Eötvös Loránd Tudományegyetem

Informatikai Kar

Informatikatudományi Intézet

Numerikus Analízis Tanszék

Kvízalkalmazás fejlesztése mobileszközre

Témavezető:

Dr. Pödör Zoltán

Egyetemi docens

Szerző:

Horváth Ádám

Programtervező informatikus BSc.

Szombathely, 2023

Szakdolgozat Témabejelentő

A szakdolgozat címe: Kvízalkalmazás fejlesztése mobileszközre

A szakdolgozat témája:

A szakdolgozat célja egy kvízalkalmazás fejlesztése mobil alapokon, ami a(z) ELTE szombathelyi kampuszához és az ELTE SEK PTI szakhoz kapcsolódik. Az alkalmazás Androidos készülékekre lesz fejlesztve Flutter [.dart] programozási nyelven, FireBase adatbázis háttér alkalmazásával.

A tervezett név: ELTELearn, melyben kvízeket lehet kitölteni, amik az ELTE SEK történetéről és egyéb elért eredményeiről/céljairól, és legfőképpen a PTI képzésről szólnak, ahhoz kapcsolódnak.

Az alkalmazás 3 fő komponensre lesz bontva:

1. Információk, melyek az egyetem történetéről, és azon belül a PTI szakról szólnak.
2. Kvízek, melyeket különféle szempontok szerint csoportosítva találhatja meg majd a felhasználó. A kvízeket csak regisztrált felhasználók tölthetik ki, míg a többi menüponthoz bejelentkezés nélkül is hozzá lehet férni.
3. Hírek, ahol a(z) (aktuális egyetemi) híreket lehet megtekinteni, melyek egymás alatt lesznek felsorolva hasonlóan, mint a Google Hírek.

Az alkalmazás lehetőséget nyújt arra, hogy mobil eszközön is elérhetőek legyenek az ELTE SEK-kel kapcsolatos hasznos információk, a kvízek pedig játékos formában teszik lehetővé ezen információk átadását.

Tartalomjegyzék

1. Bevezetés 1

2. Felhasználói dokumentáció 2

2.1. Alkalmazás indítása 2

2.2. Be- és kijelentkezés 3

2.3. Kvízek 4

2.3.1. Kvíz kitöltés menete 5

2.4. Történetek 6

2.5. Hírek 7

2.6. Kapcsolat 8

2.7. Sötét mód 9

2.8. Egyéb jelzések 10

3. Fejlesztői dokumentáció 11

3.1. Fejlesztői környezet 11

3.1.1. Programozási nyelv, - felület 11

3.1.2. Háttéradatbázis 11

3.1.3. Programozási nyelv – Háttéradatbázis kapcsolata 12

3.2. Dart programozási nyelv 13

3.3. Projekt létrehozása 15

3.4. Projekt felépítése 16

3.5. Adatbázis 19

3.5.1. Kezdőlépések 19

3.5.2. Autentikáció (Authentication) 23

3.5.2.1. Beállítás 23

3.5.2.2. Felhasználók 24

3.5.2.3. Bejelentkezés 25

3.5.2.4. Kijelentkezés 26
3.5.3. Firestore Adatbázis (Firestore Database) 27
3.5.3.1. Rules 27
3.5.3.2. Data 28
3.5.3.3. Modellek 32
3.5.4. Tárhely (Storage) 34
3.6. Harmadik féltől származó könyvtárak (3rd-party libraries) 36
3.7. Egyedi Widget-ek 37
3.8. Utils 37
4. Összefoglalás és további fejlesztési lehetőségek 38
5. Ábrajegyzék 40
6. Irodalomjegyzék 43

1. Bevezetés

A szakdolgozat célja egy olyan mobilalkalmazás elkészítése volt, amelynek segítségével bárki könnyen és gyorsan hozzáférhet az ELTE SEK (ELTE Savaria Egyetemi Központ) és azon belül legfőképpen a PTI szakos hallgatók egyetemi életének, napjaiknak legfontosabb információihoz. Az ELTELearn nevű alkalmazás úttörő szerepet vállal azzal, hogy lehetőséget nyújt az ELTE SEK-hez kapcsolódó hasznos információk és hírek mobil eszközökön való könnyed elérésére.

