Bálint István

Istvann.balint@gmail.com

Szeged, 2020

Szegedi Szakképzési Centrum Vasvári Pál Gazdasági és Informatikai Szakgimnáziuma  
Az 54 213 05 számú Szoftverfejlesztő szakképesítés záródolgozata

Live In Care

szoftver



# Tartalomjegyzék:

Tartalom

[1. Tartalomjegyzék: 1](#_Toc37158212)

[2. Bevezetés: 3](#_Toc37158213)

[3. Fejlesztői dokumentáció: 5](#_Toc37158214)

[I. Adatbázis 5](#_Toc37158215)

[a. Egyedek: 5](#_Toc37158216)

[b. Belső tulajdonságok: 6](#_Toc37158217)

[c. Egyedek, tulajdonságok típussal és névvel: 6](#_Toc37158218)

[d. Külső tulajdonságok: 9](#_Toc37158219)

[e. Összekapcsolásuk: 9](#_Toc37158220)

[II. Szoftver: 10](#_Toc37158221)

[a. Felhasználok: 12](#_Toc37158222)

[b. Gépigény: 13](#_Toc37158223)

[c. Interjúk: 13](#_Toc37158224)

[d. Tervezés 14](#_Toc37158225)

[e. Bejelentkezés: 16](#_Toc37158226)

[f. Felépítése, mappa szerkezet 19](#_Toc37158227)

[g. Hozzáadás, módosítás, törlés: 19](#_Toc37158228)

[h. Hibakezelés: 26](#_Toc37158229)

[i. Fejlesztési lehetőség 28](#_Toc37158230)

[III. A webalkalmazás 28](#_Toc37158231)

[a. Bevezetés 28](#_Toc37158232)

[b. Felhasználok 29](#_Toc37158233)

[4. Melléklet: 30](#_Toc37158234)

[I. E-K diagramm (I. ábra) 30](#_Toc37158235)

[II. Bachmann-ábra (II. ábra) 31](#_Toc37158236)

[III. Use-Case diagramm (III. ábra) 31](#_Toc37158237)

[IV. Gyerek regisztrációja 32](#_Toc37158238)

# Bevezetés:

A záródolgozatom egy olyan szoftvert hivatott megvalósítani, ami az állami nevelésben élő gyerekek nevelkedését segíti elő. Valamint az adott intézményben a dolgozók munkáját teszi gördülékenyebbé. Emellett megkönnyíti a rendszerben lévő felhasználok (lásd: Fejlesztői dokumentáció -> Felhasználok) információ áramlását.

Fontos emellett megemlíteni a más problémákat is. Papír alapon dolgoznak nagy részt az intézmények ilyen szférában különböző dokumentáció elkészítésére. Ezért ez a szoftver fel tudja váltani a papírt, hiszen így elektronikusan tud működni az ügyintézés. Miért nem előnyős a papír használata ilyen szférában? Mert természetesen sok időt vesz igénybe az egyes nyomtatványok kitöltése, és ezeket iktatni is kell. Kitöltéssel is lehetnek problémák, amit például az Intézmény ügyintéző nem tud kiolvasni, így vagy rossz vagy hiányos információ lesz a gyerekről. Problémát okoz, akár egy dokumentum elő keresése, mert azt egy nagy irattartó mappában tartják gyerekeként. Természetesem az is előfordult már, hogy nem lett meg az adott dokumentum, mert elveszett. Egyik célja, hogy modernizáljuk az intézmény telekommunikációs infrastruktúráját (felek közti gyors információ átadását). Emellett gyorsabb dokumentum előkeresést, vagy akár adott személyes adatt meg keresése a programban egy adott fiatalról. Így az egyes ügyintézések is gyorsabban tudnak végre hatódni. Mert ha csak egy státusz változás[[1]](#footnote-1) veszünk figyelembe, az több hónapot is igénybe vehet. Fiatalnak várnia kell, amit már tud, hogy változni fog. Több problémát is felvett, akár a gyerek nem tudja gyorsan az adott helyzetet / múltat lezárni, vagy fontos a gyors áthelyezés egyéb okok miatt[[2]](#footnote-2).

Úgy gondolom ahhoz, hogy a gyerek fejlesztése szóba jöjjön, ahhoz természetesen mindent tudni kell a gondozott fiatalról. Hiszen anélkül nem tudhatjuk, milyen állapotban van a gyerek. Milyen lelki vagy fizikai sérülése van, ha van. Hiszen a szülőktől való leválasztás folyamata[[3]](#footnote-3) igen nehéz tud lenni minden fiatal számára. Ezért is kulcs fontoságú a fiatal nyomon követése és szoros együtt működésé ebbe a szülőkkel. Ezzel a program csomaggal őket is szeretnék támogatni, informálni a saját gyerekéről.

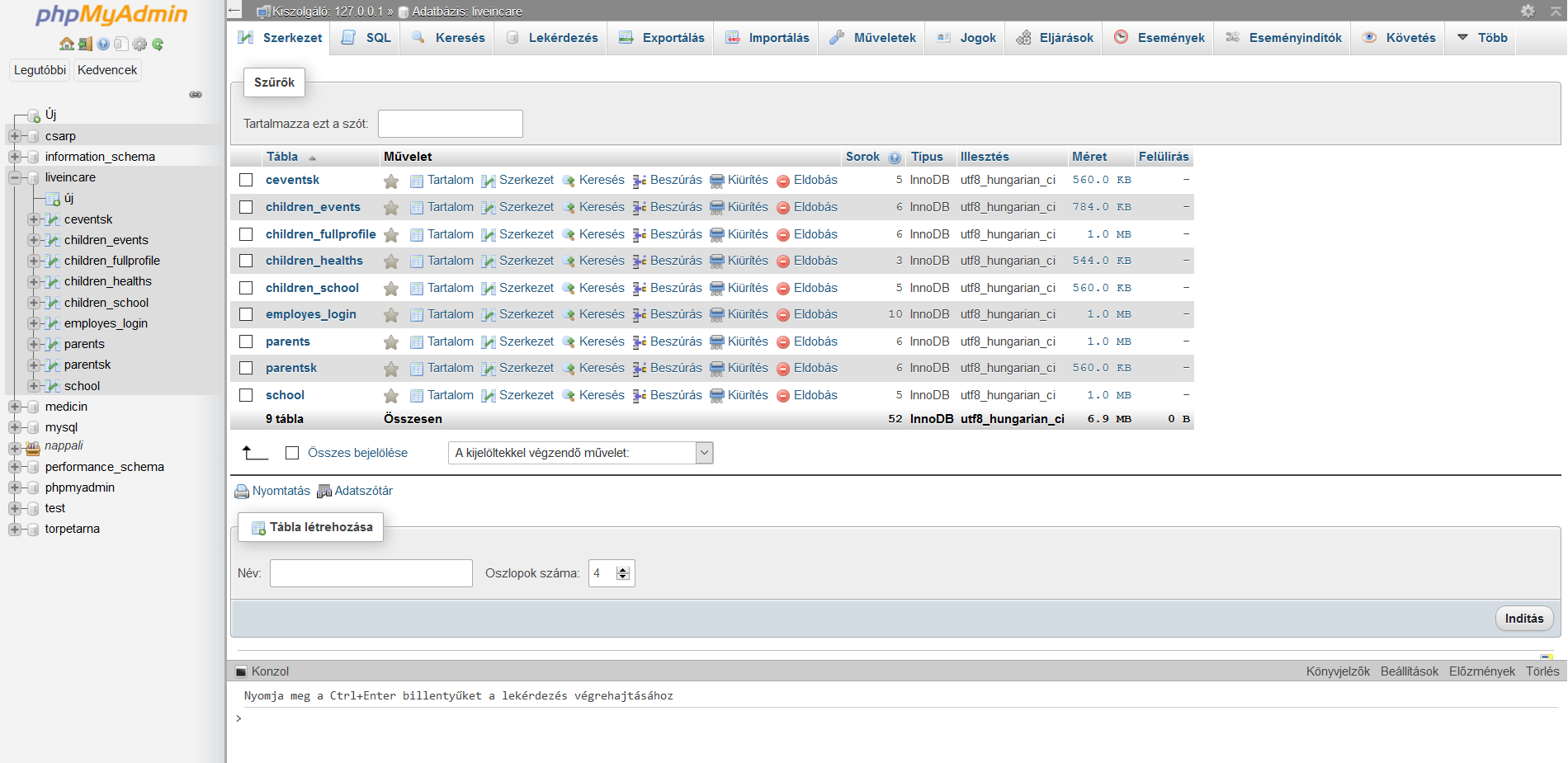
Tapasztalataim alapján is mondhatom azt nem feltétlen figyelnek oda a gyerek egyes nehézségeire (SNI, BTM) egyes intézményekben. Hanem átlagos gyereként kezelik. Itt viszont a bejelentkezés mellett, gyors információ áramlás lehetséges a dolgozok között. Így a nevelő oda tud figyelni, amit a Pszichológus tanácsol, vagy egyéb más vizsgálatokról szerezhet információt.  
  
Az előbbi okok folytán határoztam el, hogy ezt a munkát segíteni kell egy webes és egy desktop felülettel. Szerepkörök alapján lesz a beléptetés, minden szerepkör egy adott részhez nyúlhat / módosíthat. Ez szereptől függ. Nagy részt adatok tárolása lesz a célja. De amint említettem a nevelőt segíteni fogja, ha meg tudja időben, hogy milyen problémái lehetnek a gyereknek, hiszen a Pszichológus szakvéleményt ír, amire a nevelő azonnal rá is tud nézni a szoftver segítségével. Fontos része lesz a szoftvernek, hogy a szülök is meg tudjanak a gyerekről információkat, hogy merre járt, milyen sport eredményeket ért el, milyen egészségügyi vizsgálatokon vett részt. Hiszen a szülő elsődleges célja alap esettbe, hogy tudja magától nevelni a gyereket, vagyis kihozza a fiatalt az intézmény falai közül, ehhez egy alapot kell megteremtenie, ami sok munkával járhat. Fontos megejteni, hogy ez is beléptetés alapján lesz, és csak is akkor léphet be a szülő, ha ehhez a gyerek 100%-osan hozzá járult. Természtesen az átláthatóság is szerepet játszik a szoftver létrehozásában. Jobban meg tudják vizsgálni melyik fiatal pontosan hol van (melyik területi intézményben).

# Fejlesztői dokumentáció:

Egy szoftver megtervezése során fontos, szempont a programtervezés folyamata, ami több részből tevődik ki. Adottság, lehetőség, korlát, elemzés, szerződés, részletes terv, kódolás, tesztelés, program bevezetés és átadás. Nekünk most első négy szempont és a részletes terv, valamint a kódolás, amik fontosak. Felkel mérnünk, hogy milyen adottsággal rendelkezi az adott intézmény. Milyen lehetőségek vannak (pl.: több intézmény van egy helyen), és az egyik legfontosabb milyen korlátok vannak (pl.: puszta közepén van az intézmény, rossz internet kapcsolat). Elemzés, hogy milyen a jelenlegi rendszer adottságai (fejlesztés lehetőség). Részletes terv pedig a leírása annak, mi szükséges az adott szoftverhez.

## Adatbázis

Adatbázis szerves részét képezi egy szoftvernek, hiszen itt tárlódnak az adatok és innen lehet lekérdezni az adatokat. A tanulmányaim során a phpMyAdmin (MySQL) adatbázis kezelő felületet használtam, ennek okén fogva a szakdolgozatomba is ezt alkalmazom. Szükségem volt egy virtuális saját szervere és ehhez kellett a XAMPP nevezetű program (*0. ábra*). Virtuális serveren „találkozik” a weblap és az asztali alkalmazásunk. Fontos előre megadni az adatbázisunk irány vonalait. És azzal kezdeni. Én is így terveztem meg szoftver és a weblapomat. Nyilván ebben a későbbiekben változtattam, ahogy a funkciókat programoztam.



*0. ábra: EK-diagramm*

### Egyedek:

Az egyed az, amit le akarunk írni, amelynek az adatait tároljuk és gyűjtjük az adatbázisban. Az egyedet idegen szóval entitásnak nevezzük. Egyednek tekinthetünk például egy személyt. Fontos meghatározni milyen adatokra van szükségünk, nekem ezek az interjúk alatt derültek ki, mi az ami nagyon fontos adat a gyerekről, vagy más egyedekről.

