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](#_Toc37144555)

[2. Bevezetés: 2](#_Toc37144556)

[3. Fejlesztői dokumentáció: 4](#_Toc37144557)

[I. Adatbázis 4](#_Toc37144558)

[a. Egyedek: 4](#_Toc37144559)

[b. Belső tulajdonságok: 4](#_Toc37144560)

[c. Egyedek, tulajdonságok típussal és névvel: 5](#_Toc37144561)

[d. Külső tulajdonságok: 7](#_Toc37144562)

[e. Összekapcsolásuk: 8](#_Toc37144563)

[II. Szoftver: 8](#_Toc37144564)

[a. Felhasználok: 11](#_Toc37144565)

[b. Gépigény: 12](#_Toc37144566)

[c. Interjúk: 13](#_Toc37144567)

[d. Tervezés 14](#_Toc37144568)

[e. Bejelentkezés: 16](#_Toc37144569)

[f. Hozzáadás, módosítás, törlés: 18](#_Toc37144570)

[A. A webalkalmazás 21](#_Toc37144571)

# 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 őket. A tanulmányaim során a phpMyAdmin (MySQL) adatbázis kezelő felületet használtam, ennek okén fogva a szakdolgozatomba is ezt alkalmazom. Itt „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 tervezgettem.

### 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 (*III. ábra*). Itt láthatjuk, milyen viszony alapján csatlakoznak az egyes táblák egymáshoz.

A képen szöveg, térkép látható

Automatikusan generált leírás

*III. ábra: EK-diagramm*

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

A képen képernyőkép, számítógép, laptop látható

Automatikusan generált leírás

*VI. 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`);

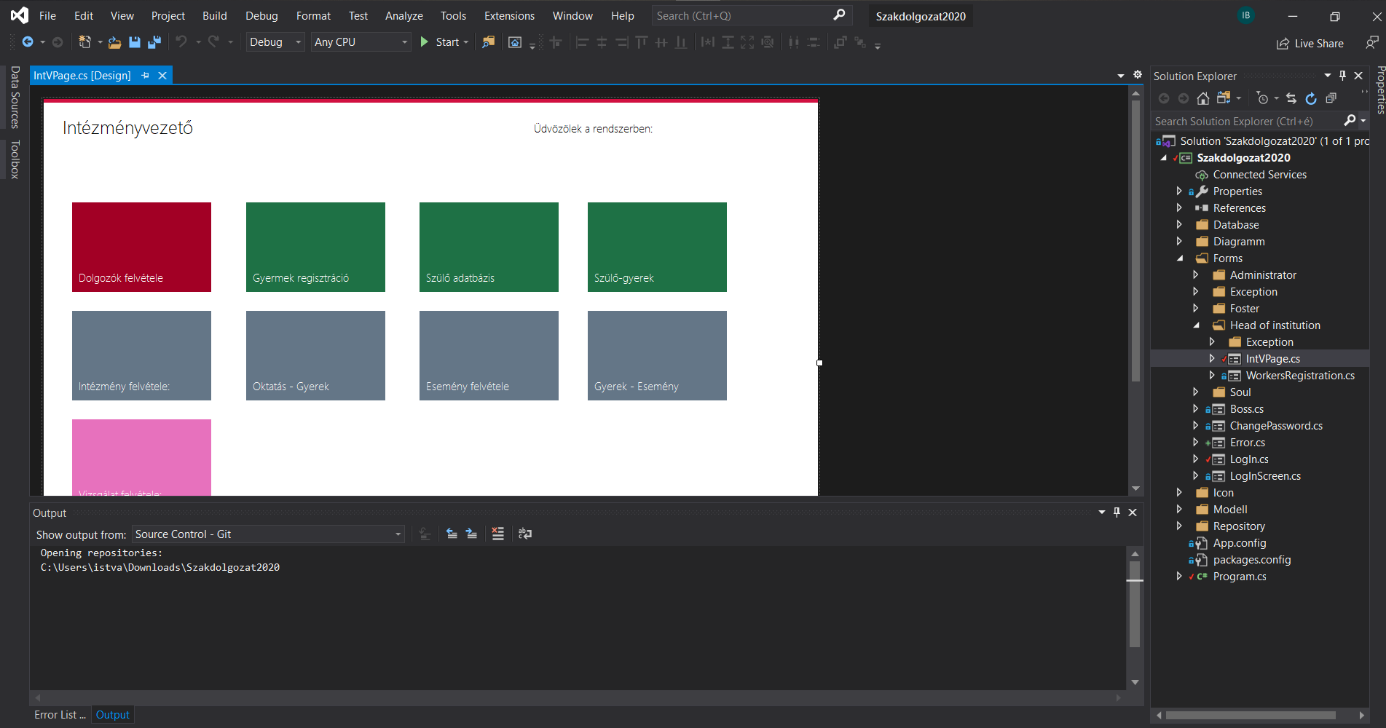
## Szoftver:

Az asztali alkalmazáshoz, ki kell találni elsőként, hogy pontosan mit szeretnék csinálni. Hiszen anélkül nehezen lehet előre haladni. 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) 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 (*I. 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.



*I. ábra: Microsoft Visual Studio 2019 felülette*

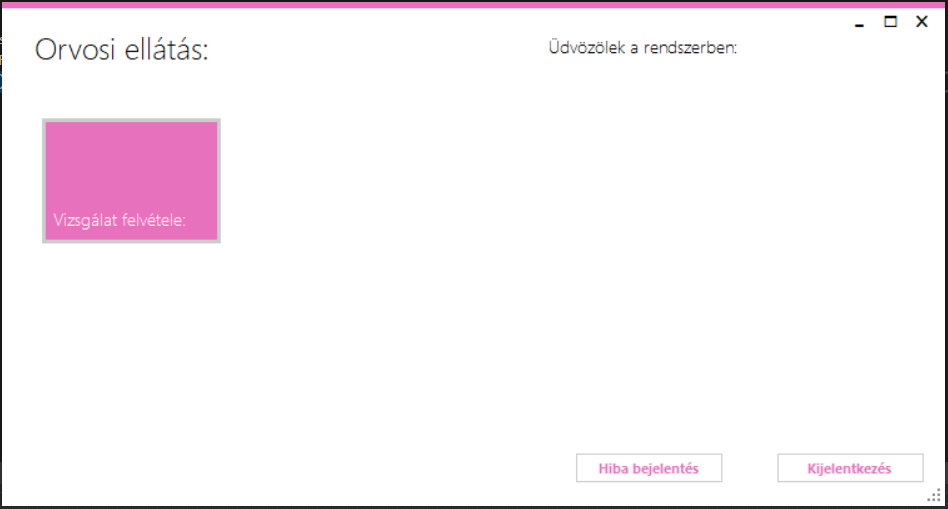
Továbbá a szoftvernek midig át kell esnie egy tesztelés folyamaton is. Így ki tudjuk az egyes gyenge pontokat szűrni. Az egyes modelleken belül adatok bevitele, módosítása, beolvasása tesztelve van, hogy nem kerüljön olyan adatt az adatbázisba, ami nem belevaló. Egyéb esettben.

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 (*II. ábra*) más – más színű a felülete (szürke, piros, rózsaszín, zöld). Az általános ablakok (bejelentkezési ablak) is külön színt kapott.



*II. 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.

*A képen szöveg, térkép látható

Automatikusan generált leírás*

*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 dolog, csak Metro Framework-kal sok új kapu nyílt meg, amire tervezéskor nem gondoltam.

* Bejelentkezés grafikai terve:

A képen képernyőkép látható

Automatikusan generált leírás

* Az intézmény vezető felületének grafikai terve:

*A képen képernyőkép látható

Automatikusan generált leírás*

* A gyerek regisztrációja

A képen képernyőkép látható

Automatikusan generált leírás

* Vizsgálat regisztráció:

A képen képernyőkép látható

Automatikusan generált leírás

* Esemény felvétele:A képen képernyőkép látható

  Automatikusan generált leírás

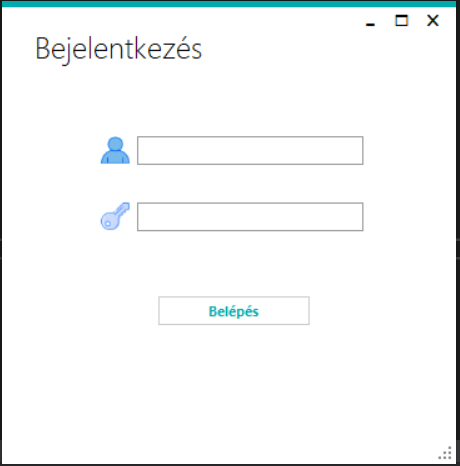
### Bejelentkezés:

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.

