**Vak Bottyán János Katolikus**

**Műszaki és Közgazdaság Technikum,**

**Gimnázium és Kollégium**



**54 213 05 Szoftverfejlesztő szakképesítés**

***Mini Napló***

**Készítette: Silkó Levente 2023**

# **Tartalomjegyzék**

[**Tartalomjegyzék** 2](#_Toc132465216)

[**Előszó** 3](#_Toc132465217)

[**Bevezető** 4](#_Toc132465218)

[Megvalósíthatósági elemzés 5](#_Toc132465219)

[Elvárások: 5](#_Toc132465220)

[Megvalósíthatósági elemzés: 5](#_Toc132465221)

[Opcionális (ha nem helyi szerverről fut): 5](#_Toc132465222)

[Alkalmazni kívánt programozási nyelv(ek): 6](#_Toc132465223)

[Követelmények (funkcionális) 6](#_Toc132465224)

[Követelmények (nem funkcionális): 7](#_Toc132465225)

[Működés: 8](#_Toc132465226)

[Felhasználói felület**:** 8](#_Toc132465227)

[**Felhasználói dokumentáció** 9](#_Toc132465228)

[Bevezetés: 9](#_Toc132465229)

[Felhasználói felület 10](#_Toc132465230)

[Bejelentkezési képernyő: 10](#_Toc132465231)

[Főoldal felépítésé 13](#_Toc132465232)

[Órarend oldal felépítésé 14](#_Toc132465233)

[Értékelések oldal felépítésé: 16](#_Toc132465234)

[Hiányzások oldal kinézete 19](#_Toc132465235)

[Felhasználó profil oldal felépítése 20](#_Toc132465236)

[Felhasználói felület kiegészítő elemei 21](#_Toc132465237)

[**Fejlesztői dokumentáció** 24](#_Toc132465238)

[Specifikációk 24](#_Toc132465239)

[Bevezetés 25](#_Toc132465240)

[Problémák 26](#_Toc132465241)

[Megoldások 26](#_Toc132465242)

[Projekt mappa szerkezete 27](#_Toc132465243)

[Statisztika 27](#_Toc132465244)

# **Előszó**

Az oktatás iránt elkötelezett diákok számára rendkívül fontos az iskolai információkhoz való hozzáférés, mint például az adott napon tartandó órák, az elért jegyek vagy más hasonló információk. A kormány által kifejlesztett Kréta alkalmazás erre nyújt megoldást, azonban a felhasználók tapasztalhatták, hogy az alkalmazás kinézete és funkcionalitása kissé elavult.

Ezen a ponton a jelenlegi projekt célja, hogy modern dizájn stílusokat, elemeket és újra gondolt elrendezéseket használjon, hogy frissítse és élettel tölthesse meg a felhasználói felületet, miközben egyszerűbbé teszi annak használatát. Ennek eredményeként az alkalmazás használata még hatékonyabb és kényelmesebb lesz a felhasználók számára.

# **Bevezető**

A záródolgozat témája egy mobil-os E-napló létrehozása, amely a saját tapasztalatból fakadó problémákat kívánja megoldani, miközben az alkalmazás felületét újra gondolja és egyszerűsíti. A cél olyan megoldást biztosítani, amely az átlagos felhasználó számára is vonzó, modern és letisztult, miközben hatékonyan kezeli a régi problémákat.

## Megvalósíthatósági elemzés

### Elvárások:

* Felhasználó saját fiókjával be tud jelentkezni.
* Megtudja tekinteni a legutóbbi eseményeket, jegyeket.
* Láthatja az aznapi óráit
* jegyeit tantárgyak szerint átlagolva megtekintheti, és diagram(okkal) kiegészítve.
* hiányzásait követheti.
* általános információkat megnézheti a fiókjáról.

### Megvalósíthatósági elemzés:

* humán erőforrás: 1 fejlesztő, óraszám: ~180
* hardver erőforrás:

**Minimum:**

* Operációs rendszer: Android 11 (Ios, windows, web)
* Memória: 1GB
* Háttértár: Minimum 300MB
* Periféria: Érintőképernyő

**Ajánlott:**

* Operációs rendszer: Android 13 (Ios, windows, web)
* Memória: 4GB
* Háttértár: 1GB
* Periféria: Érintőképernyő

### Opcionális (ha nem helyi szerverről fut):

* Kréta szerverek

### Alkalmazni kívánt programozási nyelv(ek):

* Dart
* Flutter (keretrendszer)
* GetX (state / route menedzser)
* Hive (adatbázis)

Szoftver erőforrás:

* Visual Studio Code (v1.76.2)
* Android Studio Emulator

Üzemeltetés:

* Csak az alkalmazás telepítésére van szükség

Karbantartás:

* Nem kell biztosítani

Megvalósítás:

* Kb. 180 óra, Költség: Nincs (csak villanyszámla, wifi/mobilnet [3MB])

### Követelmények (funkcionális)

Bejelentkezési oldal megjelenítésekor:

* Szövegdobozokkal való interakció,
* Gomb bejelentkezéshez.

