtekintése és új bejegyzés írása. |
| 6 | Új páciens felvétele | Új páciens felvételének elindítása. |
| 7 | Páciens Törlése | A 3. pont szerint kiválasztott személy törlése. |

#### Páciensek felvétele

Ezen a felületen van lehetősége új pacienst felvennie a rendszerbe. Kérem vegye figyelembe hogy a legtöbb adat megadása kötelező és figyeljen arra hogy minden rubrikába a megfelelő adat kerüljön. Az adatok helyes kitöltését követően a „Mentés” gombbal elmentheti, míg a „Vissza” gombbal mentés nélkül törlődnek a bevitt adatok.

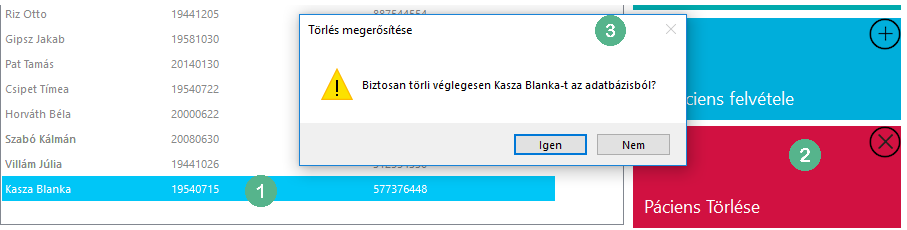
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *(\*a csillaggal jelölt adatok kitöltése nem kötelező)* Maximási karakterszám: | | | | |
| 1 | Vezetéknév | Szöveges. Csak betűt és szóközöket tartalmazhat. | Össz.:54 kar. |
| Keresztnév | Szöveges. Csak betűt és szóközöket tartalmazhat. |
| 2 | Irányítószám | Szám. Csak 1111-9999 közötti számot tartalmazhat. | Össz.:255 kar. |
| Település | Szöveges. Település vagy város neve. |
| Utca | Szöveges. közterület neve és típusa (pl: utca, tér,) |
| Házszám | Szöveges. Betűt, számot és írásjelet is tartalmazhat. |
| 3 | Születési hely | Szöveges. Település vagy város neve. | Max.: 54 kar. |
| Születési név\* | Szöveges. Amennyiben nincs nem kell kitölteni. | Max.: 52 kar. |
| Anyja neve | Szöveges. Csak betűt és szóközöket tartalmazhat. | Max.: 52 kar. |
| 4 | Születési idő | Naptárból kiválasztható dátum. Hónapra kattintva, évek, majd évtizedekre váltva a pontos időpont kiválasztható. |  |
| 5 | TB-szám | Szám mező. Egyedi társadalom biztosítási azonosító. | Max.: 9 kar. |
| Telefonszám | Szám mező. kötőjelet, szóközt, perjelet is tartalmazhat. | Max.: 20 kar. |
| E-mail cím\* | Speciális mező. Szöveg és speciális jeleket is tartalmazhat. | Max.: 54 kar. |
| 6 | Megjegyzés\* | Szöveges mező. Bármilyen szöveges adatot tárolhat. | Max.: 2M kar. |
| 7 | Mentés gomb | Kitöltött adatot esetén elmenti az új pácienst. |  |
| 8 | Vissza gomb | Mentés nélkül visszatér a fő ablakra. |  |

#### Páciens adatainak módosítása

Ezen a felületen a már felvett páciensek adatait tekintheti meg illetve módosíthatja azokat. A megnyitást követően a lap megtekintés módba kerül amennyiben adatokat szeretne módosítani kérem kattintson a (12.) a kék „Szerkesztés” gombra. Ez esetben a „Szerkesztés” gomb átvált zöld „Mentés” gombra, amire kattintás hatására módosul az adatlap. Mint ahogy a páciens létrehozásánál, itt is figyeljen arra hogy a „nem kötelező” mezők kivételével mind legyen kitöltve, az adatlap mentése csak ez esetben lehetséges.