Az ELTELearn alkalmazás Android platformra készült, és a Flutter (.dart) keretrendszer segítségével fejlesztettem, melynek a programozási nyelve a Dart. A Flutter egy nyílt forrású, könnyen tanulható keretrendszer [1], amelyet a Google készített. Az alkalmazás adatbázisát a FireBase [2] biztosítja, amely szintén a Google által fejlesztett és nagyon jól együttműködik a Flutter keretrendszerrel. A FireBase adatbázis alkalmazása biztosítja, hogy az applikáció használata könnyen és gyorsan történhessen, valamint a felhasználók számára az alkalmazás nagyon stabil és megbízható legyen.

Az alkalmazás készítésének ötlete egy korábban készített kvíz alkalmazásból származik, amelyet azért készítettem, hogy gyakorolhassam a programozást ezen a nyelven. Azonban felmerült az ötlet, hogy hogyan tudnám újra alkotni ezt az alkalmazást úgy, hogy segítsen az egyetemre jelentkezőknek és hallgatóknak, valamint az egyetemnek is.

Így alakult ki az ELTELearn alkalmazás, melynek célja, hogy egyedi megoldást nyújtson az ELTE SEK-hez kapcsolódó információk mobil eszközön való elérhetőségének terén. Az alkalmazásban a hallgatók és jövőbeli hallgatók, jelentkezők megtalálják a legfontosabb információkat, híreket, eseményeket, jegyzeteket és hasznos linkeket az egyetemmel és a képzésekkel kapcsolatban. Ezenfelül az alkalmazás játékos kvízekkel lett színesítve, melyekhez kapcsolódó információkat, olvasmányokat is meg lehet találni az alkalmazás különböző menüpontjaiban. Az alkalmazás segítségével a felhasználók könnyen kapcsolatba léphetnek az egyetemmel, a képzési programokkal és a hallgatói élettel kapcsolatos információkkal.

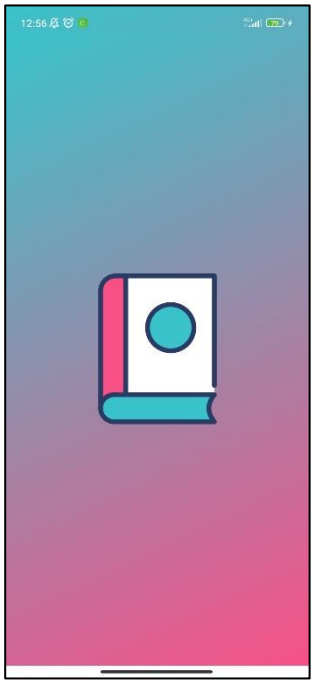
Az alkalmazásnak egyedi és könnyen használható a felülete, valamint a benne található hasznos információk lehetővé teszik, hogy az ELTE SEK hallgatói, valamint a jövőbeli hallgatók és jelentkezők könnyedén és gyorsan megtalálják azokat az információkat, amelyekre szükségük lehet. Az ELTELearn alkalmazásnak köszönhetően az ELTE SEK diákjai bármikor és bárhonnan egyszerűen hozzáférhetnek az egyetemi életükhöz kapcsolódó fontos információkhoz. Így az alkalmazás akár a beiskolázás támogatásához is felhasználható a jövőben.

2. Felhasználói dokumentáció

2.1. Alkalmazás indítása

Alkalmazásnak az ikonja, amely a letöltése után látható a mobileszközünk kijelzőjén.