Főoldal megjelenésekor:

* Alsó Navigációs Menü segítségével navigálás.
* A Profil oldalon jobb fent kijelentkezés gomb

Oldalválasztás végrehajtása:

* Navigációs menüre való érintéssel

Hatékonyság:

* Az alkalmazás könnyedén folyamatosan fut.
* Gyors válaszidő
* Pontos visszajelzés

Megbízhatóság:

* Hibalehetőség maximum szerver oldalról történhet.

### Követelmények (nem funkcionális):

Biztonság:

* Biztonságos adatkommunikáció a telefonnal.
* Biztonságos szerver kérések.

Hordozhatóság:

* Telefonon használva lehetséges

Felhasználhatóság:

* Megfelelő instrukciók biztosítása

Környezet: -

### Működés:

* Gyors működés
* Megbízható használat
* Biztonságos működés
* A hibalehetőségek kizárása
* Gyakori használat

### Felhasználói felület**:**

* modern design
* Átlátható felhasználói felület
* Könnyen kezelhető
* Interaktív elemek
* Intuitív felület
* Animált váltások

# **Felhasználói dokumentáció**

## Bevezetés:

Az alábbi mobil alkalmazás, egy létező és a közoktatásban ismert digitális naplót (Kréta) alkalmazást, próbálja újra gondolni, modern dizájn trendekkel, és bizonyos esetekben fejleszteni a már meglévő funkcionalitást.

Az alábbi dokumentáció azt a célt szolgálja, hogy egyszerűen és érthetően, elmagyarázza az alkalmazás felépítését és használatát. Illetve arra szolgál, hogy bemutassa az összes interakciót, amivel a felhasználó találkozhat.

Felhasználói felület dizájnok:

A felhasználó felület megtervezésében nagy szerepet játszott az úgynevezett Glassmorphism, egy viszonylag új és divatos dizájn trend a webdesign területén, amely az átlátszó, üvegszerű hatásokra épül. Továbbá az alkalmazás felületének fókusza fölleg a letisztultságat vette célba, hogy a felhasználónak ne okozzon gondot az eligazodás az alkalmazáson belül.

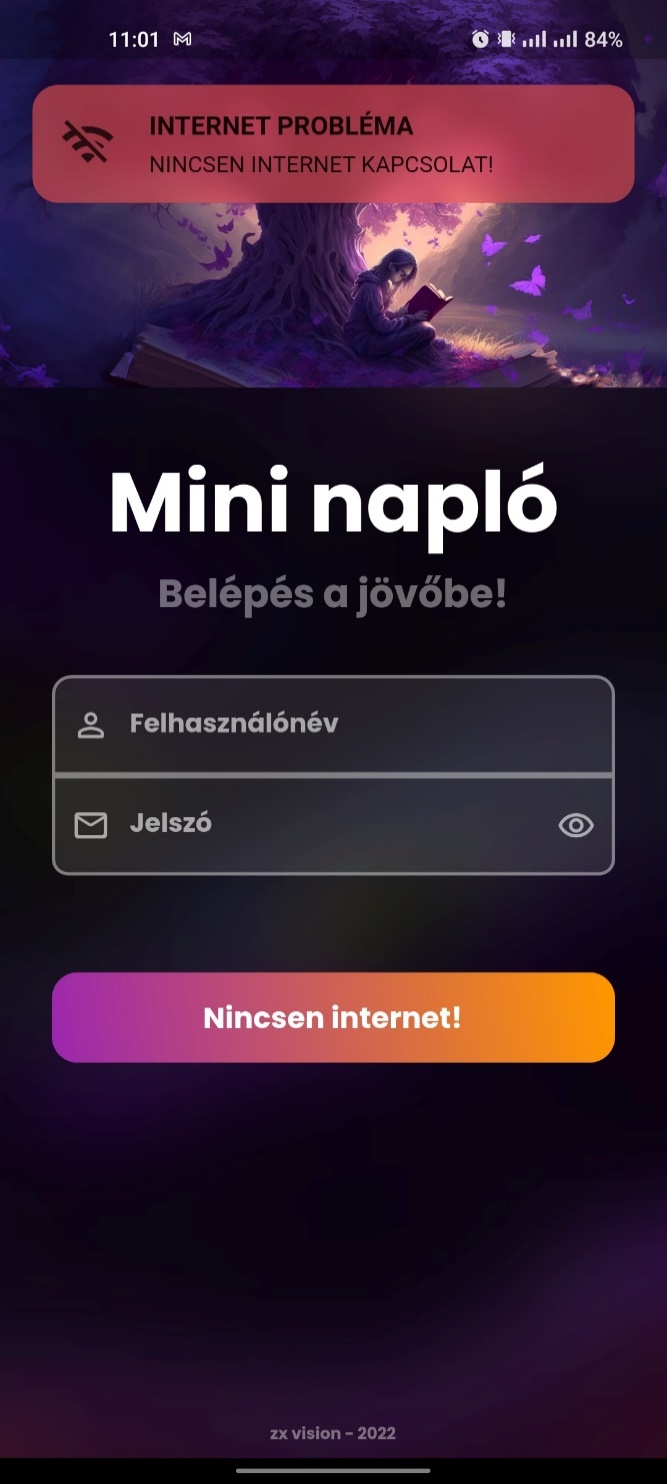
Mi is az a Glassmorphism?