* Gyerekek teljes profilja (**children\_fullprofile**)
* Szülők, ahova felregisztrálják őket (**parents**)
* Gyerek egészségügyi vizsgálatai és pszichológiai vizsgálata eredményét itt tároljuk (**children\_healths**)
* Gyerekkel történt események szülőknek, gyerek hozzárendelés nélkül (**children\_events**)
* Iskolai intézmények legfontosabb adatai ide regisztrálhatjuk fel. (**school**)
* Dolgozok adatai és belepésének adatai (**employers\_login**)
* Kapcsoló táblák
  + Esemény és a gyerek teljes profilját kapcsolja össze (**ceventsk**)
  + A gyerekek összekapcsolása szülőkkel, vagyis melyik gyereknek kik a szülei (**parentsk**)
  + Gyerek adott oktatási jogviszonya egy intézménnyel (**children\_school)**

### Belső tulajdonságok:

Egy adatmodellben valamennyi egyedet egyenként véges számú tulajdonsággal írunk le. Ezek a tulajdonságok együttesen alkotják az egyed belső szerkezetét, és az egyed belső tulajdonságainak hívjuk.

* **children\_fullprofile** (ID, cname, csex, cidcardnumber, ctajnumber, cbirth, cbirthplace, ccoming, clocation)
* **parents** (ID, pname, pbirth, psex, pidcardnumber, loginpermission, loginuser , loginpsw, reg)
* **children\_events** (ID, title, details, img, by)
* **children\_healths** (ID, childrenID, type, details, special\_treatment,treatdate, by)
* **children\_school** (ID, schoolID, childrenID, fromDate, expectedFinish,type, headteacher)
* **employes\_login** (ID, ename, emaidenname, esex, ebirth, ebirthplace, ejob, elocation, idcard, euname, epassword)
* **ceventsk** (ID, childrenID, eventsID, timer)
* **parentsk** (ID, pID, childrenID)
* **school** (ID, schoolname, schoolLocation, schoolPhone)

### Egyedek, tulajdonságok típussal és névvel:

* **children\_fullprofile (gyerek tábla):**
  + **ID** (egész szám, maximum 11 hosszú, elsődleges kulcs), gyerek azonosító
  + cname (szöveg, 50 hosszú karaktersorozat), gyerek neve
  + csex (szöveg, 10 hosszú karaktersorozat), gyerek neme
  + cidcardnumber (szöveg, 10 hosszú karaktersorozat), gyerek személyigazolványszáma
  + tajnumber (szöveg, 10 hosszú karaktersorozat), gyerek TAJ száma
  + cbirth (dátum), születési ideje
  + cbirthplace (szöveg, 40 hosszú karaktersorozat), születési helye
  + ccoming (dátum), intézménybe belépés ideje
  + clocation (szöveg, 40 hosszú karaktersorozat), intézmény helye, ahol lakik
* **parents (szülő tábla):**
  + **ID** (egész szám, maximum 11 hosszú, elsődleges kulcs), szülő azonosító
  + pname (szöveg, 40 hosszú karaktersorozat), szülő neve
  + pbirth (dátum), szülő születési ideje
  + psex (szöveg, 10 hosszú karaktersorozat), szülő neme
  + pidcardnumber (szöveg, 10 hosszú karaktersorozat), szülő személyigazolványszám
  + loginpermission (1 bites egész szám), a regisztrációhoz engedély
  + loginuser (szöveg, 25 hosszú karaktersorozat), regisztrációs felhasználó
  + loginpsw (szöveg, 25 hosszú karaktersorozat), regisztrációs jelszó
  + reg (1 bites egész szám), regisztrálást ellenőrző (1 = volt)
* **children\_events (esemény tábla):**
  + **ID** (egész szám, maximum 11 hosszú, elsődleges kulcs), esemény azonosító
  + title (szöveg, 30 hosszú karaktersorozat), az esemény címe
  + details (szöveg, 120 hosszú karaktersorozat), az esemény leírása
  + img (kép), kép eseményről
  + by (szöveg, 20 hosszú karaktersorozat), felvevő ember
* **children\_healths (vizsgálati tábla):**
  + **ID** (egész szám, maximum 11 hosszú, elsődleges kulcs), vizsgálat azonosító
  + childrenID (egész szám, maximum 11 hosszú, idegen kulcs), gyerek azonosító
  + type (szöveg, 30 hosszú karaktersorozat), vizsgálat típusa
  + details (szöveg, 120 hosszú karaktersorozat), vizsgálat leírása
  + special\_treatment (szöveg, 50 hosszú karaktersorozat), különleges bánásmód
  + treatdate (dátum), vizsgálat ideje
  + by (szöveg, 30 hosszú karaktersorozat), orvos neve
* **employes\_login (dolgozok táblája):**
  + **ID** (egész szám, maximum 11 hosszú, elsődleges kulcs), vizsgálat azonosító
  + ename (szöveg, 50 hosszú karaktersorozat), dolgozó neve
  + emaidenname (szöveg, 50 hosszú karaktersorozat), dolgozó lánykori neve
  + esex (szöveg, 10 hosszú karaktersorozat), dolgozó neme
  + idcard (szöveg, 10 hosszú karaktersorozat), személyigazolványszáma
  + ebirth (dátum), születési idő
  + ebirthplace (szöveg, 40 hosszú karaktersorozat), születési hely
  + ejob (szöveg, 18 hosszú karaktersorozat), beosztás bejelentkezéshez
  + elocation (szöveg, 25 hosszú karaktersorozat), munka helyszíne
  + euname (szöveg, 25 hosszú karaktersorozat), felhasználó név
  + epasword (szöveg, 25 hosszú karaktersorozat), jelszó
* **children\_school (gyerek, amelyik iskolába jár):**
  + **ID** (egész szám, maximum 11 hosszú, elsődleges kulcs), gyerek oktatási jog azonosító
  + schoolID (egész szám, maximum 11 hosszú, idegen kulcs), iskola azonosító
  + childrenID (egész szám, maximum 11 hosszú, idegen kulcs), gyerek azonosító
  + fromDate (dátum), iskola kezdés időpontja
  + exceptedFinish (dátum), iskola várható befejezés időpontja / befejezett suli időpont
  + type (szöveg, 25 hosszú karaktersorozat), típusa
  + headteacher (szöveg, 50 hosszú karaktersorozat), osztályfőnök neve
* **school (oktatási intézmények):**
  + **ID** (egész szám, maximum 11 hosszú, elsődleges kulcs), oktatási intézmény azonosító
  + schoolName (szöveg, 80 hosszú karaktersorozat), oktatási intézmény neve
  + schoolLocation (szöveg, 40 hosszú karaktersorozat), oktatási intézmény helyszíne
  + schoolPhone (szöveg, 14 hosszú karaktersorozat), oktatási intézmény telefonszám
* **ceventsk (esemény gyerekhez kapcsolás):**
  + **ID** (egész szám, maximum 11 hosszú, elsődleges kulcs), esemény- gyerek azonosító
  + childrenID (egész szám, maximum 11 hosszú, idegen kulcs), gyerek azonosító
  + eventsID (egész szám, maximum 11 hosszú, idegen kulcs), esemény azonosító
  + timer (dátum), esemény idópontja
* **parentsk (gyerek szülőhöz kötése):**
  + **ID** (egész szám, maximum 11 hosszú, elsődleges kulcs), esemény- gyerek azonosító
  + pID (egész szám, maximum 11 hosszú, idegen kulcs), szülő azonosító
  + childrenID (egész szám, maximum 11 hosszú, idegen kulcs), gyerek azonosító

### Külső tulajdonságok:

Az egyedek kapcsolatainak összességét külső tulajdonságoknak hívjuk**.**

* Az egyedek kapcsolatait egy nagyon egyszerű ábrával szemléltethetjük (*I. ábra*). Itt láthatjuk, milyen viszony alapján csatlakoznak az egyes táblák egymáshoz.

*Mellékeltem a lap alján, mert nagyobb fél oldalnál.*

*I. ábra: EK-diagramm*

* *II. ábrán* már azt is pontosan láthatjuk, hogy melyik cellák vannak pontosan összekötve, vagy idegen kulcsként átadva az adatbázisomba.

*Mellékeltem a lap alján, mert nagyobb fél oldalnál.*

*II. ábra: Bachmann ábra*

### Összekapcsolásuk:

* ALTER TABLE `**children\_school\_ibfk\_1**` ADD CONSTRAINT `children\_school\_ibfk\_1` FOREIGN KEY IF NOT EXISTS (`schoolID`) REFERENCES `school` (`ID`);
* ADD CONSTRAINT `**children\_school\_ibfk\_2**` FOREIGN KEY IF NOT EXISTS (`childrenID`) REFERENCES `children\_fullprofile` (`ID`);
* ADD CONSTRAINT `**ceventsk\_ibfk\_1**` FOREIGN KEY IF NOT EXISTS (`childrenID`) REFERENCES `children\_fullprofile` (`ID`);
* ADD CONSTRAINT `**ceventsk\_ibfk\_2**` FOREIGN KEY IF NOT EXISTS (`eventsID`) REFERENCES `children\_events` (`ID`);
* ADD CONSTRAINT `**parentsk\_ibfk\_1**` FOREIGN KEY IF NOT EXISTS(`childrenID`) REFERENCES `children\_fullprofile` (`ID`);
* ADD CONSTRAINT `**parentsk\_ibfk\_2**` FOREIGN KEY IF NOT EXISTS(`pID`) REFERENCES `parents` (`ID`);
* ADD CONSTRAINT `**children\_health\_ibfk\_1**` FOREIGN KEY IF NOT EXISTS (`childrenID`) REFERENCES `children\_fullprofile` (`ID`);

## Szoftver

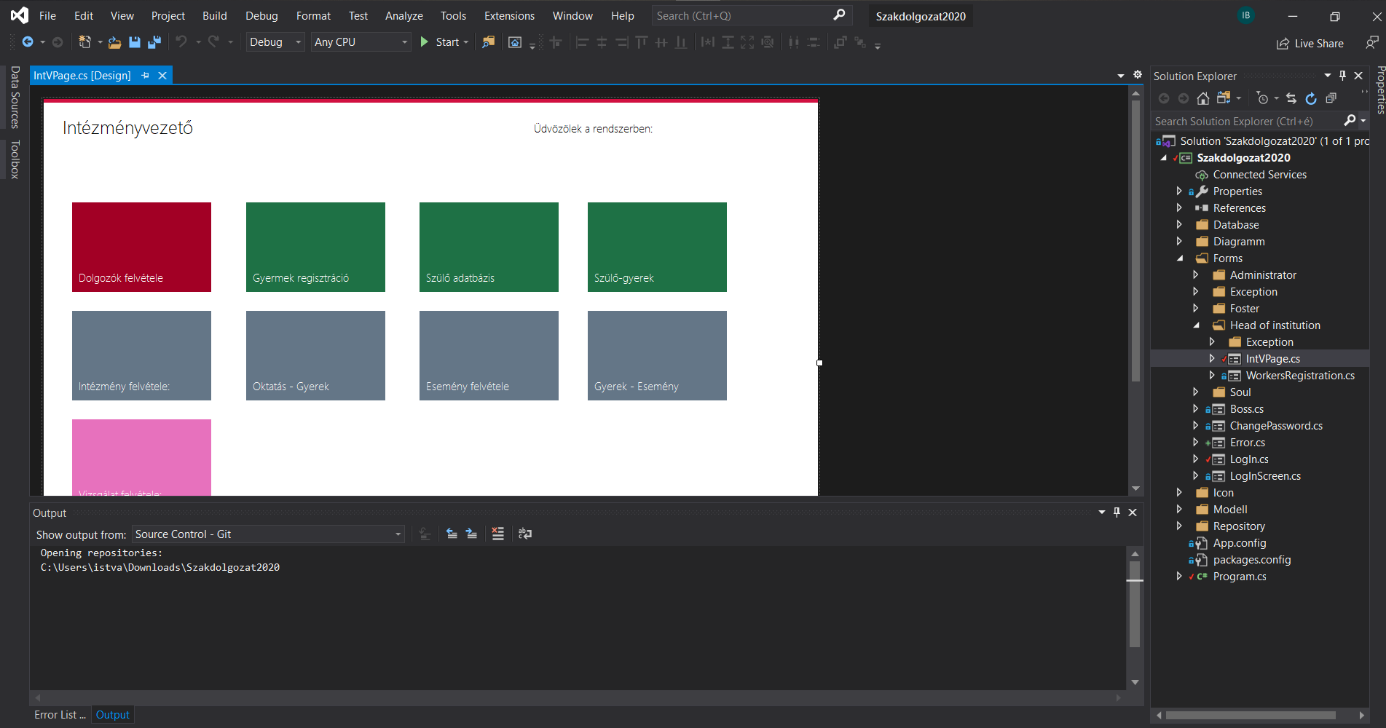
Az asztali alkalmazáshoz, ki kell találni elsőként, hogy pontosan mit szeretnék csinálni. Hiszen anélkül nehezen lehet előre haladni. Fontosnak tartottam, hogy a dolgozok közötti információ áramlás gyorsabb legyen, mint eddig volt. Fontos mérföldkőnek gondolom azt, hogy ha egy pszichológus csinál egy felmérést egy gyerekről az már a bekerülés legelején történjen meg és a nevelő, a szoftver segítségével tudja ezt meg. Hogy a gyerek lelki dolgaira is oda tudjon figyelni. Gyakran érzékeltem, hogy ez nem tartják fontosnak.