|  |  |  |  |
| --- | --- | --- | --- |
| *(\*a csillaggal jelölt adatok kitöltése nem kötelező)* Maximási karakterszám: | | | |
| 1 | Név | Szöveges. Csak betűt, szóközt és kötőjelet tartalmazhat. | Max. 54 kar. |
| 2 | Lakcím | Szöveges. Alábbi sorrendben (6270 Szeged, Tó út, 11.) | Max. 255 kar. |
| 3 | Születési idő | Szám mező. Születési idő (év,hó,nap), kötőjel szünet nélkül. | Max. 8 kar. |
| 4 | Születési hely | Szöveges. Település vagy város neve. | Max. 54 kar. |
| 5 | Születési név \* | Szöveges. Amennyiben nincs nem kell kitölteni. | Max. 52 kar. |
| 6 | Anyja neve | Szöveges. Csak betűt és szóközöket tartalmazhat. | Max. 52 kar. |
| 7 | TB-szám | Szám mező. Egyedi társadalom biztosítási azonosító. | Max. 9 kar. |
| 8 | Telefonszám | Szám mező. kötőjelet, szóközt, perjelet is tartalmazhat. | Max. 20 kar. |
| 9 | E-mail cím\* | Speciális mező. Szöveg és speciális jeleket is tartalmazhat. | Max. 54 kar. |
| 10 | Megjegyzés\* | Szöveges. Bármilyen szöveges adatot tárolhat. | Max. 2M kar. |
| 11 | Vissza | Mentés nélkül visszatér a fő ablakra. |  |
| 12 | Szerkesztés | Minden mező szerkeszthetővé válik |  |
| Mentés | Helyesen kitöltött adatok esetén elmenti azokat. |  |

#### Páciens törlése

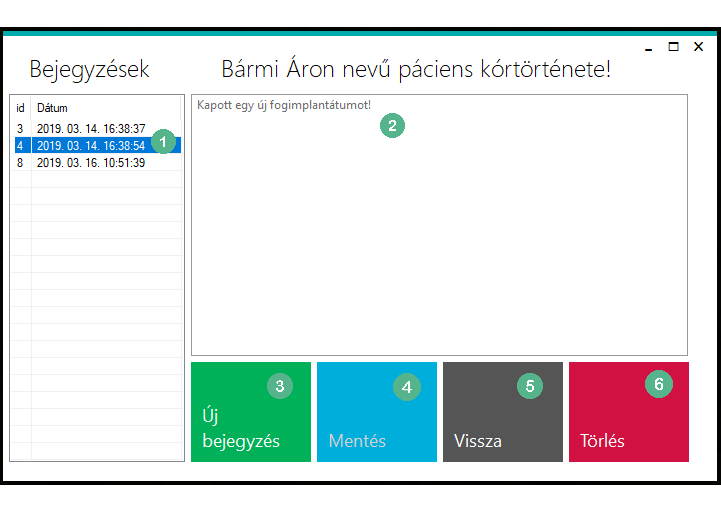
Ezzel a funkcióval van lehetőség egy már regisztrált páciens törlésére. Törlés elött a program megerősítést kér. FIGYELEM! A törlés után a páciens és minden hozzá kapcsolódó anyag végérvényesen törlődik. Törlés folyamata:

1. kiválasztja a törölni kívánt pácienst a listából

2. Rákattint a „Törlés” gombra

3. Ha biztos benne hogy törölni kívánja a kiválasztott pácienst rákattint az „Igen”, ellenkező esetben a „Nem” gombra.

#### Páciens bejegyzései / Kórtörténet

Ezen a felületen van lehetőség megtekinteni a paciensekhez íródott bejegyzéseket / kórtörténetet. Az ***1. jelölésű*** listában időrend szerint csökkenő (címsorra kattintva növekvő) sorrendben megtekintheti az eddig bejegyzett összes látogatást. És kattintással megnyithatja az. Megnyitva a kiválasztott bejegyzés megjelenítődig ***a 2. jelölésű*** mezőben. A ***3. jelölésű*** „Új bejegyzés” gomb új bejegyzést készítését tehetővé ilyenkor az a ***2. mező*** kiürül és már írhat is bele. A szöveg bármit tartalmazhat maximás hossza akár több oldal is lehet. Amikor elkészült a bejegyzés szövege, a ***4. jelölésű*** „Mentés” gombbal elmentheti azt. Az ***5. jelölésű*** „Vissza” gombbal visszatérhet a főablakba, míg a ***6. jelölésű*** „Törlés” gombbal az épp kiválasztott bejegyzés törlődik egy megerősítő ablakot követően.