A Glassmorphism trend azért is különleges, mert a tervezők általában árnyékokat, mélységet és dimenziót használnak, hogy az elemek kiemelkedjenek a háttérből, míg a Glassmorphism a hatásnak köszönhetően éppen az átlátszó felületek és az áttűnő háttér miatt fokozza a mélységet és dimenziót.

Összességében a Glassmorphism egy modern, látványos és letisztult megjelenésű dizájn trend, amely a minimalista és átlátszó hatásokat használja, hogy az elemek dinamikusabbak és szembe ötlőbbek legyenek.

Stílus megjelenése az alkalmazásban:

* Bejelentkezési képernyő szöveg dobozok
* Alsó navigációs menü
* Oldalakon található csempék (pl: Órarend oldal csempéje)
* Internet indikátor
* kijelentkezés gomb



Példának az internet indikátort hoztam, mivel ezzel a legkönnyeb demonstrálni az üveg hatás.

## Felhasználói felület

### **A képen szöveg, elektronika, képernyőkép, képernyő látható Automatikusan generált leírás**Bejelentkezési képernyő:

**Felépítésé**

Fejléc kép: ezt a képet a Midjourney AI-al készítettem. Röviden a működéséről: megadod az AI-nak hogy miket szeretnél a képeden látni, specifikus beállításokkal, mint például a képarány, ezek után generál 4 darab képet, és kiválaszthatod, hogy a négy kép közül melyiket akarod felskálázni, vagy új verziókat generálni.

Az interfész egy úgy nevezett Stack-en helyezkedik el, és a felhasználó felület és a háttér között van egy homályosítás effekt és ezen helyezkedik a Cím, Szlogen stb.

Két darab szöveg dobozzal (flutterben TextField-nek hívjak) történik a bejelentkezéshez szükséges adatok elkérése.

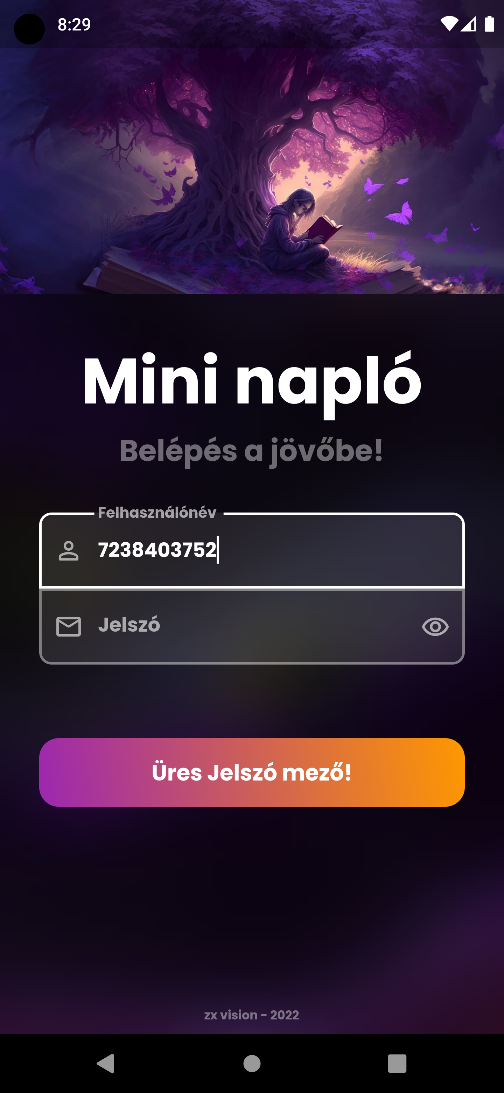
A bejelentkezésre kattintás után elindul az adatok validációja, ezután kérést intézz a kréta szerverek felé, és ha minden sikeres akkor át helyez a fő képernyőre.