Ennek több rétege is van: a felhasználok azonosítása (kiléphet be), milyen feladatokat lásson el (általában ezt a megrendelővel való egyeztetés során derül ki, és szoros együtt működéssel történik, vagy az interjú segítségével) vagyis a funkcionális[[4]](#footnote-4) és minőségi követelményeket[[5]](#footnote-5) kellett szem előtt tartani.

Az én választott fejlesztési felületem az a Microsoft Visual Studio 2019 (*III. ábra*).

Nagy előnye még a Microoft Visual Studio-nak, hogy több verziókövető rendszert[[6]](#footnote-6) is hozzá lehet kötni, én a GitHub-t alkalmaztam a szakdolgozatomba, de emellett lehet Azure-t is használni.

Azért választottam ezt, mert tisztába vagyok a kezelésével és természetesen ingyenes (2019 kiadás). Nagyon sok framware[[7]](#footnote-7) -t lehet hozzá letölteni. Én a Metro framwork-t használom, mint keretrendszer. Fontos megemlíteni, hogy a programozási nyelvem C# lett, mert az iskolai tanulmányaim alatt ezt tanultuk meg olyan szinten, hogy tudjam alkalmazni egy magasabb szinten is. Pozitívnak számít még, ennél a fejlesztési környezetnél, hogy személyre szabható. Itt akár gondolhatunk az egyes ablakok elrendezésére (például Team Explorer, Properties, Solution) és vagy a ’Dark’ módra, ahol a szemünknek is kevesebbet árt és több ideig tudjuk használni egy nap leforgássá alatt. 2019-es verzióban már több szín is elősegítik a munka gyorsabb, rendezett előre haladását. Valamint a referenciák elárulják a fejlesztőknek, hol lett használva az adott függvény, metódus. Ez platformhoz kötött, vagyis Microsoft Windows operációs rendszer szükséges hozzá. De van példa platform független programozási nyelvekre, is például Java.



*III. ábra: Microsoft Visual Studio 2019 felülette*

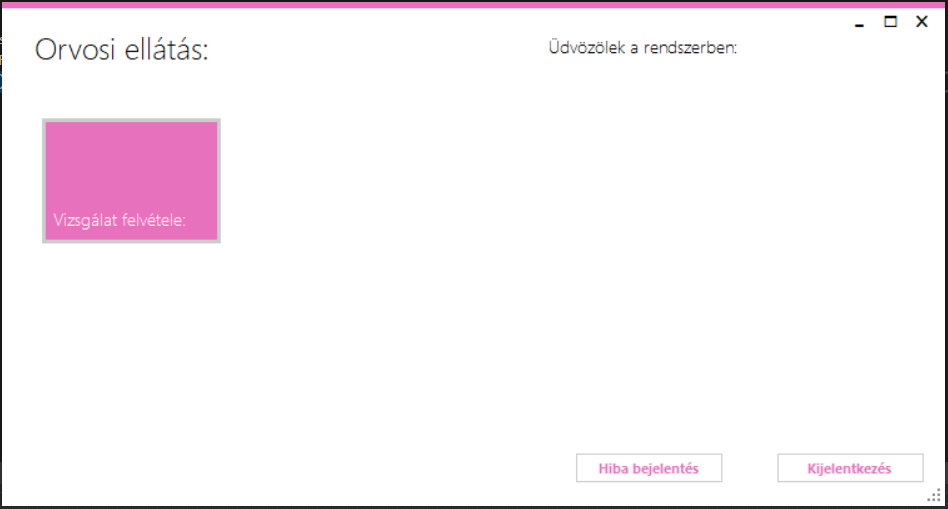
Továbbá a szoftvernek midig át kell esnie egy tesztelés folyamaton is. Így ki tudjuk az egyes gyenge pontokat szűrni. Az egyes modelleken belül adatok bevitele, módosítása, beolvasása tesztelve van, hogy nem kerüljön olyan adatt az adatbázisba, ami nem belevaló. Egyéb esettben.

Elsődleges cél volt az is, hogy úgy tervezem meg a programot minél felhasználó barátabb, de emellett esztétikus is legyen. Ez azt takarja, hogy intézményben dolgozó emberekkel szoros együtt működés és igények meghallgatása mellett terveztem és alakítottam ki a végső megoldást. Követelményfeltáráshoz elengedhetetlen a leendő felhasználok megkérdezése. Ez által elérve azt, hogy minél egyszerűben meg lehessen tanulni. Így az intézményben dolgozóknak nem kell félni egy újabb program megtanulásától. Így akár könnyebben tudnak egymásnak is segíteni.

Fókuszban volt, hogy legyen letisztult, de mégis szép megjelenése legyen és egyértelművé téve, hogy mit hova kell beírni vagy ha kiválasztani. Vagy ha az adott elem megnyomásakor mi fog történi.

Oda figyeltem arra is, hogy destruktív műveleteket igyekeztem elkerülni, vagyis, ha egy olyan parancs lefut, azt előtte kérdezze meg a felhasználót. Erre tipikus példa a törlés.

A színek használatára nagyobb hangsúly fektettem. Az egyes felhasználok, szerepkörüktől függően (*IV. ábra*) más – más színű a felülete (szürke, piros, rózsaszín, zöld). Az általános ablakok (bejelentkezési ablak) is külön színt kapott.



*IV. ábra: Pszichológus felülette*

### Felhasználok:

Következők a lehetséges felhasználok hozzáférése a szoftverhez. Minden szerepkör adott részhez fér hozzá.

Használat esett diagram arra jó, hogy lehet látni az egyes joggal bíró felhasználok, mihez tudnak hozzáférni, módosítani, hozzáadni, vagy akár törölni. Live In Care-nél (*III. ábra*) szemlélteti.

Egy nevelő egy eseményt rakhat fel, ami a gyerekkel történt. Például „Mecseken volt az osztálykirándulás alkalmával”. Ezt feltölti a program segítségével és természetesen gyerek beszámolója alapján (hogy érezte magát, mi az, amit magával visz). Később a szülő ezt a webes felületen meg is tudja nézni.

1. **Intézményvezető** - aki felelős az adott intézményért (igazgató), a dolgozók felvétele, módosítása, törlése. Mindegyik felülethez hozzá fér (teljes jogkör a programban).
2. **Intézmény ügyintéző** - aki azért felelős, hogy a gyerekek teljes profilja[[8]](#footnote-8) , a gyerekek szüleinek adatai felkerüljön az adatbázisba a programon keresztül, és szülő bejelentkezés jogait kezeli a gyereket megkérdezve.
3. **Pszichológus** - akinek a feladta a gyerekek alaphelyzet felmérése miután bekerült a rendszerbe közvetlen (SNI[[9]](#footnote-9), BTM[[10]](#footnote-10)), valamint az egyéb egészségügyi vizsgálatok rögzítése a rendszerbe.
4. **Nevelőszülő** - aki azzal járul hozzá a rendszer működéshez, hogy egyéb fontos adatok[[11]](#footnote-11) felvételében segíti a munkát.
5. **Szülő** - aki a gyerekkel történt eseményeket[[12]](#footnote-12) és a vizsgálatokat tekintheti meg webes felületen, regisztráció esetén. Ehhez rendelkezniük kell regisztrációs jelszóval és felhasználóval.
6. **Gyerek** - akiről az adatokat feltöltjük (alanya a szoftvernek) és ők segítése az elsődleges célja a programnak.
7. **Admin** – aki azért felelős, ha problémát észlel neki jelezze és a probléma helyzet felmérés utána, a problémát minél hamarabb megoldja. És ha valaki hibát észlel, azt ő kijavítsa/közbenjárjon. Ő segíti a betanítás is a dolgozok számára. Vagy felmerülő kérdések megválaszolására.

*Mellékeltem a lap alján, mert nagyobb fél oldalnál.*

*III. ábra: Use-Case Diagram*

### Gépigény:

A gépigény az, ami leírja azt, hogy mi az, ami feltétlen szükséges egy adott szoftver teljes körű működéséhez. Ami szükséges hardware teljesítmény foglal magában, vagy akkor egy másik szoftver is igényelhet.

1. Microsoft Windows operációs rendszer (Microsoft Visual Studio 2019-hez szükséges Windows 10)
2. Bevitelre alkalmas billentyűzet, egér
3. Egy monitor a megjelenítéshez
4. Hardver követelmény:
   1. 1,8 GHz vagy gyorsabb processzor. Négymagos vagy jobb ajánlott.
   2. 2 GB RAM a minimális; 8 GB RAM ajánlott (legalább 2,5 GB, ha virtuális gépen fut)
   3. Merevlemez-terület: a telepített funkcióktól függően legalább 800 MB – 210GB szabad hely.
   4. Az általános telepítések 20-50 GB szabad helyet igényelnek.
   5. Merevlemez sebessége: a teljesítmény javítása érdekében telepítse a Windows és a Visual Studio egy SSD-re.
   6. Videokártya, amely támogatja a minimális 720p felbontást (1280x720).
   7. A Visual Studio a legjobban WXGA (1366 x 768) felbontással működik.

### Interjúk:

Egy komplex program tervezése során, elengedhetetlen az interjú elvégzése. Hogy egy nagyobb képet kapjunk az egyes feladatkörökről egy programmal, hogy lehet ezt alátámasztani a jövőben. Emellett ismernünk azokat a fenntartásokat, amiktől az egyes felhasználok félhetnek és erre egy megoldást kell nyújtani a programba. Esetlegesen valamilyen elvárást támasztanak egy funkció irányába. „Úgy a leggazdaságosabb, ha ez a cella ott van ez meg emitt.” Vagy hogy a keresés funkció mi alapján szűrjön. Név? Életkor?

#### Fenntartások:

* **Intézményvezető:** „A szoftver használata és elsajátítása. Több odafigyelés és megfelelés.”
* **Ügyintéző:** „Nincs semmi félelmem, hiszen rengeteg programmal dolgozok napi szinten.”
* **Pszichológus:** „Ne legyen nyilvános az, amit ő felvisz a vizsgálat után. Mert az a nevelőnek szeretném írni, nem másnak.”
* **Gyerek:** „Megfelelő adatokat, információkat raknak fel rólam.”
* **Szülő:** „Nagyon jó ötletnek tűnik, hogy figyelhetem a gyerekem, csak nem vagyok az a nagy netezős/gépes.”
* **Nevelő:** „Egy-két dolgot ha nem értek, azt meg tudom kérdezni?” , „Napra készen tartani az adatokat.”

#### Válaszok:

* **Intézményvezetőnek:** A Live In Care szoftver úgy lehet elkészítve, hogy szem előtt tartsa azt, hogy a program „beszédes legyen”. Vagyis, ha egy cellát nem töltenek ki, ami kötelező, akkor szól a program, vagy ha valami nem helyesen működik (például nem elérhető az adatbázis), azt is közli. Tejesen felhasználóbarát.
* **Pszichológusnak:** Jelenleg a rendszer úgy van megcsinálva, hogy csak a szülő és a nevelő láthatja ezeket az információkat a gyerekről. Máshoz ez nem fog a program által eljutni.
* **Gyereknek:** Jelenleg a szülő webes bejelentkezéskor, meg tudod nézni, hogy a személyes adataidon kívül mit raktak fel rólad. Természtesen, ha ’Gyerek követés’-t engedélyezed.
* **Szülőnek:** Nem igényel semmilyen nagyobb informatikai tudást. Egy regisztrálást kell elvégezni első bejelentkezés előtt. Ha esetleg ebben nehézséget érezz keresse fel intézmény kollégáit és ők segítenek.
* **Nevelőnek:** A segítséget bármikor tudok/tudunk adni. Akár emailen, akár telefonon. A hiba visszajelzésnél gombnál megtalálod a fontosabb elérhetőségeket.

### Tervezés

#### Papírforgatókönyvek készítése:

A tervezés egy létfontosságú rész egy program megírása előtt, különösen oda kell figyelni a megjelenésre is. A tervezés során nagyobb kereteket terveztem meg, nem egy pontos előre kigondolt tervet. A megvalósítás során kellett változtatni, eltérni az eredeti tervtől. Ez nem negatív része a tervezésnek, csak Metro Framework-kal sok új kapu nyílt meg, amire tervezéskor nem gondoltam.