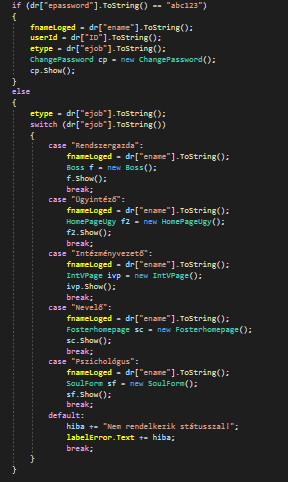


*IV. ábra: Töltő képernyő*

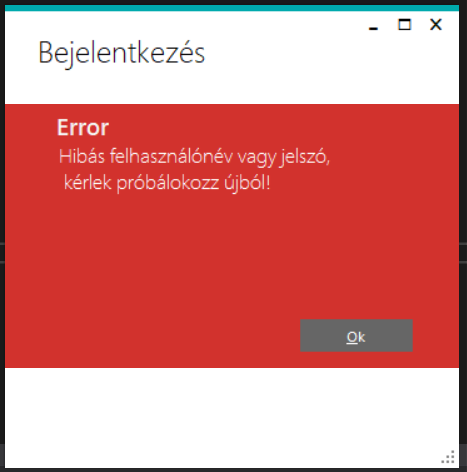
Ezután jelenik meg a bejelentkezési lehetőség. Ahol látható két input (textbox) mező.



*V. ábra: Bejelentkezési felület*

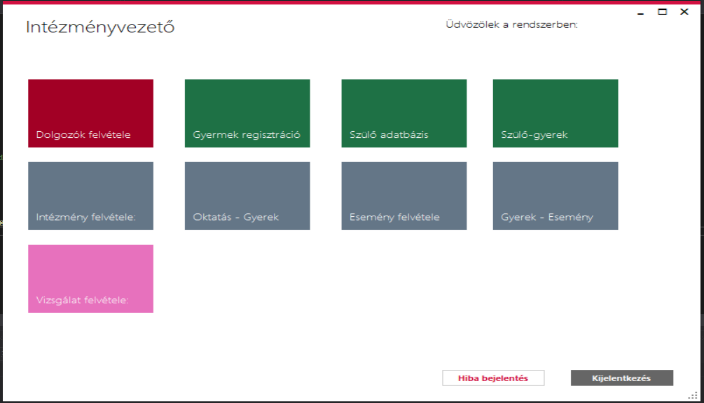


A bejelentkezés ez „if” teszi lehetővé. Ha hibás a jelszó vagy a felhasználónév akkor egy hiba üzenetet ír ki: „Hibás felhasználónév vagy jelszó, kérlek próbálkozz újból!”. Ekkor a beírt jelszó vagy felhasználónév hibás, tehát az adatbázisban nem szerepel ez a páros.



*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*

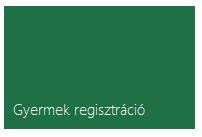
### Hozzáadás, módosítás, törlés:

* 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.