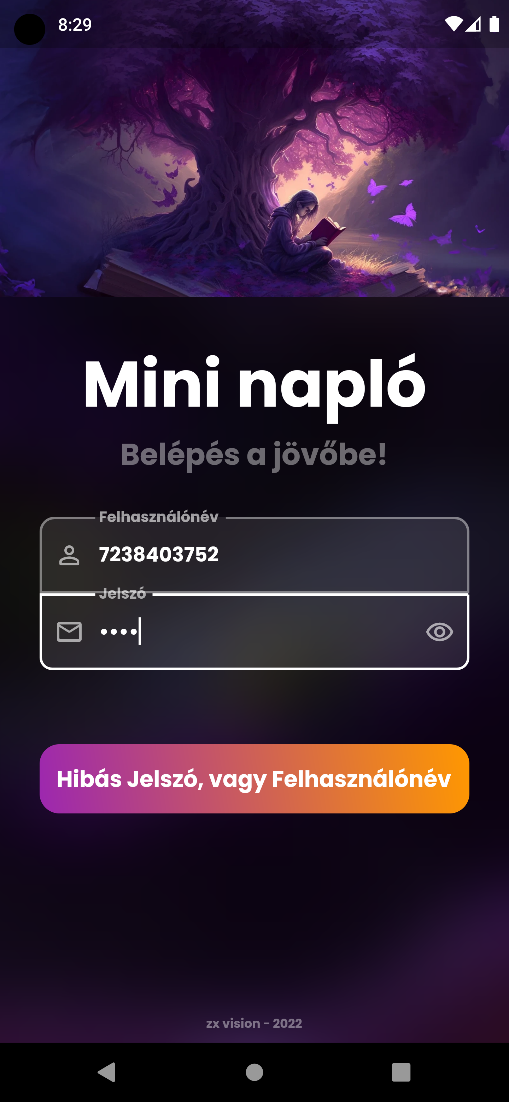
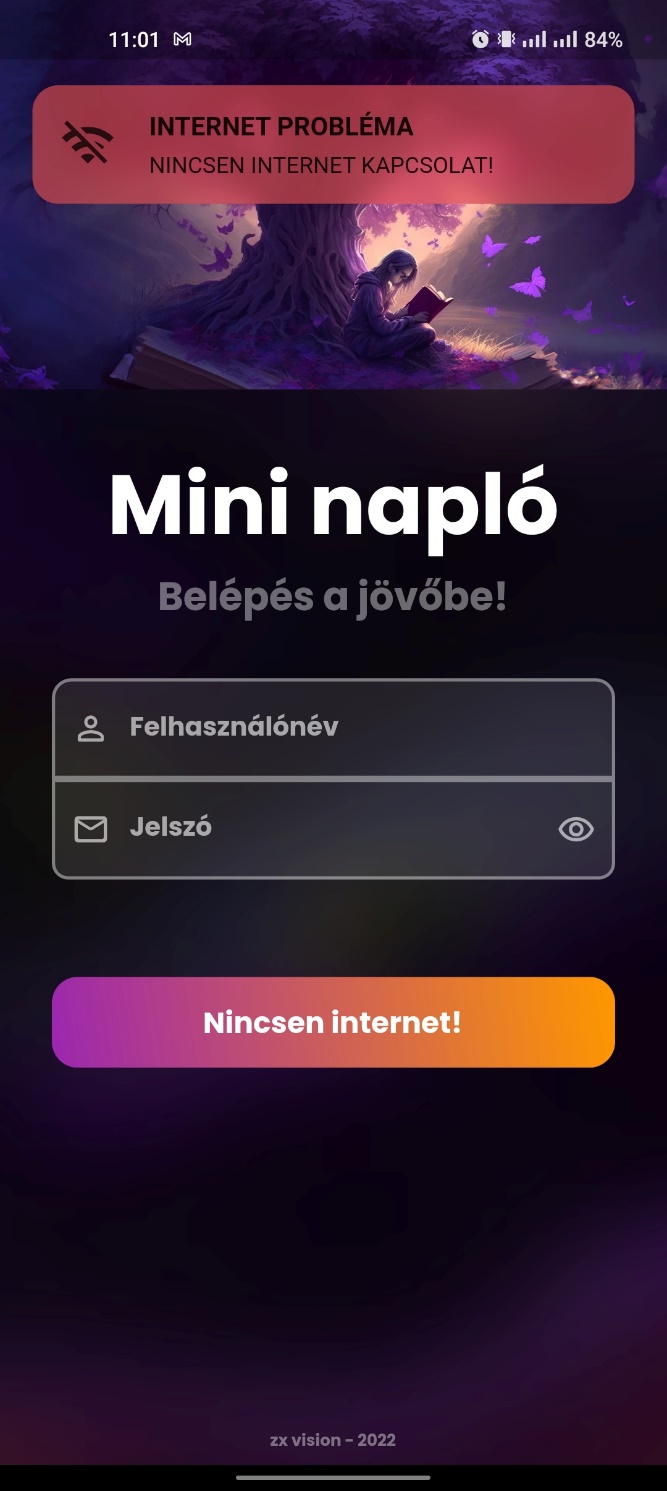
Felléphető hibák:

* Üres felhasználónév szövegdoboz
* Üres jelszó szövegdoboz
* Hibás felhasználónév vagy jelszó (szerver hívásnál derül ki)
* Nincsen internet

Hibák megjelenítésé a felhasználónak:

A felléphető hibákat a gomb szövegének kicserélésével jelenítődnek meg.