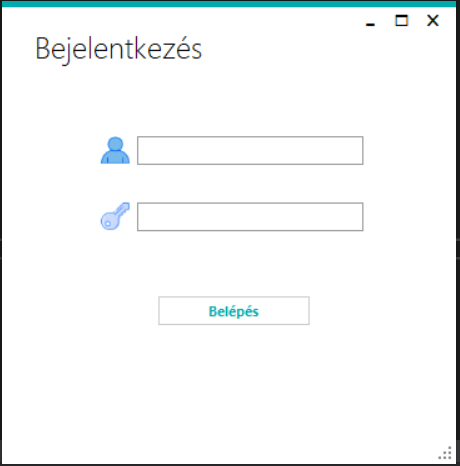
### Bejelentkezés:

A felhasználó bejelentkezés előtt egy töltő képernyőt lát (*IV. ábra*), aminek az a szerepe, hogy a program ekkor hozza létre az adatbázist. Azért ilyenkor megy végbe, hogy a bejelentkezést ez ne lassítsa egy gyengébb gépnél és természetesen azért is, mert ebben az adatbázisban van egy tábla, ami a bejelentkezést vizsgálja, hogy ki léphet be és ki nem (employes\_login). Betöltés alatt táblákat létrehozza, beszúrja a teszt adatokat, idegen kulcsokat ad hozzá és egyedi jelzőket is.

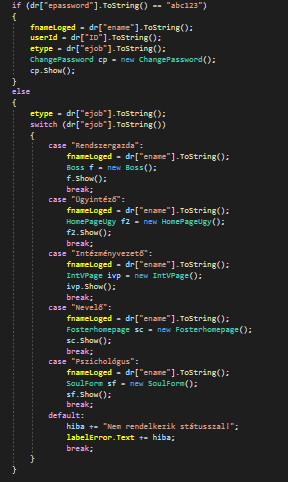


*IV. ábra: Töltő képernyő*

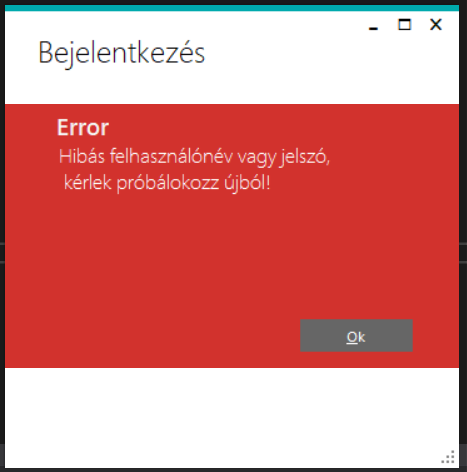
Ezután jelenik meg a bejelentkezési lehetőség. Ahol látható két input (textbox) mező.



*V. ábra: Bejelentkezési felület*

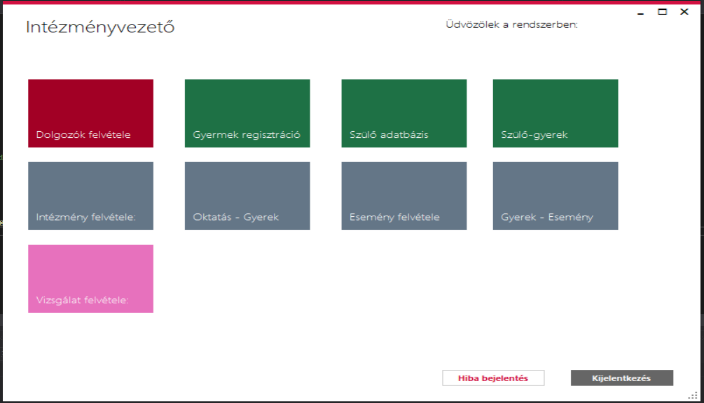


A bejelentkezés ez „if” teszi lehetővé. Ha hibás a jelszó vagy a felhasználónév akkor egy hiba üzenetet ír ki: „Hibás felhasználónév vagy jelszó, kérlek próbálkozz újból!”. Ekkor a beírt jelszó vagy felhasználónév hibás, tehát az adatbázisban nem szerepel ez a páros.



*VI. ábra: Hibás bejelentkezés és if ág a bejelentkezéshez*

A *VI. ábrán* az is látható, hogy ha a jelszónak alapértelmezett jelszó van beállítva (abc123), akkor kér egy jelszó változtatást is. A felhasználónevet automatikusan a névből generálja. Egy például egy felhasználó névre: Zuhany Rózsa -> zuhroz. A felhasználó és a jelszó is ’Case sensitivity’, vagyis számít a nagy és a kis betű.

A *VI. ábrán* az is szerepel, hogy ha sikeres a bejelentkezés, akkor attól függően, hogy milyen beosztásba vagy a munkahelyen a szerinti felületet nyitja meg. Ezt bejelentkezéskor közvetlen ellenőrzi. Adatbázisban ’employes\_login.ejob’ oszlopban található. Az asztali alkalmazásban ezt az oszlopot nézi meg a bejelentkezett felhasználó név sorával megegyező cellát. Bejelentkezéshez erre az egy táblára van szükség.

*VII. ábra: Ügyintéző (zöld) és Intézményvezető (piros) felülete*

### Felépítése, mappa szerkezet

A mappa struktúrát úgy terveztem meg, hogy a programozás közben minél kézen fekvőbb legyen. Ez hatással volt arra, hogy átlássam és gyorsabban tudjak programozni.

* Form mappa
  + Ebben található az összes form.
  + A formok úgy vannak rendezve, ahogy hozzáférnek jogokban az egyes felhasználok. Például a Soul (pszichológus) mappában van az egészségügyi vizsgálatok regisztráció formja és a bejelentkezés utáni egészségügyi vizsgálat homepage.
* Modell mappa
  + Ezekben található az összes regisztrációs egyedek modellje. Itt vannak az ellenőrzések, getterek, setterek. Itt van beszúráshoz, módosításhoz, kiolvasáshoz MySQL statement parancssorok.
* Repository mappa
  + Itt található az adott egyedhez tartozó összes listás eljárások, és DataGridView formázások egy része.
* Exception mappa
  + Adott classok Exception-jei.

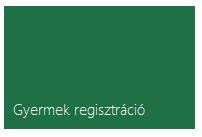
### Hozzáadás, módosítás, törlés:

* Az asztali alkalmazáson 9 féle regisztrációt lehet elvégezni. Ezek a következők:
  + Ügyintéző
    - Gyerek teljes profiljának beregisztrálása (személyes adatok)
    - A gyermek szüleinek a regisztrációja
    - Valamint a gyerek szülő összekapcsolása, hogy a rendszer lássa hogy ők rokonok.
  + Nevelő
    - Oktatási intézmény regisztrációja
    - Adott oktatási intézménnyel jogviszonyba került fiatal összekapcsolása
    - Esemény felvétele
    - Adott eseményen részt vett fiatal felvétele (szülő megnézheti)
  + Intézmény vezető:
    - A dolgozók felvétele az adatbázisba
  + Pszichológus
    - Az egészségügyi vizsgálatok felvétele, amin részt vett a fiatal (szülő megnézheti)

#### A gyermek regisztrációja

A gyermek regisztrálása által szeretném bemutatni a felvételt, módosítást és törlést. Minden esettben így használom, maximum egy-két részben tér el. Eltérés az csak, hogy egyes regisztrációknál más-más adatokat kérek be.

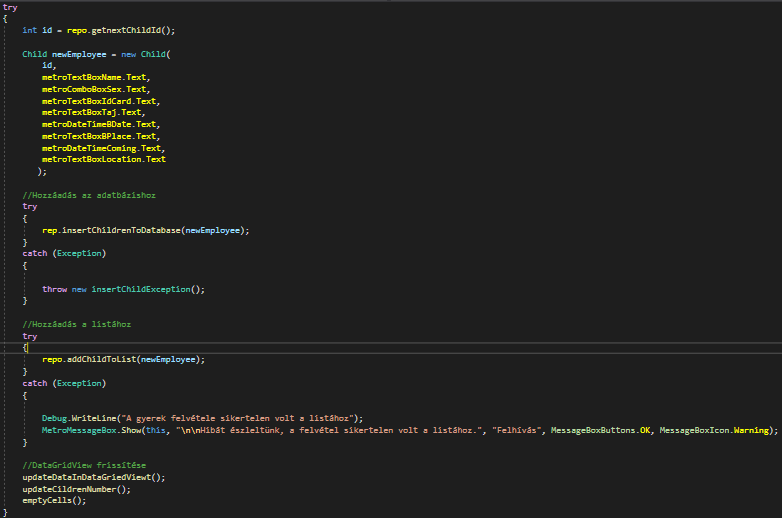
Ha a bejelentkezés sikeres volt és természetesen a mi jog körünkbe tartozik (ügyintéző, intézményvezető) a gyermek regisztrációja, akkor már regisztrálható is fel a gyerek. Ezt metroTile grafikai elem fogja jelezni, hogy van -e jogosultsága az adott felhasználónak.



*VIII. ábra: metroTile*

A regisztrációt objektum orientáltan végzem el, két fő része van. Ez minden regisztráció esetén elmondható. Elsőként az adatbázisban, majd a listában adom hozzá (lehet szülő, gyerek). Ezt azért ebben a sorendeben végeztettem el mert ha van ’egyedi’ kulcs, akkor a listához ne adja hozzá, ha az adatbázis nem engedi, mert például két ugyan olyan személy igazolvány és TAJ szám párt nem lehet felregisztrálni. Hiszen akkor adat ismétlés lenne esetlegesen.

* **A gyermek hozzáadása**

A getnextChildId() arra kell, hogy a listát megnézze mi a következő index (ID). A lista adatait megszámolja (.Count()) és egyet hozzáad, ha első akkor eggyel kezdi el indexet.

*IX. ábra: Gyermek hozzáadása*

* + **Adatbázishoz hozzáadása**

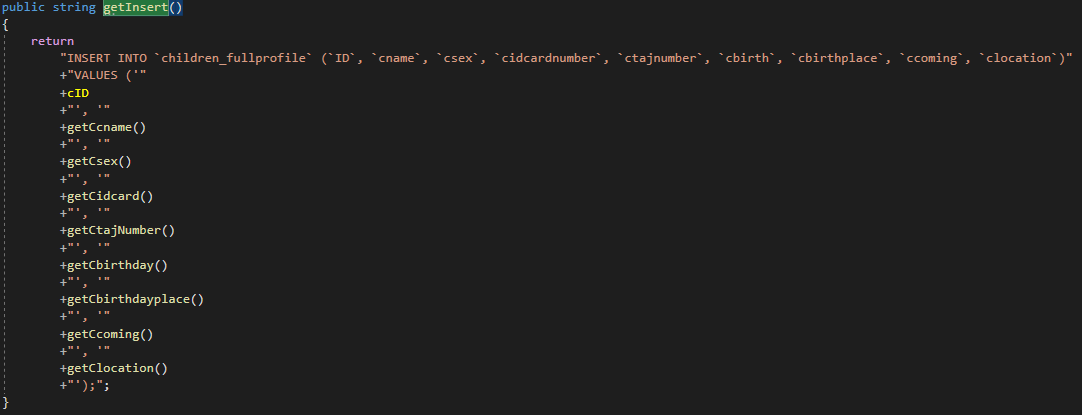
A gyermek hozzáadása csak akkor történik meg, ha bevitt adatok helyesek. Ezt minden regisztrációnál a modell-ben ellenőrzőm, hogy minden helyesen lett -e beírva az input mezőkben. Például TAJ szám az 9 db szám lehet csak. Ettől nem lehet eltérni.

Minden regisztráció esetben az adatbázishoz a következő szerint adom hozzá. Ebben az esetben Child adatait, ha minden adat helyesen lettek megadva.



*IX. ábra: Gyermek hozzáadása adatbázishoz (insertChildrenToDatabase)*

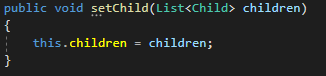
A C#-be beépített MySQL kódokat használom. Ezt bővítményként hozzá kell adni, hogy lehessen használni. A getInsert() az egy MySQL INSERT INTO statement van benne (*X. ábra*).



