HORIZON Admin Aplikáció

**Technológia:** .NET MAUI (Multi-platform App UI)  
**Adatbázis:** MySQL (XAMPP)  
**Platformok:** Windows  
**Namespace:** HorizonApp

**Célkitűzés:**  
A Horizon Admin App célja egy többplatformos alkalmazás fejlesztése, amely lehetővé teszi iskolák, tanárok és diákok adatainak kezelését, szerkesztését és megjelenítését egy felhasználóbarát felületen keresztül.

## **1. Adatok felhasználása módosítása**

A MySql-ben létrehozott adatbázis fontos táblái amiket fejlesztés során felhasználtam.

**Táblák:**

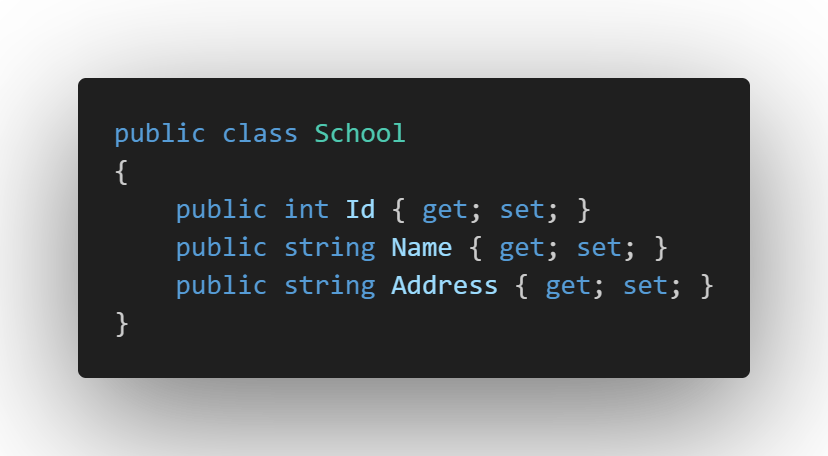
* **admins**: adminisztrátorok adatai
* **schools**: iskolák adatai
* **teachers**: tanárok adatai
* **students**: diákok adatai

**Kapcsolatok:**

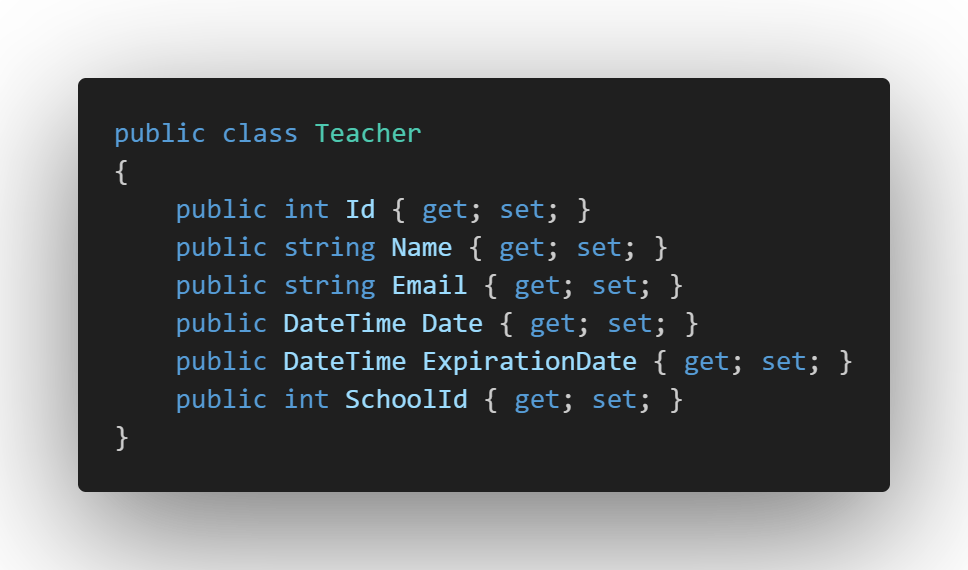
Minden tanár és diák egy iskolához tartozik (SchoolId idegen kulcs).

Az adatbázis alapján, amit a weboldalhoz is felhasználtam már adatmodelleket hoztam létre a tanároknak, diákoknak és iskoláknak a további adatkezeléshez.