### Webes alkalmazás (a páciensek részére)

A FOR weboldala kifejezetten a rendelőben előzetesen regisztrált ügyfelek részére készült, célja a páciensek részére a saját kórtörténetik, fogorvosi látogatásaik alatt készített feljegyzések megtekintése, ellenőrzése. Saját adataikat megtekinthetik és bizonyos mértékig megadhatják vagy éppen frissíthetik.

FIGYELEM! A páciens az alapértelmezett felhasználóinevét és jelszavát a rendelőbe történő regisztráció után kapja meg! Kérjük az első bejelentkezést követően a „Bejelentkezési Adatok módosítása” menüpont alatt változtassák meg gyári felhasználói nevüket és jelszavukat.

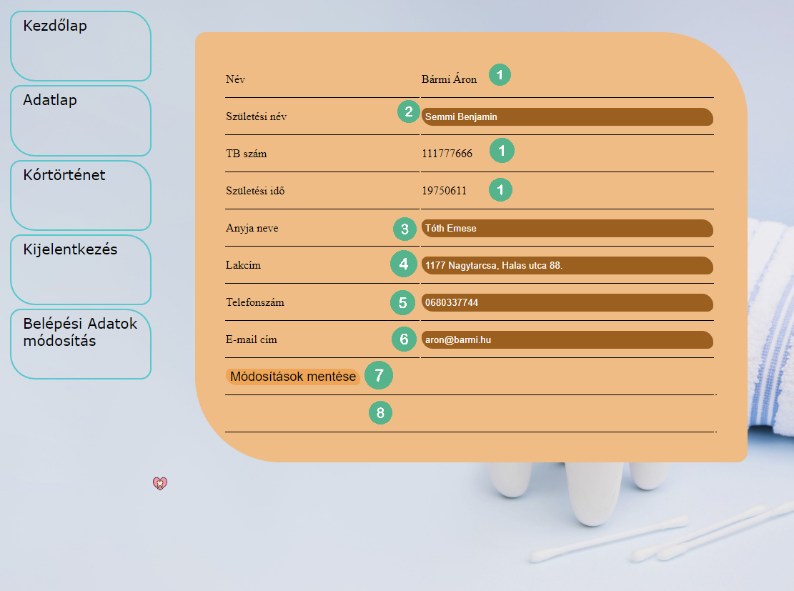
***Az ön gyári felhasználóineve: az ön TB-száma egybe írva szünet/kötőjel nélkül pl.:111222333***