*X. ábra: Gyermek hozzáadása (getInsert)*

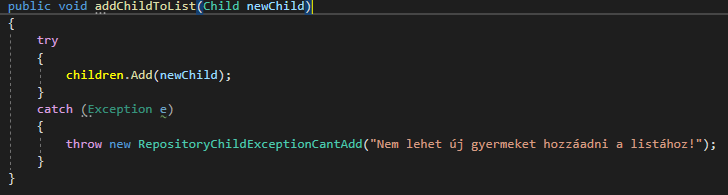
A gyerek (Child) modell osztállyal partial-ba van, vagyis megosztottan. Vagyis mintha egy fájl lenének, de mégis fizikálisan külön fájlban vannak. Ezt az asztali programozásom során sokszor alkalmaztam, mert így egy egységbe tartozó metódusokat, függvényeket tudtam egy fájlba rakni, de ettől egy osztályba vannak. Így könnyebben átlátható és könnyebben megtalálható egyes részei a programnak.

* + **Listához hozzáadása**



*XI. ábra: Gyermek lista és a setter-e*

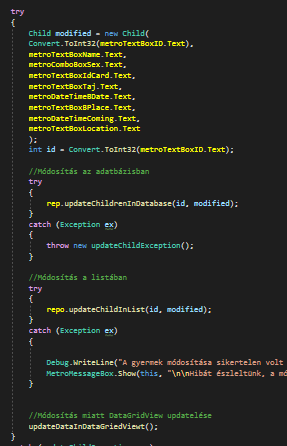
*XI. ábrán* a gyermekeknek a listája látható, amibe tárolom a gyerekek adatait. Például egy gyerek nevét, születési idejét, TAJ számát és még sok mást. Setter (setChild()) arra kell, hogy a tesz adatokat betöltse a listába, amit külön lekérünk és ennek adjuk át. Természetesen arra is, hogy a regisztrációs form-on az adatok elérhetőek legyenek.



*XII. ábra: Gyermek hozzáadása a listához (addChildToList)*

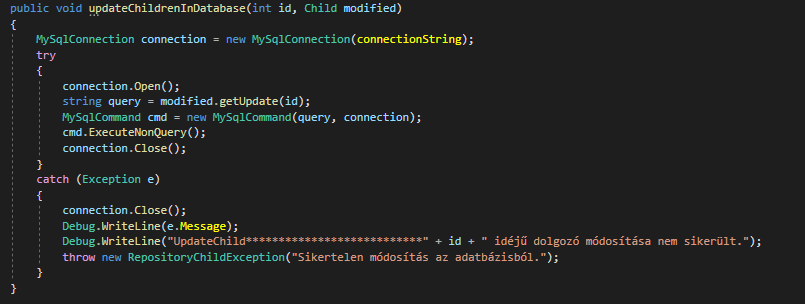
*XII. ábra* azt mutatja, hogy adom hozzá a listához Child, ha helyesen lett minden információ beírva a regisztrációs form-on.

* **A gyermek módosítás**



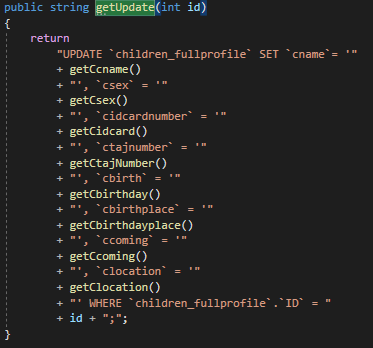
*XIII. ábra: Gyermek módosítása*

A gyermek módosítása ugyan úgy működik, mint szinte a hozzáadás. Csak helyes adatok megadásával lehetséges és ha egy sort kiválasztott, amit módosítani szeretne. Először az adatbázisba történik hiszen megint nem lehet két ugyan olyan TAJ szám és Személyigazolványszám pár. Az indexet itt nem kell ellenőrizni, hiszen egy részt a felhasználó nem tudja azt módosítani, más részt a módosítandó sor indexe ugyan az marad. Csak át kell alakítani egész számmá. Módosítás egy adott sor kiválasztásával lehetséges. Később a felhasználói dokumentáció részben részletesen leírom.

* + **Adatbázisban módosítás**

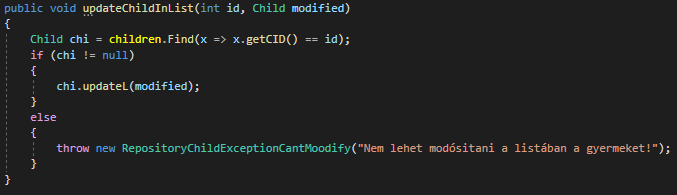
*XIV. ábra: Gyermek módosítása adatbázisban (updateChildrenInDatabase)*

Hozzáadáshoz hasonlóan itt is, egy MySQL-s UPDATE statement-et használok, getUpdate() (*XV. ábra*). Itt az id alapján update-el, mert azt nem lehet megváltoztatni. Legalább is a felhasználó nem tudja, így ez a kézen fekvő.



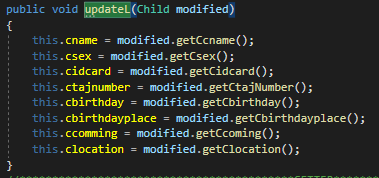
*XV. ábra: Gyermek módosítása adatbázisban (getUpdate)*

* + **Listában módosítás**



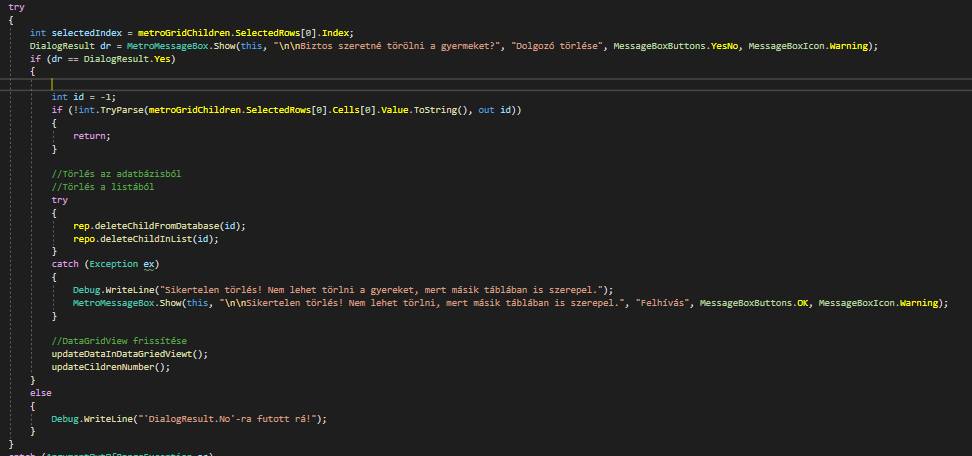
*XVI. ábra: Gyermek módosítása listában (getUpdate)*

A updateChildInList() függvényben egy id alapján módosít. Ezt úgy teszi, hogy megkeresi a listában az adott id-jű gyereket getCID() segítségével. Majd updateL() segítségével adott gyerek adatait frissíti az új adatokkal.



*XVII. ábra: updateL(), gyerek adatok módosítása*

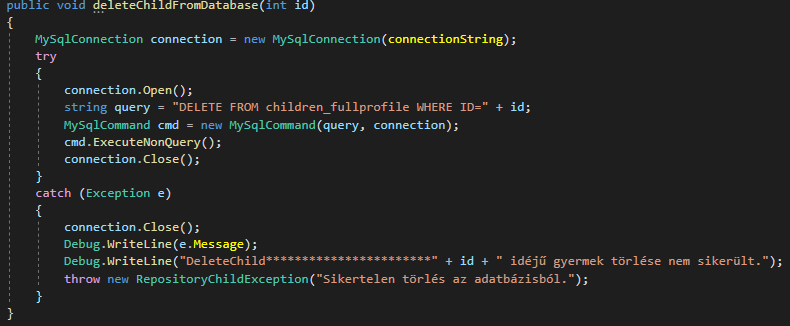
* **A gyermek törlése:**



*XVIII. ábra: Gyermek törlése*

A gyermek törlése is úgy működik, mint a módosításnál. Ki kell egy sort választani majd a törlés gombra kattintani.

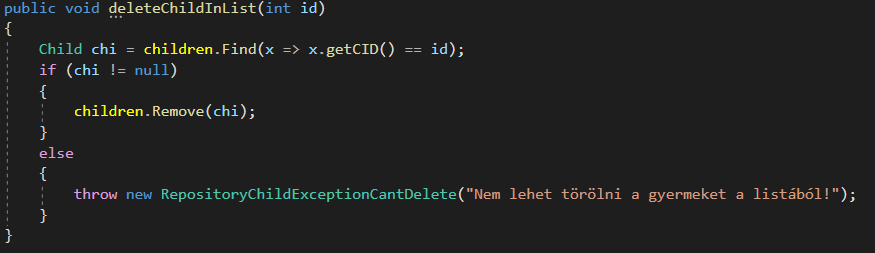
* + **Adatbázisban törlése**



*XVIII. ábra: Gyermek törlése adatbázisból*

A törlés id alapján működik. Az adott táblában indexét megfeleltetjük azzal index-szel, amit kiválasztottunk sor és azt adjuk át törléskor. Így törli az adatbázisból, ehhez MySQL DELETE statement-t használok.

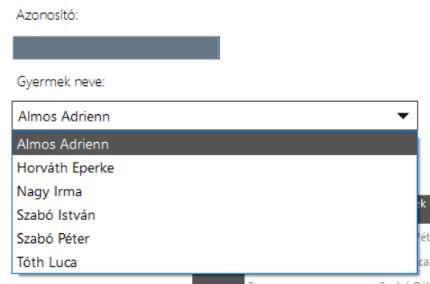
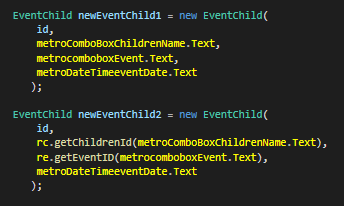
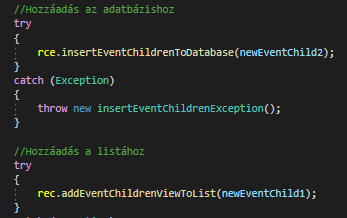
* + **Listában törlés**



*XIX. ábra: Gyermek törlése listából*

Az adott id-jű gyereket megkeresi a listában, majd a beépített C#-os Remove()-val törli a listából.

Eltérés lehet a módosítás és a hozzáadás az egyes formokon belül. Azért, mert van olyan form, amihez például a gyerekek neveit kellett ki gyűjtenem a listából és egy comboboxba feltöltenem a neveket, ami a gyerekek regisztrációs fromjához tartozott. De adatbázisba a gyerek indexét kellett beszúrnom, még a listában a nevét illesztettem be, hogy a megjelenéskor a neve látszódjon és ne az indexe.

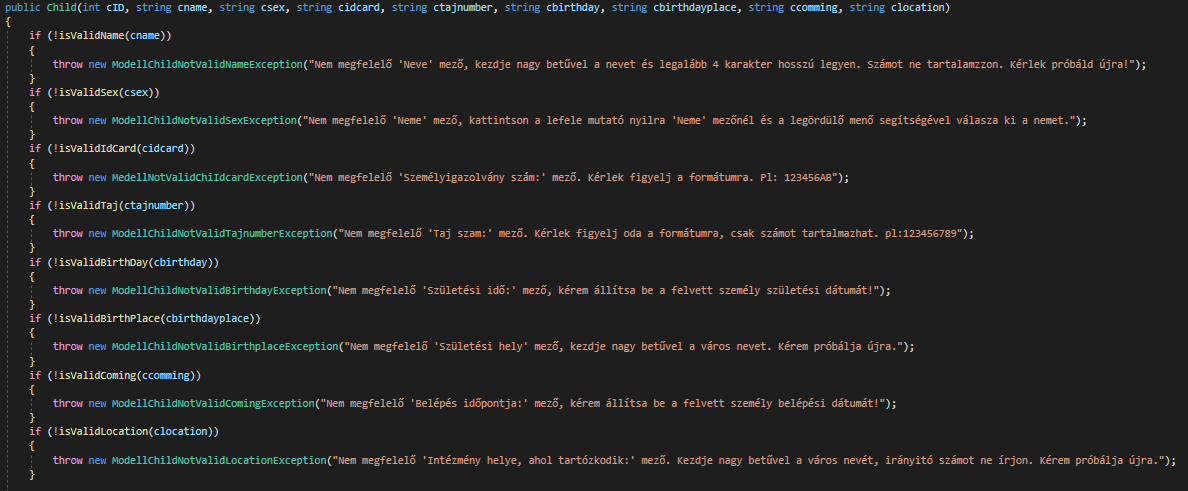
**

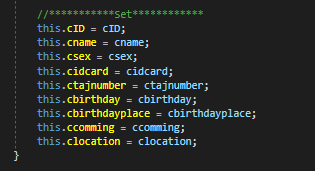
*XXIV. ábra: Adatbázis/lista hozzáadás, módosítás*

*XXIV. ábrán* az látható a legürdülő menüből kiválasztom a nevet, de nem azt szúrom be egy hozzáadás alkalmával, hanem a kiválasztott név indexét, erre jó getChildrenId() eljárás.