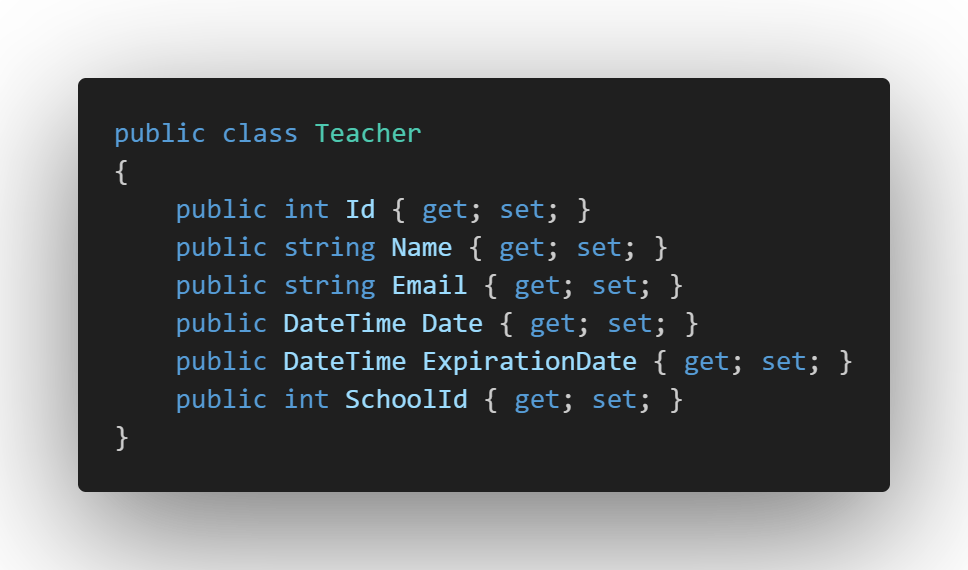
Iskoláknak:



Tanároknak:



Diákoknak:



## **2. Alkalmazás szerkezete**

**Főbb oldalak:**

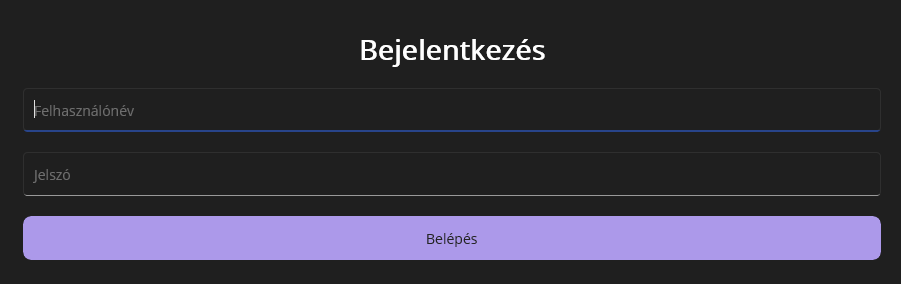
* **LoginPage:** Adminisztrátor bejelentkezés.
* **SchoolsPage:** Iskolák listázása.
* **SchoolDetailsPage:** Kiválasztott iskola tanárainak és diákjainak megjelenítése.
* **EditTeacherPage:** Tanár adatainak szerkesztése.
* **EditStudentPage:** Diák adatainak szerkesztése.