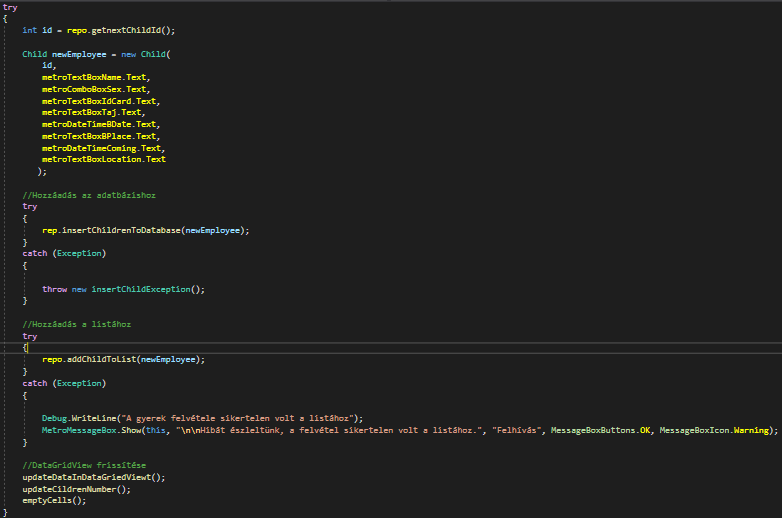
Ha a bejelentkezés sikeres volt és természetesen a mi jog körünkbe tartozik (ügyintéző, intézményvezető) a gyermek regisztrációja, akkor már regisztrálható is fel a gyerek. Ezt metroTile grafikai elem fogja jelezni, hogy van -e jogosultsága az adott felhasználónak.



*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



*IX. ábra: Gyermek hozzáadása*

* + **Adatbázishoz hozzáadása**

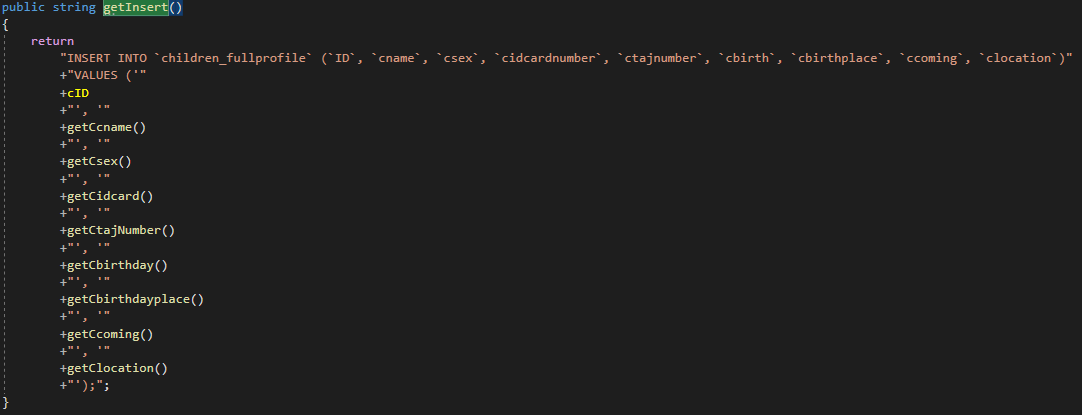
A gyermek hozzáadása csak akkor történik meg, ha bevitt adatok helyesek. Ezt minden regisztrációnál a modell-ben ellenőrzőm, hogy minden helyesen lett -e beírva az input mezőkben. Például TAJ szám az 9 db szám lehet csak. Ettől nem lehet eltérni.

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)*

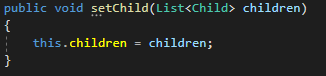
A C#-be beépített MySQL kódokat használom. Ezt bővítményként hozzá kell adni, hogy lehessen használni. A getInsert() az egy MySQL ’insert into’ statement van benne (*X. ábra*).



*X. ábra: Gyermek hozzáadása (getInsert)*

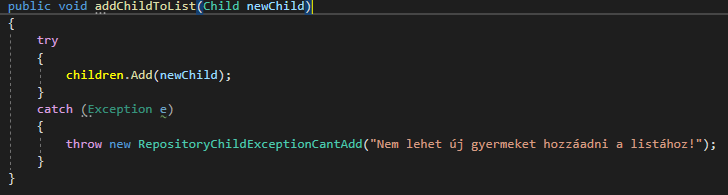
A gyerek (Child) modell osztállyal ’partial’-ba van, vagyis megosztottan. Vagyis mintha egy fájl lenének, de mégis fizikálisan külön fájlban vannak. Ezt az asztali programozásom során sokszor alkalmaztam, mert így egy egységbe tartozó metódusokat, függvényeket tudtam egy fájlba rakni, de ettől egy osztályba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

* + **Adatbázisban módosítás**

## A webalkalmazás

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)