1. ábra: Alkalmazás Ikon



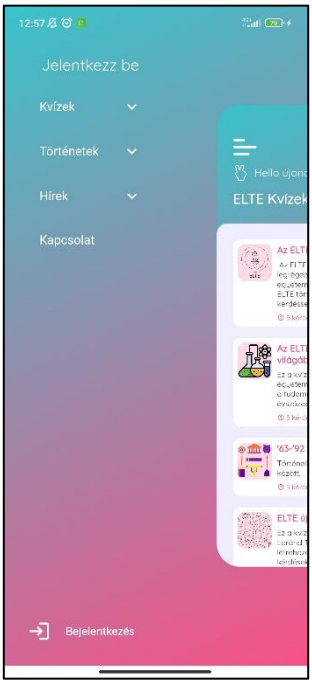
2. ábra: Splash Screen

3. ábra: Bemutatkozó oldal

Az alkalmazás kezdőfelülete bárki számára elérhető. Indítás után, először egy egyszerű és letisztult felület, az ún. Splash Screen (2. ábra) fogadja a felhasználót, amelyen egy kép található az alkalmazásnak a logójáról, majd 3 másodperc múlva továbbít a Bemutatkozó oldalra (3. ábra). Ezen az oldalon van egy rövid bemutatkozó leírás az alkalmazásról, és a nyíl megnyomásával tovább jutunk a kezdőfelületre. Az alkalmazás néhány funkciója csak regisztrációt, majd bejelentkezést követően érhető el.

2.2. Be- és kijelentkezés

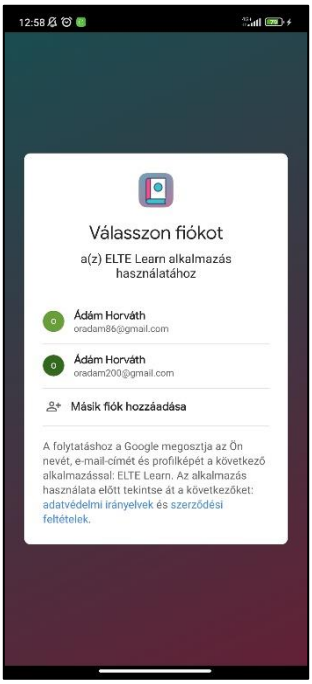
4. ábra: Kezdőfelület



5. ábra: Menüsáv

4. ábrán látható az a felület, ahova kezdésként mindig vezet az alkalmazás.
Bal felül a menü ikonra kattintva juthatunk el a Menüsávra (5.ábra), ahonnan a többi oldalra léphetünk át és/vagy be- és kijelentkezhetünk.
Egyedüliként a kvízek kitöltéséhez szükséges a regisztráció, bejelentkezés, a többi menüpont szabadon megtekinthető és használható.

6. ábra: Bejelentkezés

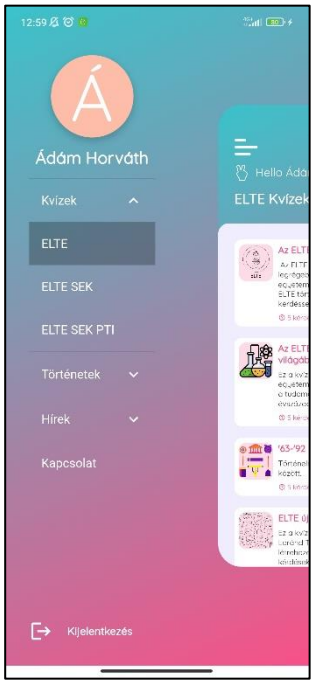


7. ábra: Google bejelentkezés

8. ábra: Bejelentkezett felhasználó

Az 5. ábrán látható **Jelentkezz be** vagy **Bejelentkezés** gomb megnyomásával elnavigál az alkalmazásban a 6. ábrán látható oldalra, ahol Google-el lehet regisztrálni/belépni. Majd ki kell választani a felhasználót (ha több van), amivel szeretnénk belépni, és a sikeres bejelentkezés után láthatóak a felhasználó (8.ábra) regisztráció során megadott adatai. A **Kijelentkezés** megnyomásával a felhasználó ki lesz jelentkeztetve, amely a 8. képen látható.

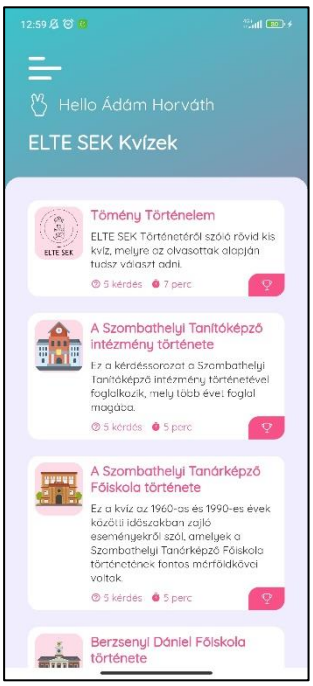
2.3. Kvízek



9. ábra: Kvízek menü

Az alkalmazás egyik fő elemét a kvízek jelentik, amiken keresztül játékosan szerezhetünk információt, illetve tesztelhetjük tudásunkat a SEK-ről, annak történetéről, a szombathelyi programtervező informatikus képzésről. A kvízek lenyíló menüpontban (9. ábra) találhatóak meg csoportosítva, értelemszerűen kvízek. 3 csoportra van bontva, melyek az ELTE, ELTE SEK és az ELTE SEK PTI. Mindegyik felületen külön-külön több kvíz van jelen, melyek kitöltésére csak bejelentkezés, regisztrálás után van lehetőség.