**Navigáció:**

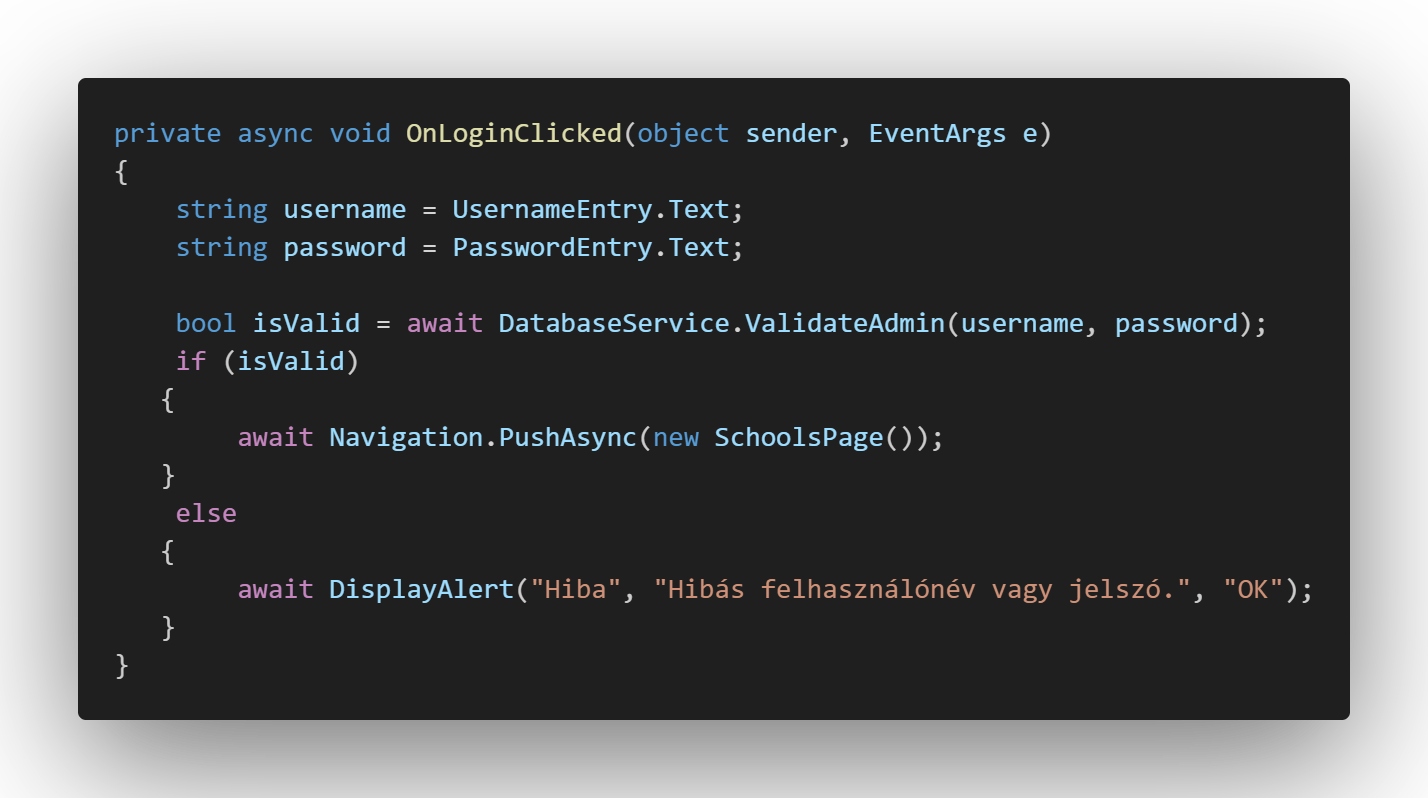
* Sikeres bejelentkezés után a felhasználó a SchoolsPage-re kerül. Itt látni fogja az összes, már regisztrált iskolát a szolgáltatásunkhoz.
* Egy iskola kiválasztása után a SchoolDetailsPage jelenik meg ahol az iskola összes tanárát és diákját láthatja.
* Tanár vagy diák szerkesztése esetén az EditTeacherPage vagy EditStudentPage jelenik meg. Itt módosítani lehet a kiválasztott személy adatait.

## **3. Bejelentkezési felület fejlesztése**

Az első próbálkozásom a kezelő felülettel nagyon egyszerű volt. Csak arra kellett, hogy legyen egy alapom amire utána dolgozni tudok.



A Háttere is csak egy egyszerű ellenőrzés volt, az első pár tesztnél még az előre bekódolt felhasználó névvel és jelszóval teszteltem. De ezt kicseréltem egy szintén egyszerű, de már az adatbázisból adatot lekérő kódra.