***Az ön gyári jelszava: az ön születési ideje, egybe írva szünetés egy írásjel nélkül pl.:19990101***

#### Menü / Belépés

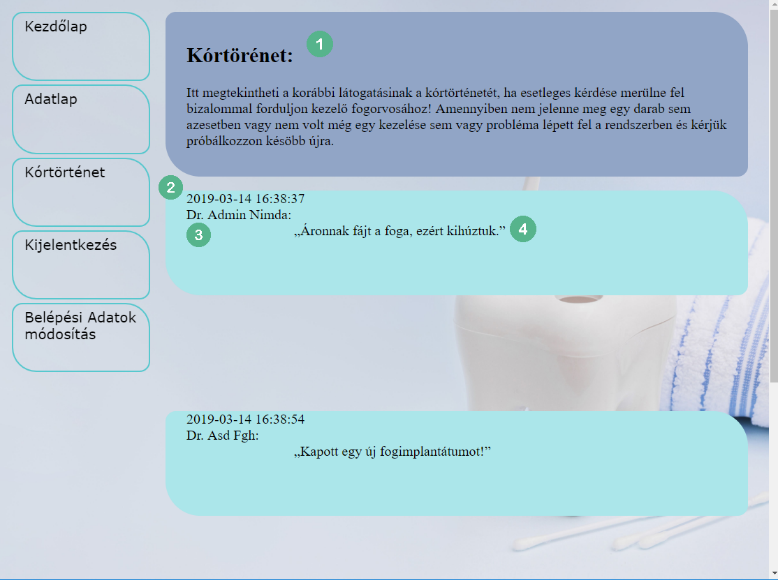
A weblap menüje és a felhasználó belépése végig minden lapon elérhető. ***1. menüpont***-on segítségével bármikor az oldal kezdőlapjára kerülhetnek ahol hasznos információkat olvashatnak a saját rendelőjük, orvosukról. A ***2. menüpont***-on bejelentkezést követően megtekinthetik a saját adataikat és részlegesen módosíthatják azokat. A ***3. menüpont***-on megnézhetik kórtörténetüket, visszatekinthetnek a korábbi látogatásikra. A ***4. menüpont*** segítségével lenyílnak a bejelentkezéshez szükséges mezők. Az ***5. rubrika***-ba a felhasználói nevet, a ***6. rubrika***-ba a jelszavat és a ***7. menüpont***-ra kattintva be is jelentkezett a páciens.

#### Adatok megtekintése

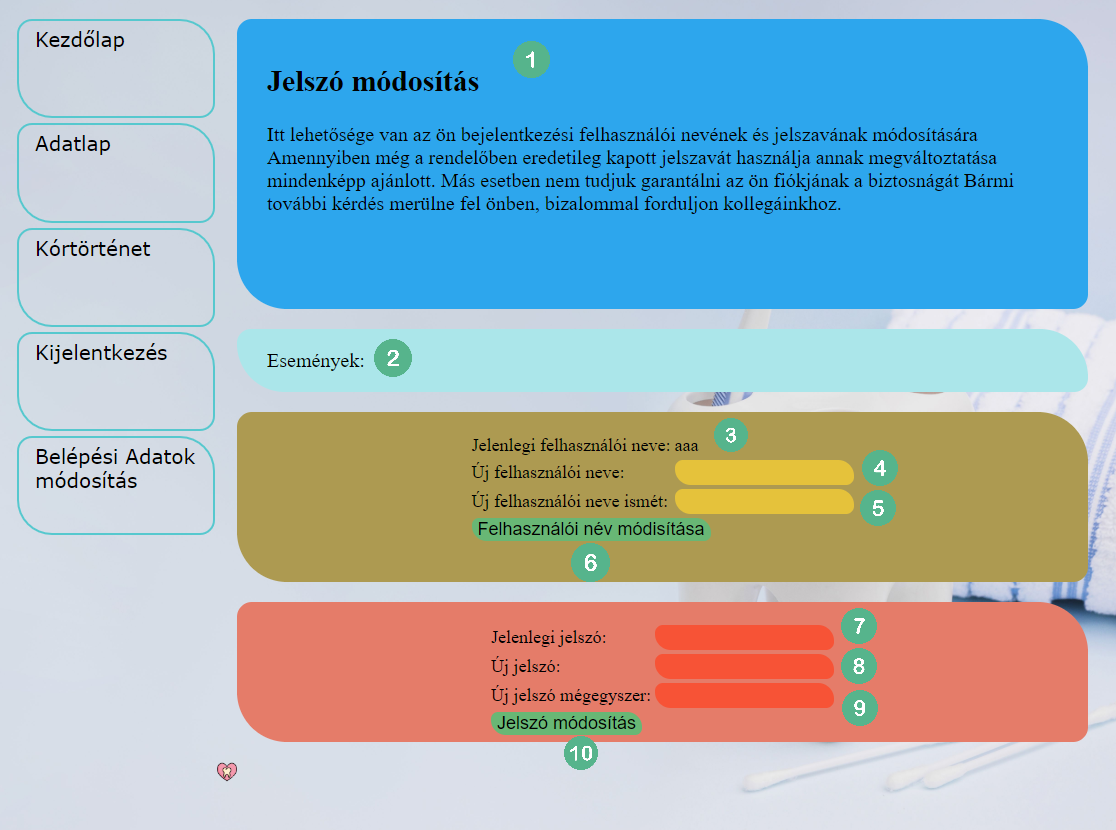
Az adatlap menüponton bejelentkezést követően lehetősége van megtekinteni az ön részletes adatlapját, minden adatot amit a rendelőben önről tárolnak. Ezek a három alap adaton (***1.pont***) kívül, módosíthatóik az alábbiak szerint. ***2.pont***: az ön születési neve (amennyiben nincs nem szükséges megadni), ***3.pont***: édesanyja születési neve, ***4.pont***: a pontos lakcíme (pl.: 6720 Szeged, Zár utca 11.), ***5.pont***: az ön telefonszáma, ***6.pont***: az ön e-mail címe.