### Hibakezelés:

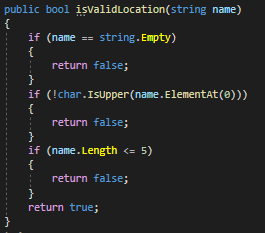
Igen fontos a hibakezelés és a felhasználó értesítése, hogy éppen miért nem jó, amit csinál vagy mi a kiváltó hiba oka. Egyik legfontosabb része a hiba kezelésemnek az a különböző regisztrációs felületek bevitt adatok ellenőrzése. Itt is a gyerek regisztráció által fogom ezt bemutatni, mert mindegyiknél így van. Modellbe nézem meg az adott adatok bevitele helytáló -e (például személyigazolvány szám, 6 szám elől majd két nagy betű).





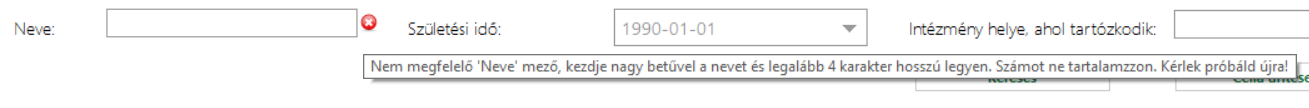
*XX. ábra: Gyerek adatainak ellenőrzése a modellben*

Ahogy a *XX. ábrán* látható, előszőr leellenőrzőm, hogy gyerekről az információk helyesen lettek -e megadva, így a teszt adatok is, le lesznek ellenőrizve miközben a listához hozzáadja a program indulásakor. Ekkor dobnak Exception-t, ha nem felelnek meg a feltételeknek.



*XXI. ábra: Gyerek adatainak egyik feltétele*

*XXI. ábra* az egy példa egy ellenőrzésre, amit a gyerek konstruktorában hívunk meg. Ha ez true-ként tér vissza akkor a bevitt adatt helyes, ha false-ként tér vissza, akkor dobja a hibát. Hiszen nem helyes a bevitt adatt.

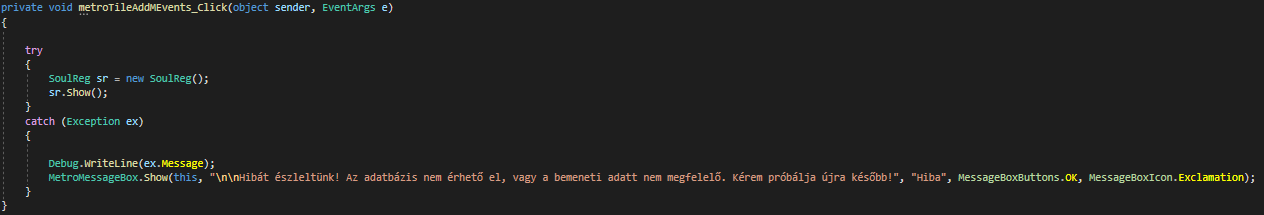


*XXII. ábra: Hiba jelzése rossz bevitt adatnál*

A *XXII. ábra* mutatja, hogy ha rákattintasz a felvétel vagy a módosítás gombra, de üres vagy nem megfelelő a bevitt adatt akkor egy piros alapon egy fehér kereszt jelenik meg villogva. Ha ráviszed a kurzorod akkor, ki írja a pontos hibát. Ez a modell-ből jön, lásd *XX. ábra*.

Törlésnél szintén van hibakezelés, ha nem jelölt ki egy sort se és úgy akar törölni, vagy ha olyan sort akar törölni, ami más táblában is van.

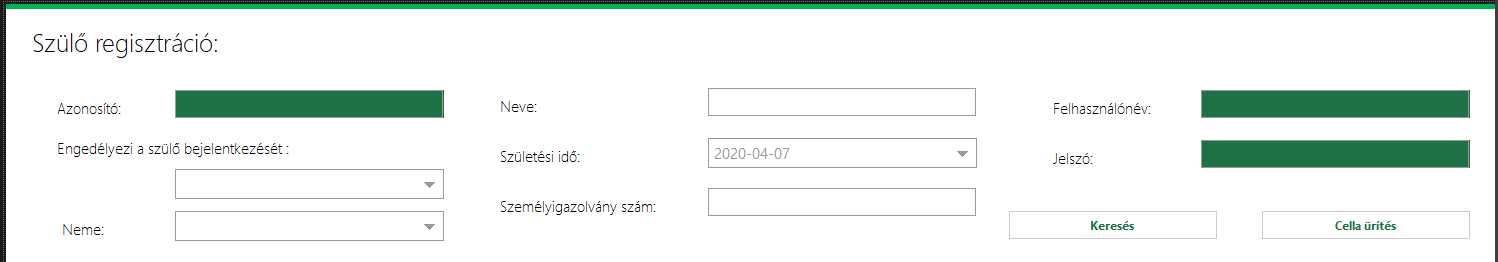
Ha adatbázis nem elérhető (*XXIII. ábra*) akkor is figyelmezteti az adott felhasználót a metroTile-ra (*VIII. ábra*) kattintás során.



*XXIII. ábra: Nem elérhető adatbázis*

### Összefüggés a weblappal

Az asztali alkalmazáshoz tartozik az is, hogy a szülőket beregisztrálja. A szülők beregisztrálása a következő képen zajlik.



*XXIV. ábra: Szülő regisztráció*

A szülő regisztrálásához *XXIV. ábra* mutatja mi szükségesek. Csak a fehér mezőket tudja az ügyintéző kitölteni. A többi automatikusan generálódik. Ez a többi formnál is így van. Miután az adatokat helyesen beírtuk. A program generál egy regisztrációs jelszót és egy felhasználót, amit az eredeti név megadásból generálja. A regisztrációs jelszó minden esetben ’abc123’ lesz. Ezzel a regisztrációs párral tud majd a szülő bejelentkezni. De van egy belépési engedélyezés („Engedélyezi a szülő bejelentkezését:” cella név), amit a gyerek dönt el, hogy szeretné -e, hogy a szülei lássa. Helyzettől függ, hogy engedélyezve van vagy sem. Mert ha a gyereket bántalmazták otthon, akkor nyilván tiltva lesz. Bármikor lehet ezen módosítani.

### Fejlesztési lehetőség

Vannak olyan tervek, amiket sajnos az idő hiányába nem tudtam megvalósítani. De szükségét érzem, hogy benne legyen. Ez azért lenne fontos, hogy későbbiekben egy cél közönség ezt ki tudja próbálni, akár egy béta verzióba. De ehhez egy-két funkciót ki kellene fejleszteni, amit a közel jövőben meg is fogok tenni.

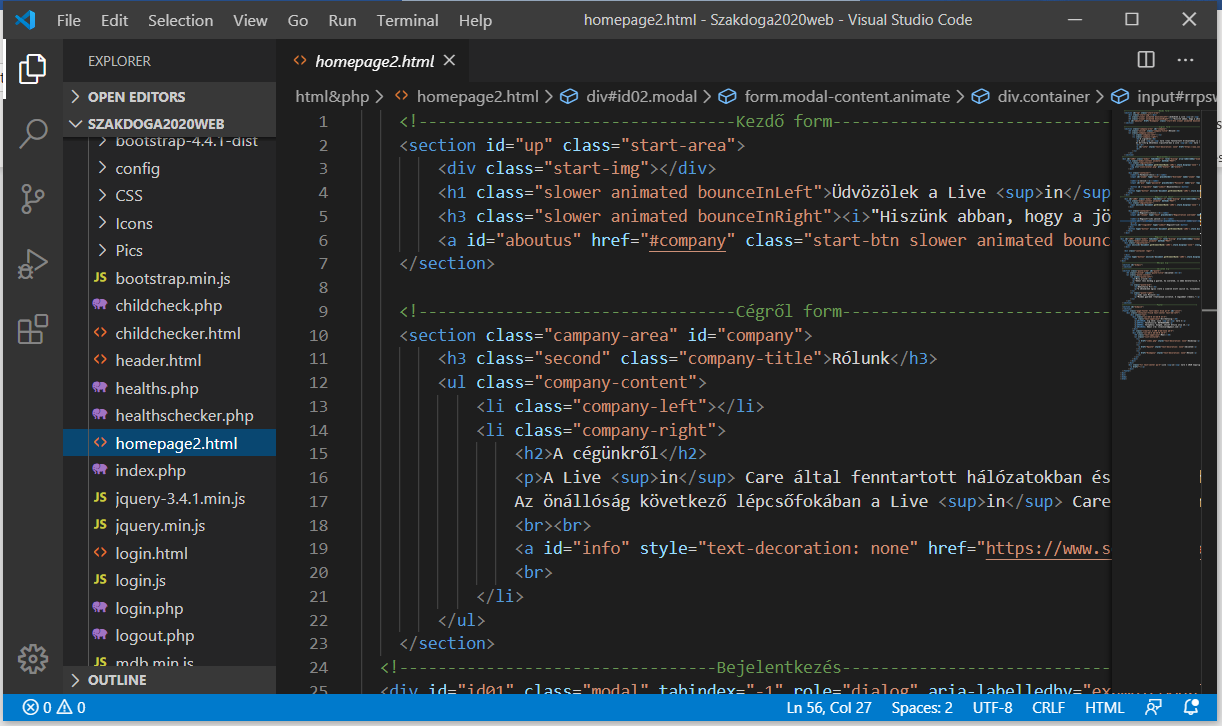
* Vizsgálat hozzáadásakor, legyen szabályozható, hogy mit lát a szülő belőle mit nem a weben majd.
* Az egyes párok hozzáadása (például gyerek-esemény, gyerek-iskola) egy formon legyen ne külön.
* A bekerülés okai, körülményeinek felvétele egyes gyerekeknél.
* Jogok szabályzása, egy felhasználó több dologhoz is hozzáférhessen, ha úgy szeretnék beállítani.
* A szül regisztrációnál biztonságosabb jelszó generálás.

## A webalkalmazás

### Bevezetés, probléma felvetés

A webfejlesztés során Visual Studio Code -t használtam (*XV. ábra*), mert tanulmányaim alatt ez a program segítette elő azt, hogy gyorsan tudjak dolgozni és hatékonyan. Fontos megemlíteni, hogy ennek van, ’Dark’ módja is, ami kevésbé rongálja szemem több órás munka mellett. Külön pozitív a programban, hogy mappa szerkezetet is meg lehet vele nyitni, így egyszere több fájlal is tudok dolgozni egy időben.

Visual Studio Code -ban .js, .php, .css, .htlm fájlkiterjesztésű fájlokkal dolgoztam. Külön fontos volt számomra a választáskor, hogy külön színeket használ az egyes kiterjesztésű fájloknál. Kiterjesztést is könnyen lehetet változtatni. Nem kellett egy felugró ablakot erre használni, csak kiterjesztést átírni. Könnyen kezelhető és sok segítséget tartalmazz, akár a kiegészítésre gondolunk, vagy a hiba keresésre.



*XXV.ábra: Visual Studio Code*

Az interjúk során, az derült ki számomra, hogy azok a gyerekek, akik bekerülnek egy állami gondoskodásba általában nem sokat tudnak a szülőkkel lenni, mert ők elmennek dolgozni, hogy minél hamarabb ki tudják a gyereküket hozni. Vagy a távolság olyan nagy a lakhelye / hajléktalan szálló között, hogy nem tudnak túl gyakran találkozni a gyerekükkel.

Valamint arra jöttem rá, hogy a szülők annak is örülnének, ha tudnák, hogy a gyerekük milyen állapotba van mind lelkileg mind fizikálisan. Mert állami gondoskodásba élő gyerekeknek van pszichológia elbeszélgetésük fél évente. Abból a szülő is tudná, hogy minden rendben van -e a gyereke. Első az egészség. De ebbe benne lenne más vizsgálatok is például szemvizsgálat, vér teszt és más is.

A szakdolgozatom ezt a részét fedné le. Hogy a szülőknek, ha nincs idejük a gyerekükkel lenni a munkahely miatt, vagy távolság miatt ez segíthet. Nyilván ez nem tudja a látogatást helyettesíteni, csak jobb oda figyelést terem a szülők irányából.