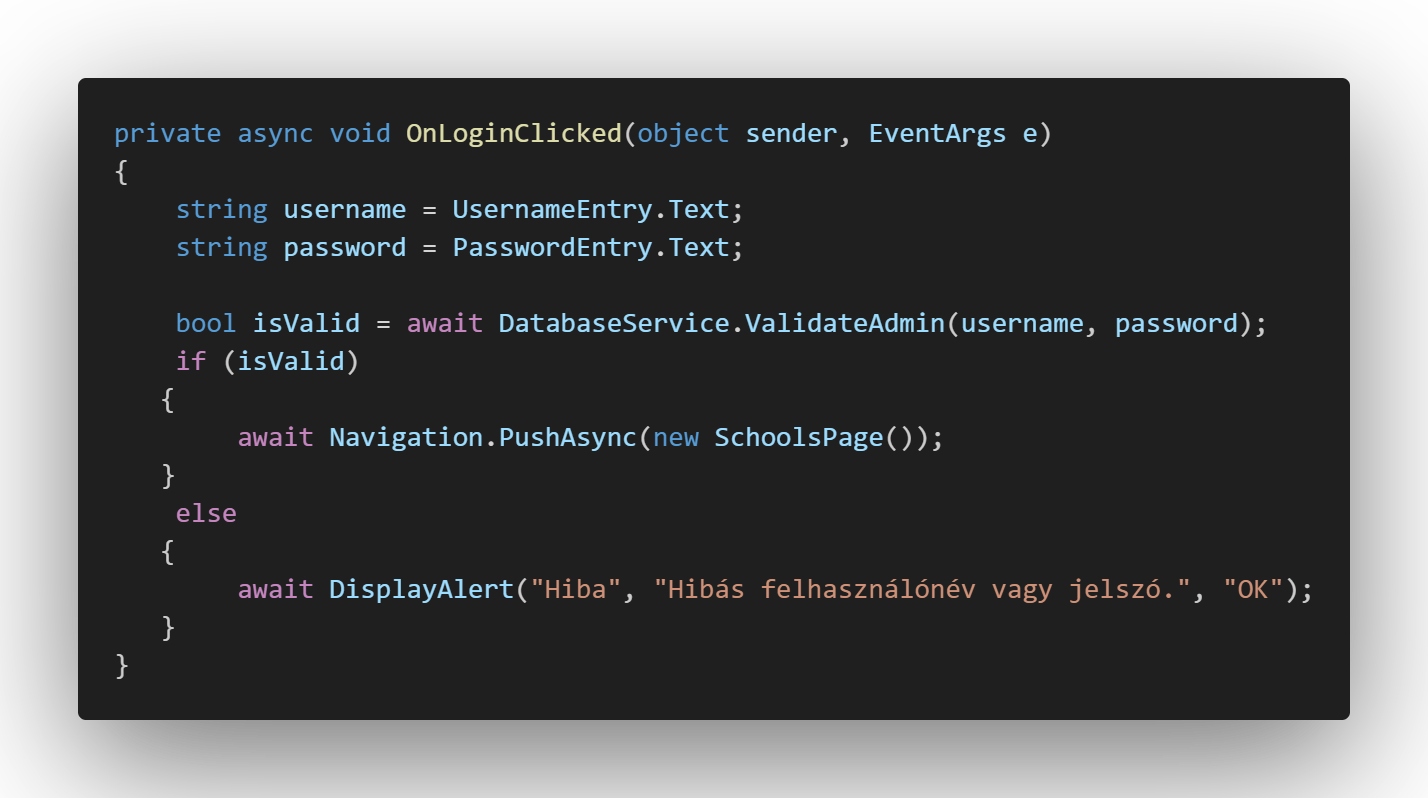
Itt nem látszik teljesen, hogy az adatbázisból kérem le az adatokat, mert ezt a részét és minden hasónló műveletet, amihez kell az adatbázis egy “központi” fájlból hajtom végre a “DatabaseService”-ből.

## **4. Schools oldal**

Miután a bejelentkezés sikeres egy olyan oldalra kerülünk, ahol láthatjuk az iskolákat névvel és címmel ellátva. Ezeket az adatokat már az elejétől kezdve adatbázisból hívtam meg.