Felhasználó felület hiba megjelenítéséi:



Sikeres bejelentkezést követően a felhasználó átkerül a főoldalra, és itt az alsó navigációs menüvel lehet lépegetni az oldalak között.

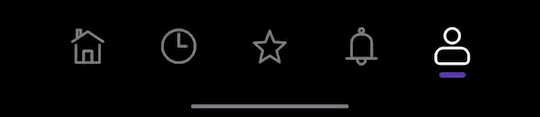
Az alsó menün az adott oldalak lesznek elérhetőek:

Oldalak:

* Főoldal
* Órarend
* Értékelések
* Hiányzások
* Felhasználó profil

A legtöbb oldalon hasonló felépítés található, egy fejléc, tartalom és az alsó navigációs menüből áll

Az alsó Navigációs menü felépítése:



Minden oldalt egy ikon jelez, ami rá kattintáskor válik aktívvá.

Rákattintás után a kiválasztott ikon színé felerősödik, és alatta megjelenik egy kis lila aláhúzás ezek szerepe, hogy a felhasználó mindig tudja, hogy melyik oldalon is jár.

Ezek mellett, egy animáció is elindul kattintáskor.

### Főoldal felépítésé



A képernyő tetején a fejléc látható, ahova tartózkodik az adott oldal neve, alatta egy halványabb felirat tudatja a felhasználóval, hogy miket is lát az adott oldalon.

A fejléc alatt van a tartalom rész, ahol úgy nevezett csempéket láthat a felhasználó.

Egy csempe egy tanár által beírt jegyet jelenít meg.

Alul az alsó navigációs menü található, ami minden oldalnál megtalálható elem.

Mindezek mögött egy Rive animációs háttér van, amiben 3 pötty mozgását lehet megfigyelni.

Egy csempe felépítésé:

A csempe enyhén átlátszó ezáltal lehet látni a háttérben lévő animációt.

Csempén megtalálható tartalom:

* Leírás
* Tantárgy
* Osztályzat

### Órarend oldal felépítésé



A fejlécen az oldal neve, és hogy mit is lát pontosan a felhasználó, ez mindig az aktuális napot jeleníti meg.

Ezek után látható az aznapi órarend

Ennek az oldala egy különböző Rive animációt használ, mint a többi. Ez a háttér még egy statikus képet is használ. Ami igy néz ki:



Minden óra egy ’csempe’. Ami itt is ezt az úgynevezett üveghatást hozza, ezért nem takarja ki teljesen a mögötte lévő animációt



Egy csempe felépítése:

* Óra száma az aktuális napon
* Tantárgy
* Mettől meddig tart az aktuális óra
* Ikon, ami mindig a tantárgyhoz kapcsolódik

### Értékelések oldal felépítésé:

A képen szöveg, elektronika látható

Automatikusan generált leírás

A képen szöveg, elektronika látható

Automatikusan generált leírás

A fejlécben ismét az oldal címe.

A képen szöveg, elektronika látható

Automatikusan generált leírás

Ezen az oldalon más az elrendezés mivel itt két fő panelre van bontva

A fent látható képen az első panel látható. Ezen a panelen egy kör grafikon helyezkedik el, ami azt mutatja, hogy melyik jegy fajtából mennyi darab van a felhasználónak. Az elrendezése, a grafikon darabszámokkal ellátva, és a jobbszélen a jegyek.

A képen szöveg, elektronika látható

Automatikusan generált leírás

A második panelen tantárgyak szerint látható az átlag, amit két tizedes pontossággal jelenítődik meg felhasználónak, és ezek csökkenő sorrendben jelenítődnek meg. Minden átlaghoz egy szín is társítható a kerekítés függvényében. ennél a panelnél is megfigyelhető az érdekes felépítést tekintve, mivel szintúgy átlátszósággal játszik, de a tantárgyak között kis rések figyelhetőek meg, és az elsőnek a felső jobb és bal oldalán megtalálható kerekítések, és ez csak az elsőnél van, illetve az utolsó két oldalánál az alsó és a felső elem között csak téglalap formák jelenítődnek meg

### Hiányzások oldal kinézete



Fejlécben az oldal elnevezése

Alatta két darab csempe helyezkedik el az egyiken az igazolt hiányzások öszessitése látható a felhasználó számára, míg a másikon az igazolandó hiányzások.

Ezután tanárgyanként listázva látható az adott tantárgyról való hiányzástokat.