Az adatok mentése a ***7.gomb*** megnyomásával történik, amennyiben az ön adatai sikeresen megváltoztak, vagy egyik mező hibásan lett kitöltve az esetben a ***8.pont***-nál kap egy üzenetet.

#### Kórtörténet / Bejegyzések megtekintése

A kórtörténet menüpont alatt bejelentkezést követően lehetősége van megtekinteni az ön kórtörténetét, a látogatásai alatt készült feljegyzéseket. Így bármikor napra készen megnézheti legutóbb mikor, ki, milyen panasszal és hogyan is kezelte. Az ***1.pont***-nál egy általános tájékoztatót olvashat. A ***2.pont*** a látogatás pontos időpontját, a ***3.pont*** a bejegyzést készítő orvos nevét, míg a ***4.pont***-ban magát a látogatás dokumentálását olvashatja.

#### Belépési Adatok módosítása

A Belépési Adatok módosítása menüpontban a bejelentkezést követően önnek lehetősége van az ön felhasználói nevének és jelszavának a módosítására. Az ***1.pont***-nál egy általános tájékoztatót olvashat. ***Felhasználói név módosítása:*** A ***3.pont*** az ön jelenlegi felhasználói nevét jeleníti meg, a ***4-5.pont*** mezőjébe adja meg a választott új nevet (eltérőnek kell lennie a jelenlegitől), a ***6.pont***-ra kattintva elmenthite azt. ***Jelszó módosítása:*** A ***7.pont*** mezőjébe adja meg az ön jelenlegi jelszavát, a ***7-8.pont***-ba a választott új jelszavat (eltérőnek kell lennie a jelenlegitől) és végül a ***10.pont***-ra kattintva elmentheti azokat ***2.pont***-nál mind sikeres mind sikertelen adat módosításnál üzenetet kap a folyamat eredményéről.

# Összefoglaló

Az elkészült két frontos alkalmazás most már képes a fogorvos és a páciensek igényeit is kielégíteni. Az orvos a rendelőjében mindig napra kész lehet a beteg adataival kapcsolatban, képes nyomon követni és folytatni a kórtörténetét. Míg a páciens kényelmesen immáron otthonról is megtekintheti a fogorvosa által tárolt személyes adatlapját, annak bizonyos részeit módosítani. Képes a saját látogatási alatt készült bejegyzéseket megtekinteni, ezáltal az egész kórtörténetére rálátása lett.

Ezzel a project első lépését befejezettnek tekintem, kiemelve csak az első részét és azt is leginkább az időm hiánya miatt. Az egész feladat legnagyobb tanulsága hogy ez bizony egy időigényes folyamat, miközben írtam hol magát a programot hol pedig a dokumentációt, mindig eszembe jutott egy új gondolat hogy tehetném jobbá gyorsabbá a programot, mit kéne máshogy esetleg kevesebb kóddal megvalósítani. Ezt több esetben meg is tettem, de a határidő közeledtével ezek egy része ki kellett hogy maradjon. Tervben volt egy sokkal átfogóbb teszt folyamat, egy minden apró részletre kiterjedő ellenőrző réteg integrálása. Egy közös üzenő felület ahol a beteg és az orvos kommunikálhat és egy esetleges határidő napló amiben a minden időpont megtekinthető és személyre szabható. Sok hasznos dolgot tanultam a készítés közben magáról a több rétegű programozásról és úgy hiszem ha elölről kezdeném, egy még összetettebb és egyben még egyszerűbb kódot tudnék létrehozni.