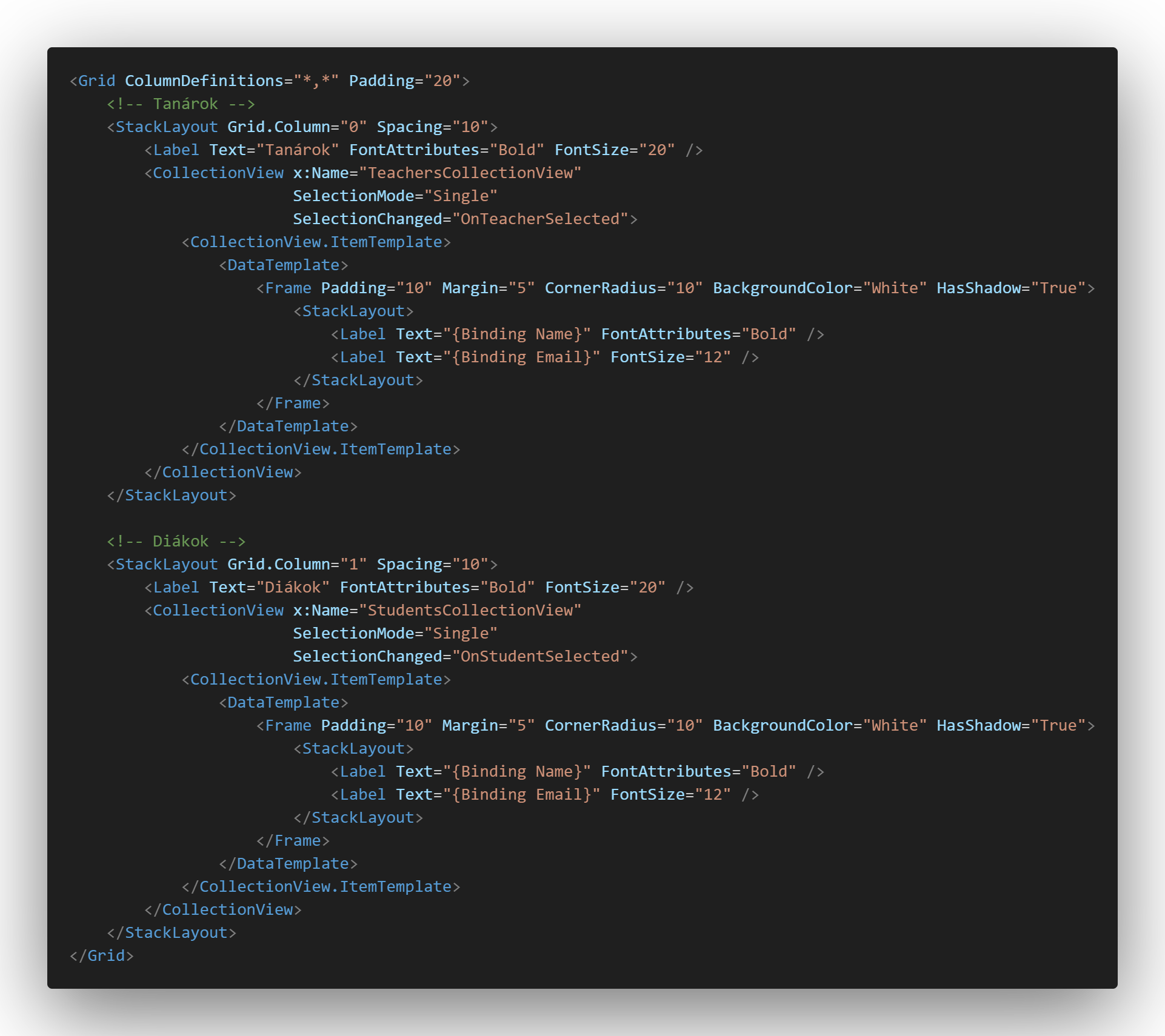


Ennek az oldalnak a fő és egyetlen funkciója, hogy navigálni lehessen a több iskola között és miután kiválasztottunk egyet átvigyen minket a részletesebb nézetre a SchoolDetailsPage-re.



## **5. Iskolák részleteinek megjelenítése**

Eleinte a személynek csak a neve és email címe volt látható ezen az oldalon. Kártya szerinte voltak elosztva a diákok és tanárok egymás alatt. Ez eléggé megnehezítette a keresést specifikus emberek után szóval a későbbiekben ezt kijavítottam.



Ennek az oldalnak is a fő feladata csak az alany kiválasztása és az elérhető személyek ki listázása. Hasonlóan, mint a SchoolDetailsPage-nek. Ezért is a háttere ennek az oldanak csak egy lekérés az adatbázisból és egy átirányítás a következő szerkesztő oldalra.