10. ábra: ELTE kvízek



11. ábra: ELTE SEK kvízek

12. ábra: ELTE SEK PTI kvízek

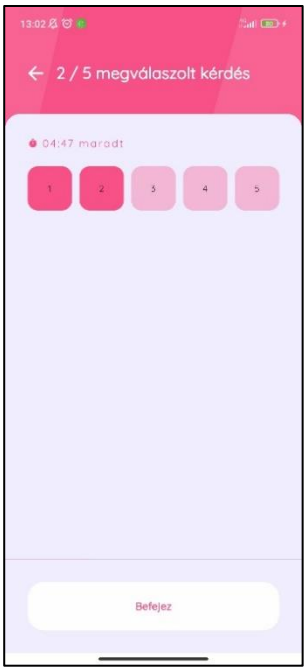
Mind a három említett almenüben kvízeket találunk (10., 11., 12. ábra), melyek az adott témához kapcsolódnak. Minden kvíznek van egy-egy képe, címe, és egy rövid leírása, és ezek alatt az látható, hogy hány kérdésből áll, majd mellette, hogy mennyi idő van a kvízek kitöltésére. A kvízek oldalain ezenkívül még látható felül egy üdvözlő szöveg, és alatta pedig az aktuális menüpont, amibe belépett a kliens.

2.3.1. Kvíz kitöltés menete

13. ábra: Első kérdés

14. ábra: Utolsó kérdés

A kvízeket időre lehet kitölteni, ennek megfelelően az alkalmazás felületén látható, hogy van egy visszaszámláló, amely a megadott perctől kezd visszaszámolni, mellette a kérdésnek a sorszáma (K 01). Ezektől jobbra van még egy menüpont (9 db pont), ahol a kérdéseket lehet megtekinteni (15.ábra), hogy hányadiknál járunk, és melyikre adtunk választ. Lejjebb haladva a kérdések, és alattuk a válaszlehetőségek, majd a tovább és a visszalépésre szolgáló gombok kaptak helyet.



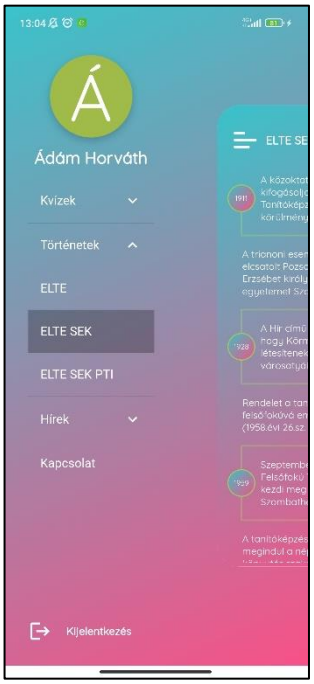
15. ábra: Kérdések megtekintése

16. ábra: Helyes válaszok

17. ábra: Válaszok megtekintése

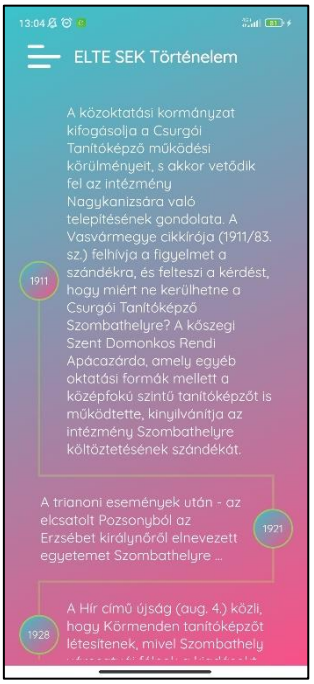
A legutolsó kérdésnél a tovább gomb kicserélődik a befejez gombra, mely megnyomása után átvezet a 15. ábrán látható oldalra, hogy a felhasználó még egyszer le tudja ellenőrizni a leadás előtt. Miután le lett adva, akkor kiértékelésre kerül a kvíz és ez látható a 16. ábrán. Ahol ugyanis felül és a kvízek színein (zöld, piros) látható, hogy hány kérdésre adott a felhasználó helyes választ. Ha rányomunk a kérdésekre a kiértékelés oldalán (16. ábra), akkor megmutatja, hogy mi volt a helyes válasz. Majd visszajutunk a főoldalra.