### Felhasználok

*Mellékeltem a lap alján, mert nagyobb fél oldalnál.*

*I.ábra: Use-Case diagramm*

*I. ábra* mutatja az összes felhasználót és hogy adott felhasználok milyen jogokkal rendelkeznek. A webes a szülő részre korlátozódik. Hatással van rá más felhasználó is.

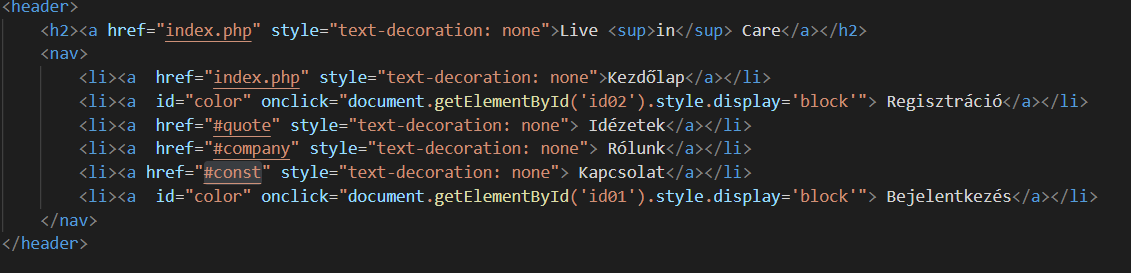
### Homepage (kezdő lap):

A kezdő lapot 3 részbe szedtem szét. A head, menu részét külön html fájlba raktam és body részt megint külön. Azért így csináltam meg, mert egyes adatok lekérdezésénél a head-t az adott php fájlba meghívtam, hogy elérje a hivatkozásokat például jquery-3.4.1.min.js vagy például style.css -t formázás szempontból.

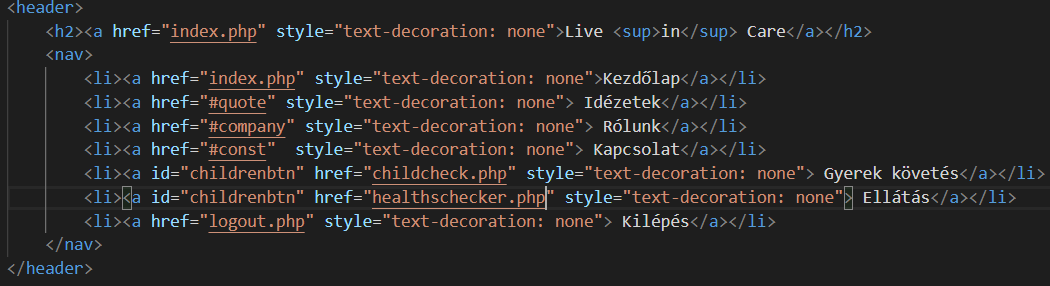


*XXVI. ábra: header.html*

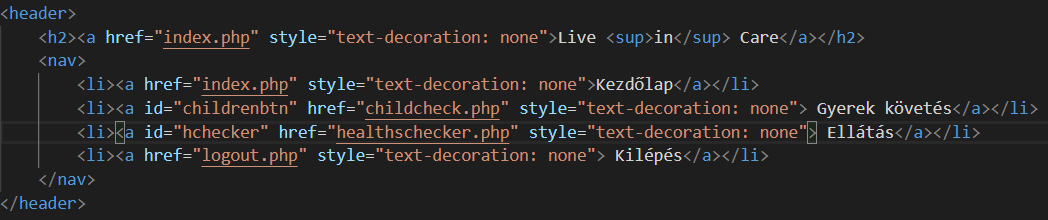
A menü 3 részből áll. Egyik az, amikor bejelentkeztél már (*XXVIII. ábra*), mikor még nem jelentkeztél be (*XXVII. ábra*) és valamint, amikor Gyerek követésre, vagy Ellátás menüre kattintottál (*XXIX. ábra*). Menüt style.css-sel formáztam meg.





*XXVII. ábra: menu.html*

*XXVIII. ábra: menuin.html*



*XXIX. ábra: childchecker.html*

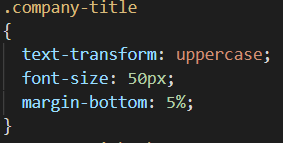
A kezdő lap az homepage2.html nevű fájl van benne. Itt a body része van benne, meg a footer és html záró része (a bodynak és a html-nek záró tagje). Kezdő lapom <section>-kből épül fel, azért úgy csináltam, hogy egyes menü pontokból legördüljön adott web részhez.

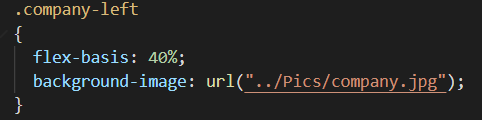
Főbb részei:

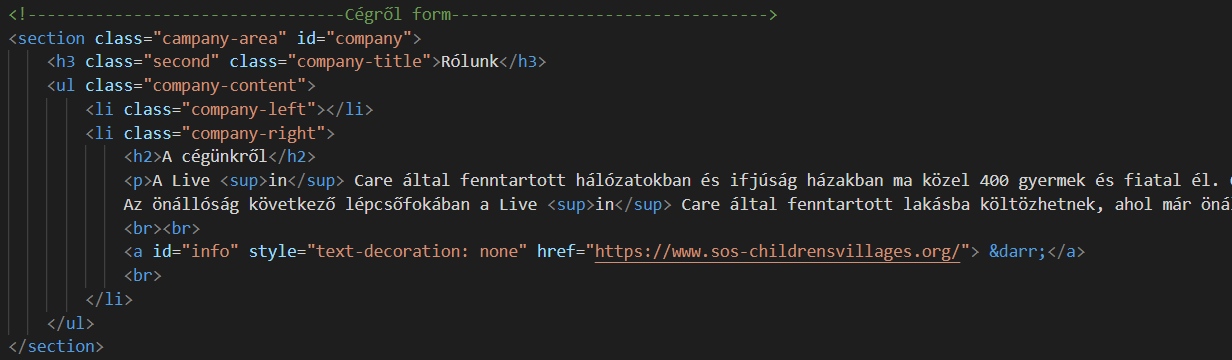
* Főképernyő (betöltéskor ezt láttod betöltéskor)
* Rólunk
* Szöveg mögötti kép
* Idézetek
* Szöveg mögötti kép
* Footer rész

Rólunk és Idézeteket egy sima mentén csináltam meg. *XXX. ábrán* látható a szerkezeti felépítése. A class=company-left -nél található a kép css-sel oldottam meg a kép beillesztést és a bal felére helyezést (felosztást). A másik oldalon van a szöveg class=company-right -nál, ami viszont jobbra van helyezve. Jobbra és balra helyezést felosztás segítségével csináltam meg, ami nem más, mint flex-basis, és meg kell adni mennyi százaléka legyen az egyik oldal és a másik oldal a szélességből. Valamint egy gombot helyeztem el, ami egy url-re mutat, és nyit meg kattintáskor. Rólunk címet is css-be formáztam.

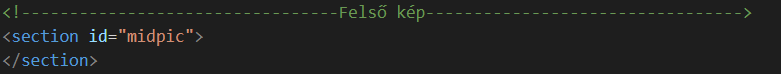
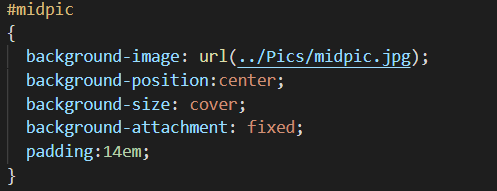








*XXX. ábra: homepage2.html-ből Rólunk magyarázat (style.css, homepage2.html)*

Háttérben a képeket css segítségével oldottam meg. Középre helyeztem a background-position:center-rel, majd a background-size: cover-rel a hattérbe helyeztem, majd fixaltam hogy ott nem mozduljon el a background-attachment: fixed -del. *XXXI. ábra* látható piros keretbe ezt a terület.

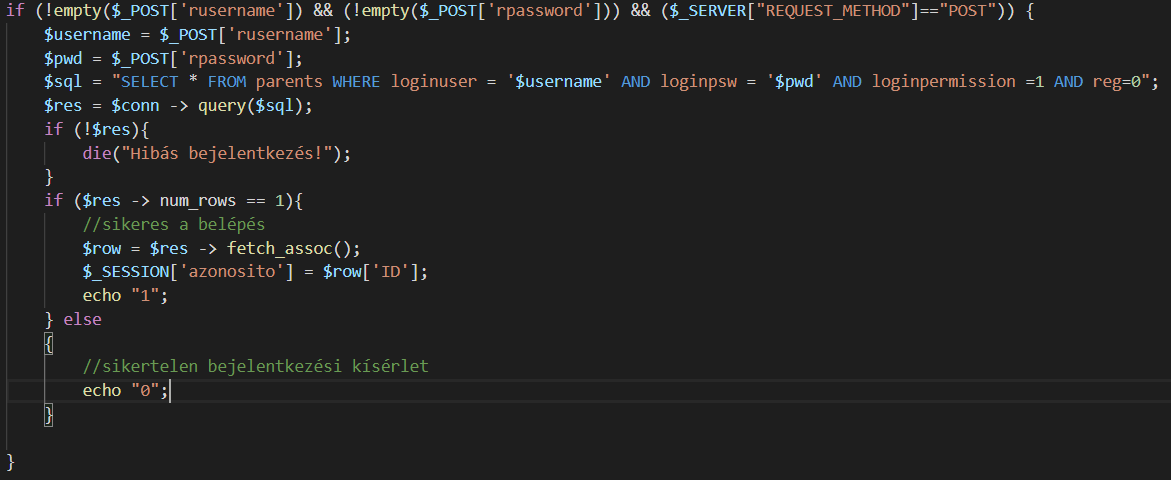
*XXXI. ábra: Szövegek mögötti kép (style.css, homepage2.html )*

### Regisztrációs felhasználónév és jelszó használata:



*XXXII. ábra: Regisztrációs jelszó ellenőrzése (homepage2.html)*

A webes felületre kizárólag csak szülők tudnak bejelentkezni. Amikor egy szülőt beregisztrálnak az asztali alkalmazáson, akkor kap a nevéből generált regisztrációs nevet és ehhez egy regisztrációs jelszót, amit majd meg kell változtatnia.





*XXXIII. ábra: Regisztrációs jelszó ellenőrzése (reg.php, login.js)*

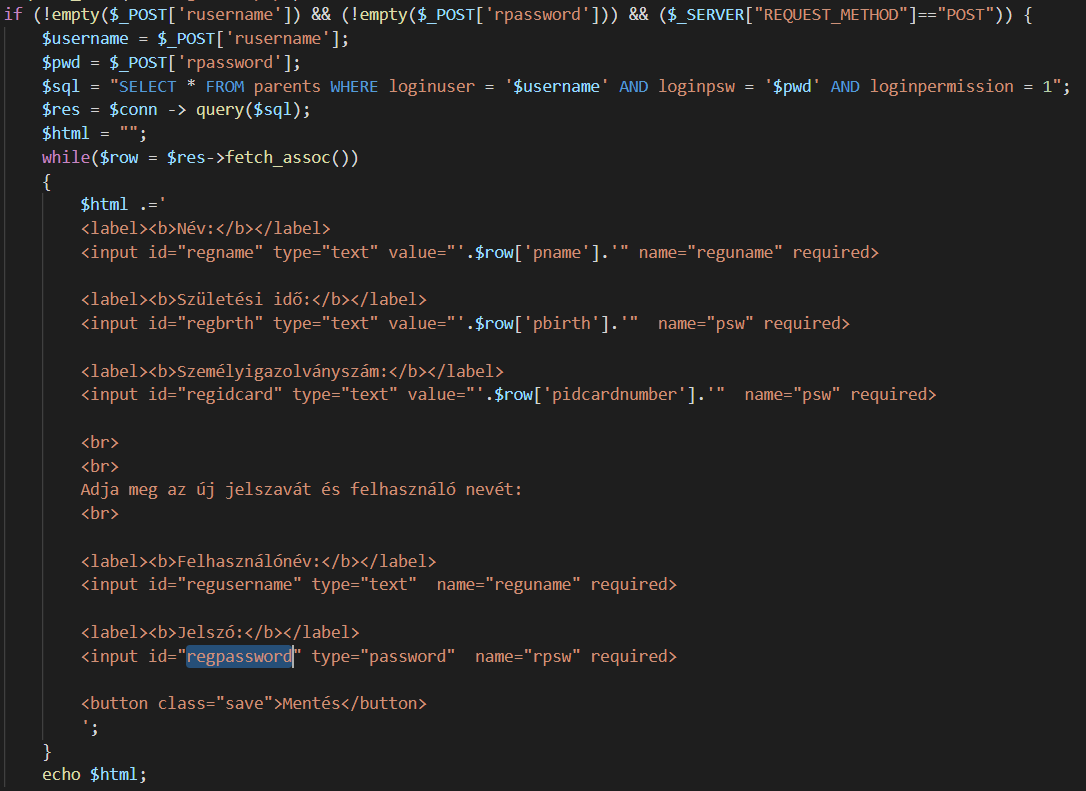
A login.js (kék keretes kép, *XXXIII. ábra*) fájlban a gomb lenyomásakor, kiolvassa a két inputból a beírt felhasználónevet és jelszót, ezt elmenti, majd át adja reg.php-nak, az le ellenőrzi, hogy van -e ilyen jelszó pár az adatbázisban, és regisztrált -e már (adatbázisban ez a reg oszlop, ha igen akkor 1-es van). Ha igen, akkor login.js-ben, data egyenlő lesz 1-gyel, és ez a modal (*XXXII. ábra*) ablak eltűnik és felugrik egy másik.