## **6. Diákok/Tanárok szerkesztése**

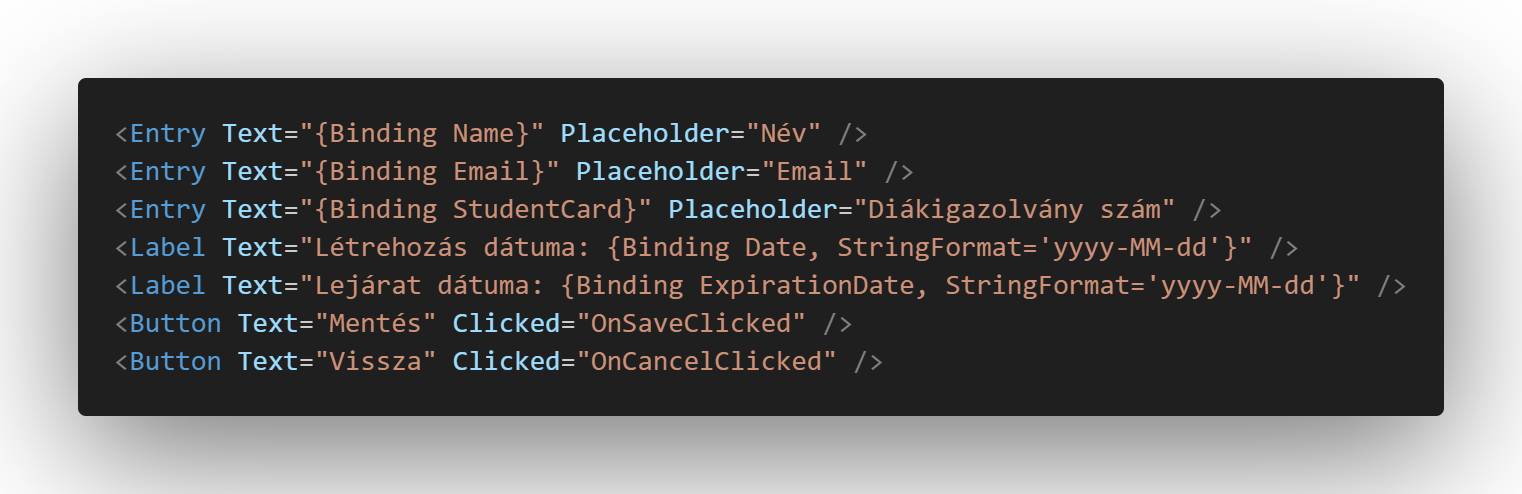
Eleinte ezt az oldalt egy PopUp ablakként képzeltem el de fejlesztés során több problémába akadtam ennek adódóan, ezért ezt elhanyagoltam és sima “ContentPage”-et csináltam ahogy az előzőknél is.

Az alkalmazásban az adminisztrátorok teljes hozzáférést kapnak a diákok adataihoz.  
Az oldal tartalmazza:

* Diák nevét (editable)
* Email címét (editable)
* Diákigazolvány számát (editable)
* Regisztráció dátumát (csak olvasható)
* Lejárati dátumot (csak olvasható)

Ez az adatstruktúra jól lefedi a rendszerünk szükségleteit, hiszen a diákokat előre iskolák hozzák létre (előregisztrációval), ezért a dátum mezők csak információs célt szolgálnak.

A felhasználó a **"Mentés"** gombra kattintva tudja rögzíteni a változásokat az adatbázisba, amit az UpdateStudent() metódus hívása biztosít a DatabaseService osztályban.



A tanárok adatainak szerkesztése szinte teljesen megegyezik a diákok szerkesztésével.

A szerkesztéshez a következő mezők állnak rendelkezésre:

* Név (editable)
* Email cím (editable)
* Regisztráció dátuma (csak olvasható)
* Lejárati dátum (csak olvasható)

A EditTeacherPage létrehozásakor a cél az volt, hogy minél **inkább egységes dizájnt** alkalmazzunk a diák szerkesztéshez képest, hogy a felhasználó számára intuitív maradjon a rendszer.

A tanárok módosítása szintén az UpdateTeacher() metóduson keresztül történik, amely biztonságosan végzi el az adatbázisban az UPDATE műveletet.

### **7. Felhasználói felület finomítások**

A kezdeti fejlesztés során a fókusz a funkcionalitáson volt, a végső verzióban azonban jelentős hangsúlyt fektettem a dizájn és használhatóság javítására is.  
 Az egész alkalmazás **desktop only** módra lett optimalizálva, figyelembe véve a célplatformot (Windows).

A következő fő változtatásokat hajtottam végre:

* **Egységes világos színpaletta** használata (#F0F4F8, #4DA6FF, #3399FF stb.)
* Minden felületen egységes, jól látható **Black TextColor**
* Kényelmesebb, tágasabb elrendezések a CollectionView-ken belül (HorizontalItemSpacing, VerticalItemSpacing használata)
* A tanárok és diákok megjelenítése **2 oszlopos Grid** rendszerben egymás mellett, így gyorsabban és átláthatóbban lehet kezelni az adatokat.

### **8. Hover effektek és stílusok**

A MAUI VisualStateManager segítségével **hover és pressed állapotokat** adtam a gomboknak, ezzel modern és interaktív felhasználói élményt biztosítva.  
 Létrehoztam egy **PrimaryButtonStyle** nevű stílust, amely minden fő gombnál alkalmazásra került:

* Alap: #4DA6FF
* Hover: #3399FF
* Pressed: #267FCC

Ezáltal a felhasználói interakciók azonnal vizuális visszajelzést kapnak.