Tantárgy hiányzás panelt 3 részre lehet osztani, az első rész ami azt mutatja hogy mennyi órát hiányozott a felhaszáló, a második része a tantárgy neve, és a harmadik eleme az ikon

### Felhasználó profil oldal felépítése



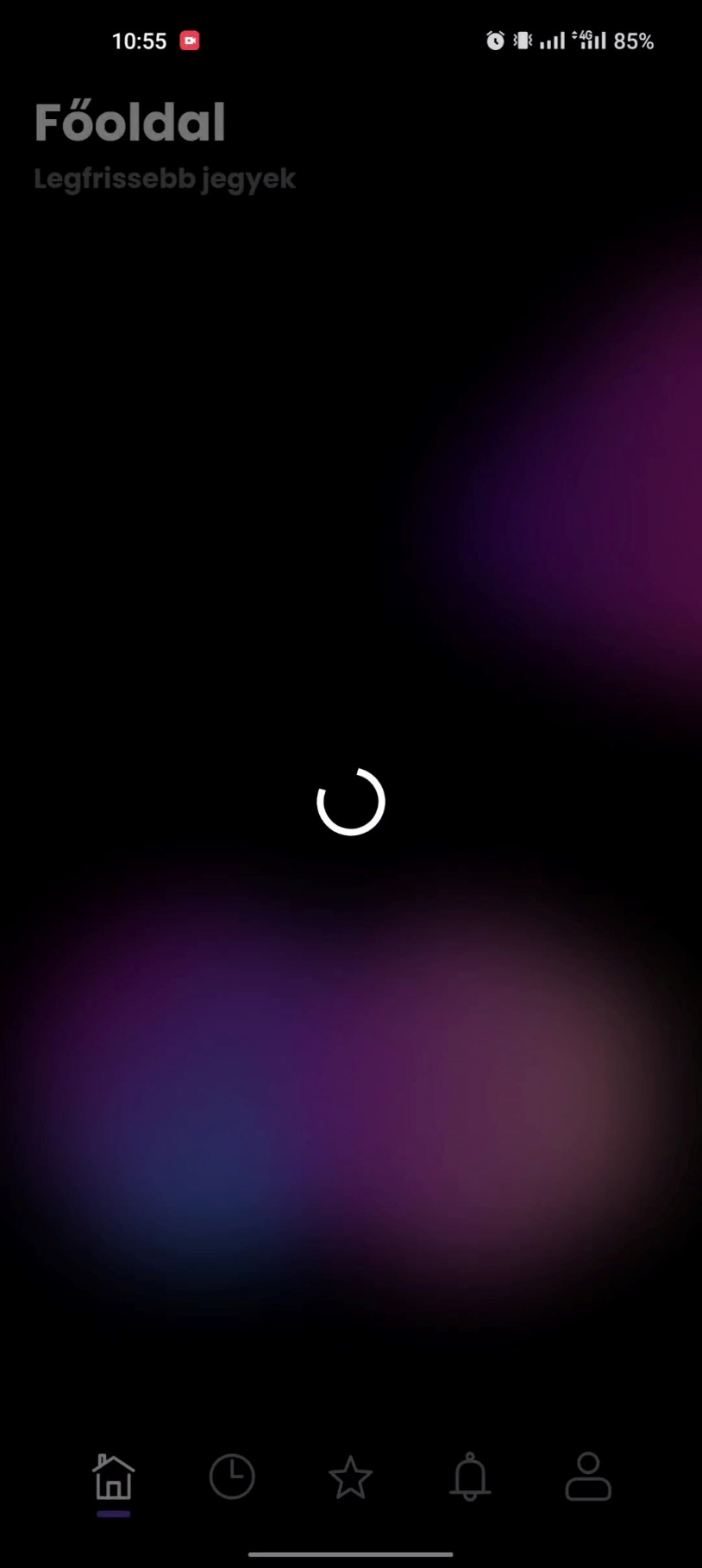
A fejlécben vélhető jelentősebb eltérés az előző oldalakhoz képes, az oldal neve helyett dinamikusan változóan megjelenítődik a felhasználó neve,

alatta egy statikus profil szöveggel.

A fejlécben továbbá megtalálható egy kijelentkezés gomb, ami visszadob a bejelentkezés képernyőre.

A fejléc alatt az adott tanulóról jelenítődnek meg az információk külön csempéken.

### Felhasználói felület kiegészítő elemei

Amint az alkalmazást a felhasználó már sikeres belépés után bezárja, és következő kívánt használatnál újra nyitja egy kis középre igazított töltés indikátort lát, ez addig látható amig az adatok bekerülésre nem kerülnek, ez alatt enyhe sötétítés kerül a felhasználó felületre.

Adat frissítés

Az adat frissítse egy kézmozdulattal történik, mint minden modern telefonos alkalmazásban manapság.

Működésé főoldalon újjal történő lehúzással aktiválható.