### Adatain ellenőrzése és jelszó változtatás:



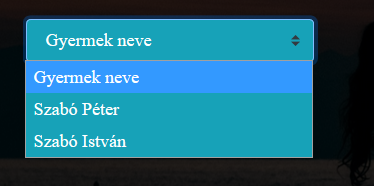
*XXXIV. ábra: Regisztráció mentés (homepage2.html)*

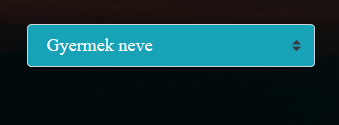
Utána megnyitja a második modult, ha helyes a regisztrációs jelszó és felhasználónév (*XXIV. ábra*), és feltölti a modult html kóddal.



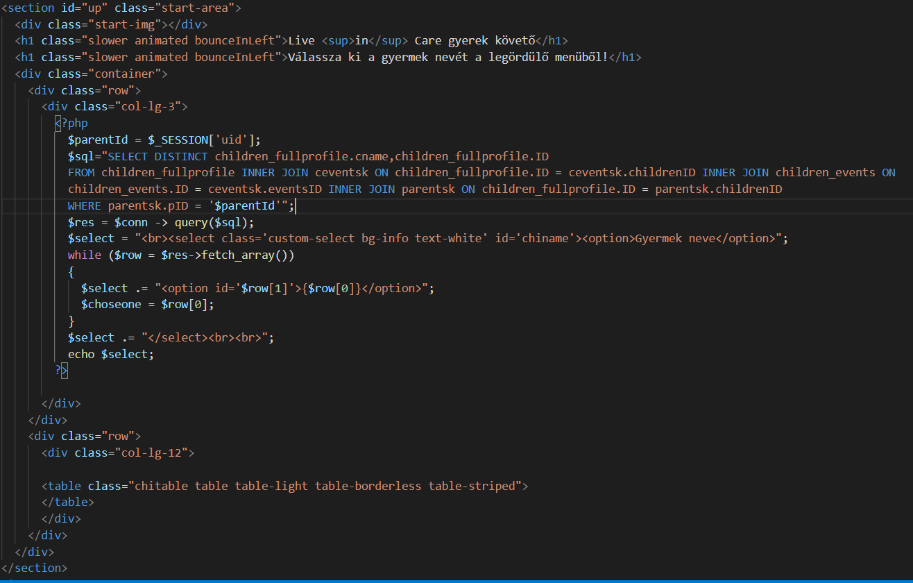
*XXXV. ábra: Adatok mentése, jelszó változtatása (preg.php)*

*XXXV. ábrán* az látható, hogy *XXIV. ábra* reger class nevű div-et feltölti az adott felhasználó adataival, amit preg.php végez el, az ajax által átadott jelszó és felhasználónév segítségével. Miután az új jelszót beírtuk, és a mind a két jelszó megegyezik (új jelszó és megerősítés is), és rákattintunk a mentés gombra, akkor a reg oszlopban a cellát 1-re álltja ennél a felhasználónál. Ami azt jelenti, hogy regisztrált felhasználó. Ezután már nem tud regisztrálni újból. Emellett természetesen a jelszót is megváltoztatja az új jelszóra és a többi változtatott adatot is, ha azok nem voltak megfelelően félvéve. Mert asztali alkalmazás által helyesen lettek félvéve, de itt van lehetőség az adatokat ellenőrizni és esetlegesen javítani a rosszul felvitt adatot. Utána már lehetséges is a bejelentkezés. Bejelentkezést login.php ellenőrzi, amit login.js ad át neki. Ugyan az a menete, mint a *XXXIII. ábránál.* Ellenőrzés és ha van egyezés az adatbázisban, akkor csak itt átirányít az index.php-ra a bejelentkezés után. Itt játszik szerepet menuin.html, menu.html és a childchecker.html, mert megváltozik a menü sáv. Például a Regisztrációs menü pont eltűnik bejelentkezéskor hiszen, akkor már nincs szükség rá.

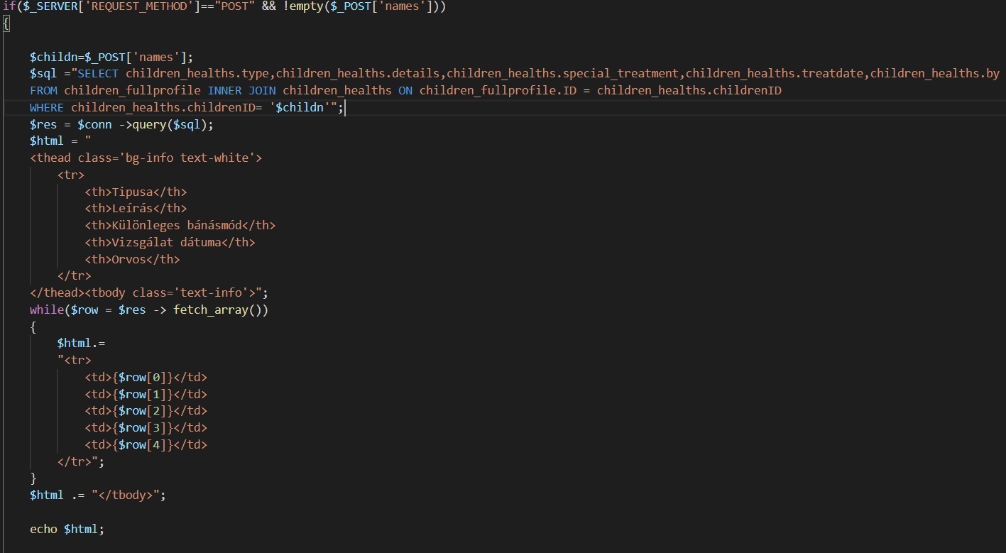
Miután beléptünk két új menüpont lesz. Gyerek követés és Ellátás menüpontok. Két rész majdnem ugyan úgy működik csak más adatokkal. Gyerek követés szemszögéből magyarázom el a működését. Gyerek követés, azt a funkciót adja, hogy adott eseményt kiírja, ami a gyerekkel történt, a kiválasztott név alapján (Kirándulás a mecsekbe). Ellátásnál, pedig az egészségügyi vizsgálatokat, amit a gyereken végeztek (szemvizsgálat). Miután rákattintasz a Gyerek követés menüpontra, akkor átirányít a gomb childcheck.php-ra.



*XXXVI. ábra: Legördülő menü*

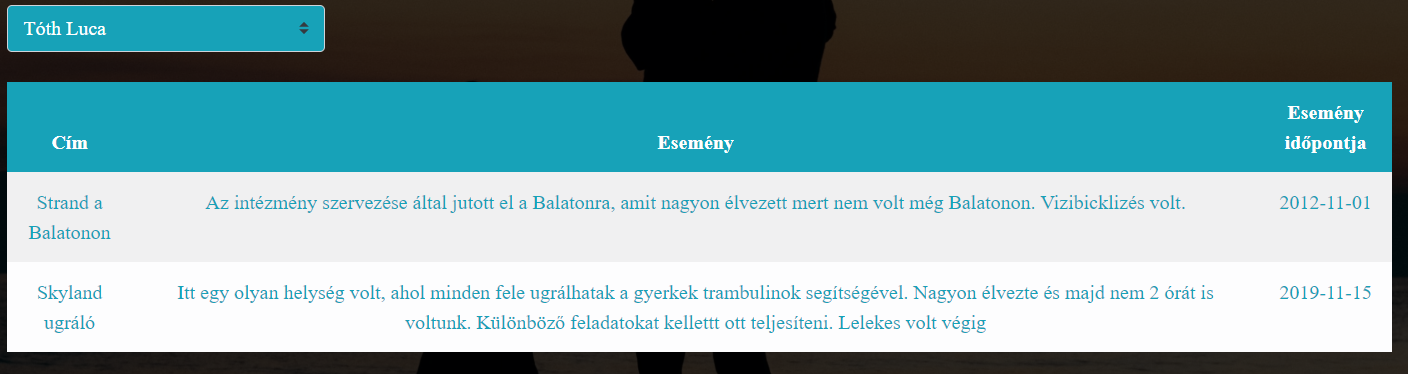


*XXXVII. ábra: Gyerek eseményeit kiírása, kiválasztott gyerek alapján (childchecker.php)*

Itt a legördülő menü-t (*XXXVI. ábra*) feltölti az adott szülőhöz tartozó gyerekek neveivel, de csak egyszer az $sql segítségével, ha tartozik a gyerekéhez bejegyzés. Miután kiválasztunk egy gyereket a legördülő menüből, a kiválasztott névhez tartozó eseményeket megjelenít egy táblázatban. Ezt a következő képen csinálja.



*XXXVIII. ábra: Gyerek eseményeinek kiírása (login.js, selctchildrendata.php)*

Miután kiválasztottuk egy gyerek nevet, login.js ezt tovább adja a nnévhez tartozó id-t selectchildrendata.php-nak, ezt id-t bele rakja $sql változóba és leszűri az adott névhez tartozó eseményekre és csak azokat kéri le. Ha van egyezés az ajax beilleszti az html kódot a chitable nevű classba (ebben az esetben table-be). A kiírásra egy példa *XXXIX. ábra*. Ha nincs ilyen esemény, akkor a gyermek neve meg se jelenik a legördülő menübe (csak „Gyermek neve” lesz a legördülő menübe), vagyis azt jelenti nincs adott szülőhöz tartozó gyermekhez bejegyzés. Ugyan ilyen módon működik az Ellátás is, csak más adatokkal.

*XXXIX. ábra: Gyerek eseményei a weblapon*

# Melléklet:

## A képen szöveg, térkép látható Automatikusan generált leírásE-K diagramm (I. ábra)

## A képen képernyőkép, számítógép, laptop látható Automatikusan generált leírásBachmann-ábra (II. ábra)

## A képen szöveg, térkép látható Automatikusan generált leírásUse-Case diagramm (III. ábra)

## Gyerek regisztrációja

### A képen képernyőkép látható Automatikusan generált leírás

1. Ennek több fajtája is van. Pl.: utógondozottság (ha fiatal 18 élet évét betöltötte, akkor átkerülhet egy olyan intézménybe, ahol csak 18 év feletti fiatalok vannak.) [↑](#footnote-ref-1)
2. Akár a nevelő bántalmazza a gyereket vagy / mert nem tud a gyerek és a nevelő együtt működni. [↑](#footnote-ref-2)
3. Az, amikor a gyerek frissen kerül az intézménybe és elfogadtatni vele, hogy most már itt fog lakni, nem a szüleivel, de szülök. [↑](#footnote-ref-3)
4. Milyen funkciókat várunk el a szoftvertől. [↑](#footnote-ref-4)
5. Például ide tartozik: design, válaszidő, könnyen kezelhető legyen így könnyen tanulható is. [↑](#footnote-ref-5)
6. A lényege az, hogy nyomon tudod követni a programod alakulását, és akár egy régi állapotba visszaállítani, kommenteket hozzá fűzni és természetesen közös. [↑](#footnote-ref-6)
7. Keretrendszer lehet például: Metro frameware (máshogy néz ki a megjelenésben). [↑](#footnote-ref-7)
8. Gyerek minden okmányainak adatai és egyéb információkat tartalmazza. [↑](#footnote-ref-8)
9. Sajátos tanulású gyerek. [↑](#footnote-ref-9)
10. Sajátos beilleszkedés igényű gyerek (lelki, egészségügyi eredetű). [↑](#footnote-ref-10)
11. Egészségügyi állapot, események, iskolázottság felvétele. [↑](#footnote-ref-11)
12. Ez lehet kirándulás, iskolai program vagy más egyéb programleválasztás. [↑](#footnote-ref-12)