Ezt a jövőben majd bővíteni szeretném, mert egyenlőre nem mindenen van ilyen fajta animáció. Sok időmbe került megtanulni ez pontosan hogyan is működik így idő hiányában nem tudtam már felhasználni ezt több helyen

### **9. Funkcionális bővítések**

**Diákoknál:**

* Bevezettük a **StudentCard** mezőt (adatbázisban INT, C# modellekben int?).
* Szerkesztéskor megadható vagy módosítható.
* Az EditStudentPage-en megjelenik és módosítható.

**Tanároknál:**

* Bevezettük az **IdentityCard** mezőt (adatbázisban INT, C# modellekben int?).
* Szerkesztéskor megadható vagy módosítható.
* Az EditTeacherPage-en megjelenik és módosítható.

### **10. DatabaseService javítása**

A DatabaseService osztályt jelentősen refaktoráltam:

* **reader.IsDBNull() + GetInt32() + ToString()** biztonságos használata
* Minden Database hiba kezelése **await** **MainThread.InvokeOnMainThreadAsync()** segítségével (biztonságos UI hívások)
* Felesleges kódduplikációk megszüntetése
* A **StudentCard** és **IdentityCard** mezők teljes körű beépítése

Javítást és több hibakezelést kapott a **bejelentkezési** oldal.



A jövőben biztonságosabbá tenném ezt a rendszert, mert a jelenlegi állapotában ez eléggé sebezhető. Ez a filozófia kiterjed az egész fájlra, mert itt minden lefut hibák nélkül azonban végtelen külső dámadásokra.

Az iskolák lekérésére ezt használtam.



A tanárok és diákok lekérdezése is nagyon hasonló már a kiválasztott iskolánál annyi eltéréssel, hogy ott sokkal több adatot kérek le és dolgozok fel.

Miután változásokat hajtottunk végre egy diákon ezt a kódrészletet fogja meghívni a program, hogy feltöltse az új adatokat az adatbázisba.



Itt lekezelek pár gyakori hibát azonban nem az összeset. A jövőben ezeket kibővíteném, hogy nehezebben lehessen nagy hibákat véteni.

A tanárok szerkesztő programja szinte teljesen ugyan ez pár apró változtatás kivételével.

A törlés gombok ezt a kódrészletet hívják meg. Éppen a diákok törlésgombja lesz látható, de ahogy az előbb is a tanároké szinte ugyan ez.



### **11. Végleges UI struktúra**

A végleges alkalmazás fő oldalain a következő fő elrendezést alkalmaztam:

* **LoginPage** → középre igazított, egységes mezők és gombok, jól látható placeholder szövegek.
* **SchoolsPage** → CollectionView, kényelmes spacing + hover gombok + modern világos design.
* **SchoolDetailsPage** → Grid 2 oszlopos elrendezés (Tanárok + Diákok), kártya alapú megjelenítés, minden adat jól látható.
* **EditStudentPage / EditTeacherPage** → egységes szerkesztőfelület, PrimaryButtonStyle, readonly Date/ExpirationDate mezők.

### **12. oldal: Fejlesztés Közbeni Problémák, Megoldások**

**1. .NET MAUI workload mismatch**

* **Hiba:** .NET MAUI 9.0 app, de csak MAUI 8.0 workload volt telepítve → hibát dobott.
* **Megoldás:** Frissítettük Visual Studio-t és telepítettük a MAUI workload 9.0 verzióját.

**2. SqlConnection nem ismert**

* **Hiba:** SQL Server helyett MySQL-t használtunk → átálltunk MySqlConnection-re (MySql.Data NuGet).

**3. toolkit:Popup**

* Eredeti terv volt a CommunityToolkit Popup használata, de végül inkább külön oldalakat készítettünk.

**4. UI problémák**

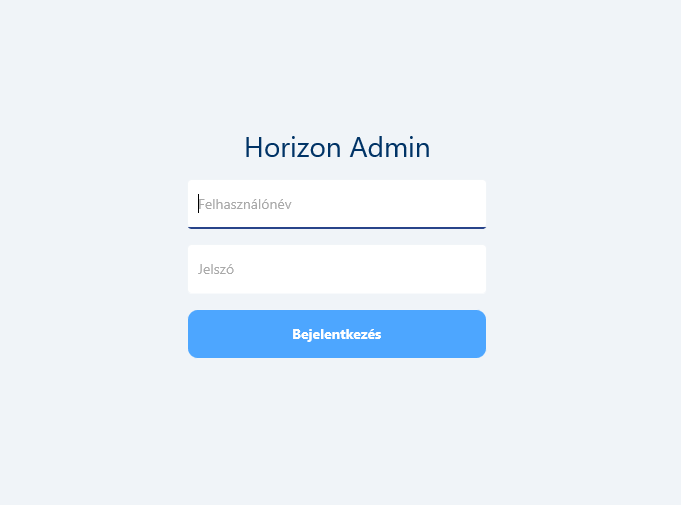
* CollectionView elemek túl szorosan helyezkedtek el → padding és margin beállításokkal javítottuk.