2.4. Történetek



18. ábra: Történetek menü

19. ábra: Történet



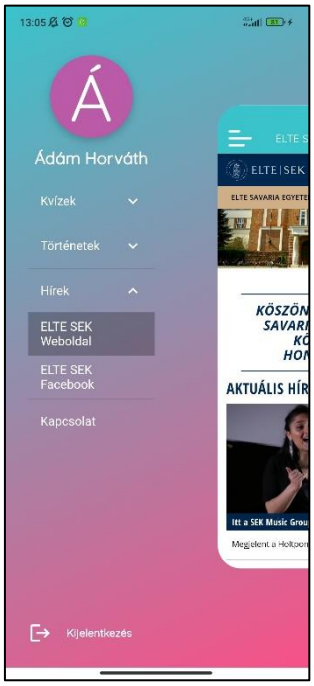
20. ábra: Történet kibontása

Történetek menüpont célja, hogy a felhasználó az alkalmazás keretein belül ismereteket szerezzen a SEKről, a Szombathelyen folyó programtervező informatikus képzésről.

A menü 3 részre van osztva, ugyan úgy, mint a kvízek, mivel ezek szorosan kapcsolódnak egymáshoz. És pedig azért kötődnek egymáshoz szorosan, mert a történetekben olvashatóak a kvízek helyes kitöltésére szolgáló információk. Ezek az információk/történetek kinézetének definiálására egy úgynevezett idővonal szolgál, mely kibontható. Ez azt jelenti, hogy ha több mint 4 sornyi szöveg van egy-egy ilyen időpontnál, akkor ezt pontokkal jelöli a végén. A szöveg megérintésével ez kinyitható, és ezáltal a többi szöveget is meg lehet nézni. Ezek megtekintéséhez nem szükséges bejelentkezni.

Az történeteket, információkat a hivatalos oldalakról érkeznek, amelyek az ELTE [3], ELTE SEK [4] [5], ELTE SEK PTI [6] [7] oldalak.

2.5. Hírek



21. ábra: Hírek menü

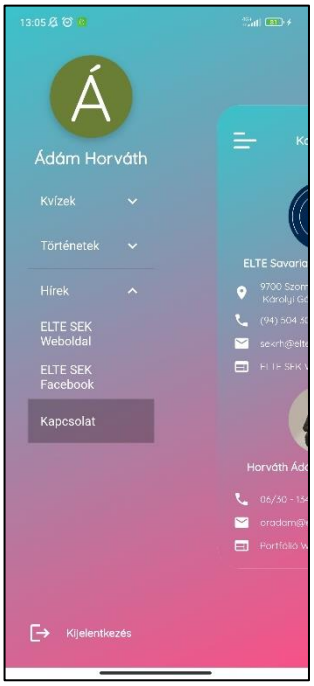
A hírek menüpont (21. ábra) célja, hogy a felhasználó valóban aktuális információkhoz is hozzáférjen az alkalmazáson keresztül. A hírek 2 részből áll, melyek az ELTE SEK weboldalára és a Facebook csoportjára mutatnak. Ezek ugyanúgy regisztráció nélkül is megtekinthetőek, ugyanis itt találhatóak meg a legfrissebb egyetemi információk, és legfőképpen a Facebook információk, ahol posztok formájában láthatóak a legfrissebb hírek. Ezek az oldalak egy ún. webes nézetekben jelennek mindig, melyre webview_flutter csomag szolgál. Ezek ugyan úgy jelenítik meg, mintha telefonon néznénk meg az adott oldalt.