Kinézete a kép tetején látható, lehúzásnál egy kis animáció játszódik le és ezután lép a képen látható állapotába.

Amint végződik az adatok frissítése az oldal felépítése visszatér az eredeti elrendezéséhez.

Internet hiánya indikátor

Az alkalmazás a kréta szervereiről kéri le az adatokat ezért elhanyagolhatatlan, hogy legyen internet kapcsolat, mivel internetet igényel a bejelentkezés, adatok lekérése, és adatok frissítése, ezért fontos, hogy tudassuk a felhasználóval, hogy miért nem tud elérni egyes funkciókat, mint azt a bejelentkezés képernyőn is láthatóvá teszi a fenti indikátor és még a gomb felirata is.

Amint az internetelérés megszűnik, megjelenítődik egy beúszó animációval a képen látható indikátor, ez minden oldalon állandóan látható, ha az adatok már bevannak töltve akkor az oldalak között nyugodtan navigálhatunk, de a jelzés tudatja velünk, hogy nem lehetséges az adatok frissítése, mivel nincsen internet kapcsolat.

Amint az internetelérés újra vissza ált az indikátor azonnal kiúszik a képernyőről, és a funkciók ismét elérhetővé válnak.

# **Fejlesztői dokumentáció**

## Specifikációk

A projekt elkészítéséhez Flutter keretrendszert választottam, ami a Dart programozási nyelvel párosul. Adatbázisnak a Hive adatbázis rendszert választottam, ami nem relációs adatbázis(noSql). Az alkalmazás egy előre elkészített ikon csomagot használ, ami letölthető, ez a 3dicons néven található meg.

De mi is az a Flutter?

Flutter egy nyílt forráskódú, felhasználói felületeket építő framework, amely lehetővé teszi, hogy egyetlen kódbázis segítségével készítsük el az alkalmazásunkat minden platformon, beleértve az iOS-t, az Androidot, a webet és a desktopot is. A Flutter alkalmazásokat a Dart programozási nyelvvel írják, amelyet a Google fejlesztett ki.

A Flutter nagyon népszerűvé vált az elmúlt években azáltal, hogy egyszerűbbé tette a mobilalkalmazások és a felhasználói felületek fejlesztését. A Flutter nagyobb rugalmasságot biztosít a fejlesztőknek, mert a platformok közötti átmeneteket automatikusan kezeli, így a fejlesztőknek nem kell külön-külön megírniuk a kódot minden egyes platformhoz. A Flutter előnyei közé tartozik az, hogy gyorsabb alkalmazásfejlesztést tesz lehetővé, a felhasználói felületek magas szintű testreszabását, valamint az, hogy a fejlesztőknek nem kell ismerniük több programozási nyelvet vagy platformot. A Flutter-t sok nagyvállalat használja, például a Google, a Alibaba, a Tencent és a BMW is. Az ilyen vállalatok általában olyan nagyméretű alkalmazásokat fejlesztenek, amelyeket több platformon kell futtatni.

Flutter app felépítése

A Flutter működése az ún. "widgetek" használatán alapul, amelyek a felhasználói felületelemek építőkövei. A widgetek különböző formátumokban és méretekben jelenhetnek meg, és különböző adatokat, mint például szövegeket, képeket és gombokat jeleníthetnek meg. A widgetek összetételével lehet összerakni az alkalmazások felhasználói felületét, és különböző animációkkal és hatásokkal lehet őket tovább testre szabni.

A Flutter nagyon népszerű fejlesztők körében, mert lehetővé teszi, hogy az alkalmazásokat gyorsan és hatékonyan fejlesszék ki, miközben magas szintű testre szabást biztosítanak a felhasználói felületeken.

## Bevezetés

Az alap eltervezés, hogy készítsek egy modernebb krétát. Mi is kell ennek elkészítéséhez? Először is megfelelő eszközt kellet kiválasztani a feladathoz, sok keresgetés, és utána járás után a flutter-re eset a választásom, az érve, hogy platform függetlenül tudom építeni az alkalmazásomat, de mit is jelent ez? Egészpontosan azt, hogy ha akarom Windows-os környezetben futtatom, vagy a weben, illetve IOS-on is vagy Androidon is bármikor le buildelhetném, és futtathatnám, Az egyediség kedvéért maradtam az Androidnál. A keretrendszer megvan, mi kell még? Az a rövid válaszom, hogy nagyon sok minden, bele gondolva miket is kell megvalósítani a program „backend-jének”? Felhasználó interakciókat kezelni, Szerver hívásokat végezni, Az alkalmazás állapotát menedzselni, Adatokat tárolni, és még sok mást. Belegondolva egy komplex „piac kész” alkalmazást kellett építeni. A bevezetést olvasva sem hangzik egy egyszerű programnak, és nem is az, ebben a fejlesztői dokumentációban egy le egyszerűsített bevezetést mutatok a program kódjába.