**5. Telefonos nézet**

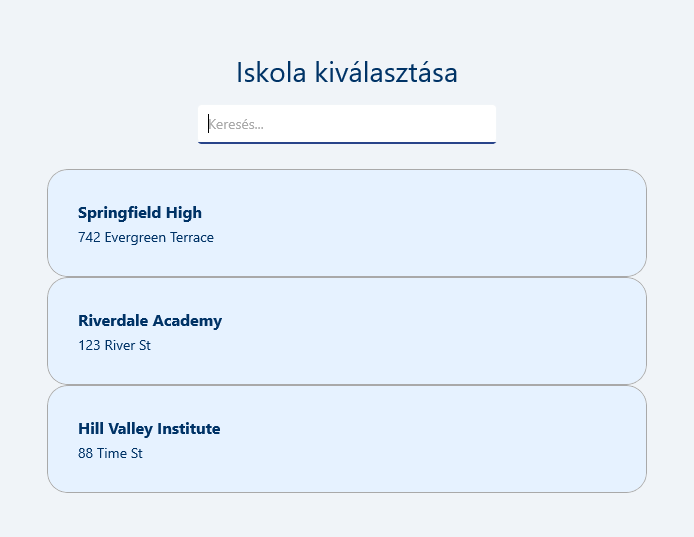
* Eredetileg telefonra is optimalizálva szerettem volna az applikációt de a UI felülettel több problémám is akadt amire nem tudtam megoldásokat így dönteném kellett, hogy telefonra vagy Desktopra fogom elsőnek megcsinálni az appot.

### **13. Az applikáció UI bemutatása**

Elsőnek a bejelentkezés oldalra (LoginPage) fog dobni minket.

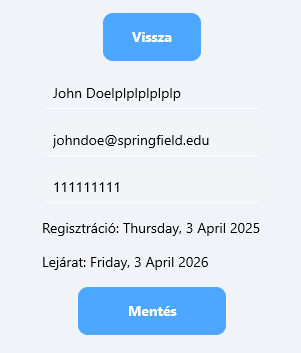


Miután itt megadjuk a felhasználónevünket és jelszavunkat tovább enged minket az Iskolák kiválasztásához. (SchoolsPage)



Itt tudunk specifikus iskolákra rákeresni, vagy csak kiválaszthatjuk őket a listából. Amint rányomunk az egyikre tovább navigál az Iskolák részletes nézetéhez. (SchoolDetailsPage

Itt a piros gombbal tudjuk törölni a kiválasztott személyt vagy szerkeszteni tudjuk őket a lila gombal. Láthatóan már pár adat előre látható. Miután rányomunk a szerkesztésre átrak minket vagy a **EditStudentPopup**-oldalra (nem PopUp csak nem lett megváltoztatva a neve) vagy **EditTeacherPopup**-ra attól függően tanárra vagy diákra nyomtunk rá.



Ez még a telefonos nézetre volt tervezve, de úgy véltem amúgyse kell több helyet elfoglalnia.

### **14. Fejlesztés összegzése**

A végső alkalmazás megfelel az eredeti célkitűzéseknek:

**Stabil desktop admin alkalmazás Windowsra a HORIZON-nek**

Minden alapfunkciót biztosít is:

* Admin bejelentkezés
* Iskolák listázása
* Iskola részletek megtekintése
* Tanárok és diákok szerkesztése
* Stabil adatbázis kapcsolat (MySQL)

### **Záró gondolat**

A Horizon Admin App fejlesztése során az elsődleges cél a stabilitás, áttekinthetőség és felhasználóbarát kezelés megteremtése volt.  
 A projekt során folyamatos iterációval, hibák és felhasználói visszajelzések alapján került kialakításra a végleges felület és funkciókészlet.

A rendszer jelenlegi formájában teljes mértékben lefedi az iskolák, tanárok és diákok adatainak kezelését.  
 A felépített architektúra és az egységes stílusleírás biztosítja, hogy az alkalmazás jövőbeni továbbfejlesztése (pl. jogosultsági szintek, statisztikák, exportálási lehetőségek) könnyen és gyorsan megvalósítható legyen.