De mint mondtam ezt csak egy első lépésnek, egy első fázisnak tartom ezen a kódon hobbi jelleggel tovább fogok dolgozni és fejleszteni, gyakorolni, kísérletezni fogok vele. Úgy hiszem az adatbázisom kellően összetett és maga a program struktúrája is megfelelő ahhoz hogy a későbbiekben több résszel is kiegészíthessem, nagyobb mennyiségű kód átírása nélkül.

# Mellékletek

|  |  |
| --- | --- |
| |  | | --- | | \*a MySql eljárás:  DELIMITER $$ CREATE DEFINER=`root`@`localhost` PROCEDURE `stafflogin`  (IN `username` VARCHAR(24), IN `password` VARCHAR(24)) NO SQL SELECT COUNT(\*) FROM staff WHERE username=staff.username AND password=staff.password$$DELIMITER ; |   1.melléklet |
| 2.melléklet |

|  |  |
| --- | --- |
| *3.melléklet 4.melléklet* | |
| |  |  | | --- | --- | |  | **Tartalom** | | config | Adatbázis, hozzáférési adatok | | css | Stílus fájlok | | html | HTML kódok feldarabolva | | js | JavaScript kódok | | php | PHP fájlok | | pics | Használt képek, háttér | | ../ | Kezdőlap | |  |

1. Relációs adatmodell, Wikipédia <https://hu.wikipedia.org/wiki/Rel%C3%A1ci%C3%B3s_adatmodell> [↑](#footnote-ref-1)
2. automatikusan növekvő, sql-ben az index-ek legfőbb tulajdonsága [↑](#footnote-ref-2)
3. egy vagy több soros általános beviteli mező [↑](#footnote-ref-3)
4. Meghatározott esemény(egér-, billentyű-akció, paraméter változás) hatására lefutó kód. [↑](#footnote-ref-4)
5. havi lebontású naptár, állítható paraméterekkel és tulajdonságokkal [↑](#footnote-ref-5)
6. dialog box(pl. MessageBox) visszaadási értékét ellenőrző osztály [↑](#footnote-ref-6)
7. osztály ami az előre meghatározott karaktereket egy másikra cserél ki egy string-ben. [↑](#footnote-ref-7)
8. Egymásba ültetett, közvetlenül több rétegü [↑](#footnote-ref-8)
9. kriptográf titkosítási algoritmus [↑](#footnote-ref-9)
10. boolean- logikai két értékű változó [↑](#footnote-ref-10)
11. kivételkezelő eljárás [↑](#footnote-ref-11)
12. php nyitó „<?php” és zárótag „?>” [↑](#footnote-ref-12)
13. munkamenet megnyitó funkció, oldalak között tárolt adatok elérésére és adatok tárolására. [↑](#footnote-ref-13)
14. super global tömb ami a session\_start()-ot követően folyamatos tárolásra alkalmas, a munkamenet bezárásáig ’session\_destroy()’ vagy az adattag törléséig ’unset(…)’ [↑](#footnote-ref-14)
15. MySQL kezelésért felelős PHP osztály (PHP 5, PHP 7) <http://php.net/manual/en/class.mysqli-stmt.php> [↑](#footnote-ref-15)
16. elöl tisztelős ciklus, amíg a feltétel igaz a ciklusmag újra és újra lefut, előfordulhat hogy egyszer sem fut le. [↑](#footnote-ref-16)
17. paragrafus, alap bejegyzés típus a html-ben [↑](#footnote-ref-17)