### Problémák

Milyen problémákba ütköztem az alkalmazás készítése közben?

Kezdném azzal, hogy a flutter a dart programozási nyelvet használja, ezzel csak az a baj, hogy még a projektem elött még nem is ismertem, tehát egy nyelv sajátosságait, illetve szintaxisát kellett elsajátítanom.

Második problémám az volt, hogy hogyan is érjem el az adott tanuló adatait a kréta szervereiről (jegyeit, órarendjét, stb)

Továbbá még problémák:

* hogyan mivel menedzseljem az alkalmazás state-jét
* mivel tároljam az alkalmazás működéséhez szükséges adatokat

### Megoldások

Szerencsémre nem nekem kellet vissza fejtenem a szerver kéréseket, hanem egy kis utána járás után rátaláltam egy github repositoryra ahol egy részletes dokumentáció található a vissza fejtett api-ról, szerencsémre egy python file-t is tartalmazott amiben szinte mindent szerver hívást implementált a készítő, amit szinte csak átkellet írnom Dart kódra és úgymond már van is egy működő kódom ami letudja kérdezni az adataimat, a kréta szervereiről.

A state menedzselésnek a flutter által biztosított StatefullWidget választottam elsőnek,

de ez hamar bebizonyosodott, hogy ez a megközelítés nagyon limitált funkciókat biztosít. Szétnéztem, hogy mivel is tudnám megvalósítani amire nekem szükségem van, és eleinte nem tudtam melyik state menedzsert válaszam, mivel, ha megnézzük mennyi opció közül lehet választani flutter oldalán feltüntetettek közül akkor 16 darab lehetőség közül lehet választani. Mindegyiknek megvan az előnye és a hátránya is.

Rengeteg video és kutatómunka után a GetX-re esett a választásom. Pár érv, hogy miért:

* Egyszerű szintaxis
* Teljesítmény
* Beépített modulok (Route menedzser, egyszerű widget-ek)

Egy olyan adatbázist választottam, amit a flutter fejlesztők szeretnek használni, ajánlani ez pedig nem más, mint a Hive, és miben tér el egy közismert sql-től? Egészpontosan abban, hogy ez egy nonsql adatbázis, tehát nincsenek relációk, legegyszerűben úgy lehet elképzelni, mint egy szótárat, van egy kulcs és ahhoz tartozik az érték. Mivel az alkalmazás nem tárolja a tanuló adatait, hanem mindig, ha aktuális egy szerver hívassal tölti a memoriába, ezért nincs szükség annak tárolására, igy csak a felhasználó neve, és a jelszavat kellett eltárolnom, igy tökéletesen megfelel a célnak a Hive. Előnyei közül itt is a teljesítményt és az egyszerűséget emelném ki.

### Projekt mappa szerkezete

A lib nevezetű mappa tartalmazza a program futásához szükséges összes kódot

Mappák ismertetése:

* bindings – Ez a mappa a state menedzseléshez tartozó kontrollereket tartalmazza, és ezeket az alkalmazás indításakor „inject” -eli a globális scope-ba.
* constants – A mappa kibontva több az alkalmazásban található állandó / konstans színeit, icon elérési útjait és felhasználó felület részeit (pl: betűtípusát).
* pages – A navigálható oldalak UI kódját tartalmazza, ezen felül minden oldalnak külön „widget”-jeit, esetleges adat modelljét.
* routes – Két program fájlt tartalmaz, az egyik a oldal url konstansokat, a másik fájl pedig a GetX route menedzsmentjét kezeli az alkalmazáson belül.
* screens – Azokat a UI kódokat tartalmazza, ami befedi az egész kijelzőt, ezért „képernyő” (screen), ebben a mappán összesen 3 darab screen található (error, login, main).
* services – Ez a mappa tartalmazza az úgy nevezett „bloc”-okat, azaz business logic component ez azt jelenti, hogy minden komponens, ami a logikát kezeli az appon belül.
  + api - Mappában található minden olyan kód, ami a szerverhez való kéréseket intézi (kód implementálási kommenteket az \*\_interface.dart fájlokban találhatóak).
  + client - Ebben a mappában az a kód található, ami a szervertől kapott adatokat modellekbe rendezi.
  + controllers - Itt található minden kontroller, ami interakcióba lép a felhasználó felülettel, például a kontroller, ami a bejelentkezésért felelős „login\_controller.dart”.
  + models – A szervertől visszakapott adatot (JSON), strukturált adat formátumban tárolja.

Végül egy main.dart fájl található, mely az első fájl, ami a program futtatásakor indul el. Ezzel kezdődik az alkalmazás elindulása, de erről később részletesebben.

### Statisztika

Ez a projekt a lib mappán belül jelen állapotában 47 darab .dart végződésű fájl található, amikben összesen 2687 sor program kódot írtam.