22. ábra: ELTE SEK Weboldal

23. ábra: ELTE SEK Facebook

A 22. ábrán látható ahogyan az ELTE SEK Weboldala (22. ábra) jelenik meg a korábbiakban leírt webes nézetben. Itt általában olyan információk találhatóak meg, melyeket nem kell nap mint nap frissíteni, ezzel ellentétben a Facebook csoportban (23.ábra) viszont mindig az aktuális hírekről tájékozódhatunk.

2.6. Kapcsolat



24. ábra: Kapcsolat menü

25. ábra: Kapcsolatok

A kapcsolat menüpontnál (24.ábra) az egyetemhez és a fejlesztőhöz köthető kapcsolatfelvételhez szükséges információk találhatóak meg. Ilyen például a helyzet meghatározása, telefonszám, email, és weboldal. Ez az oldal is megtekinthető regisztráció/bejelentkezés nélkül. (25.ábra)

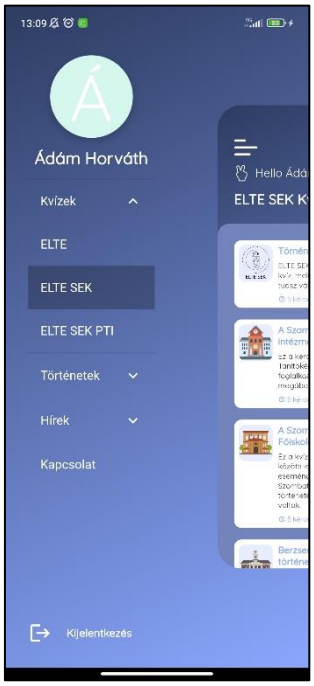
26. ábra: Helyzet

27. ábra: Email

28. ábra: Weboldal

Ha rányomunk a helyzet meghatározásra, emailre, weboldalra, akkor mindegyik elirányít a megfelelő oldalra. Az email megnyitja a rendszer levelezőjét (27.ábra). A helyzet meghatározás (26. ábra) pedig megnyit egy OpenStreetMap [8] nevű térképet, ahol meg lehet tekinteni, hogy hol helyezkedik el az egyetem, lehet nagyítani, kicsinyíteni, és visszaállítani a kezdő pozícióra a nézetet. Az oldal megnyomásával pedig az adott weboldalra irányít (28.ábra).

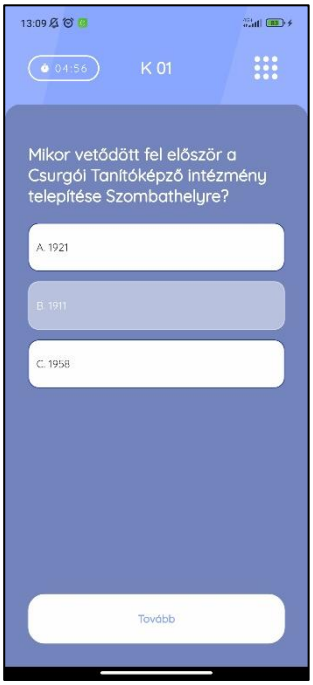
2.7. Sötét mód



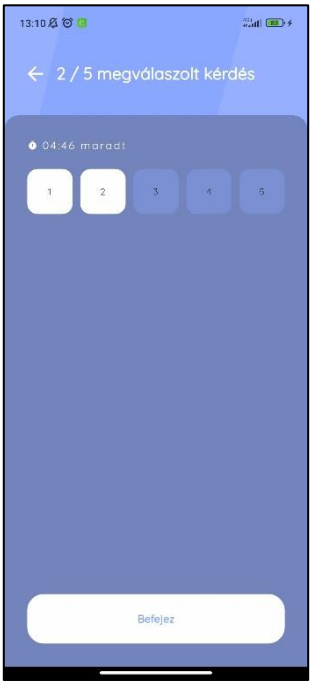
29. ábra: Sötét mód - Menü

30. ábra: Sötét mód - Történetek

31. ábra: Sötét mód - Kvízek



32. ábra: Sötét mód - Kérdések



33. ábra: Sötét mód - Megválaszoltak

Egy gyakori funkció is implementálva lett az alkalmazásban, amely arra szolgál, hogy a felhasználók választhassanak az alkalmazás felhasználói felületének világos vagy sötét színsémája között, ezt angolul Light/Dark mode-nak hívják. Ez a pár kép, az alkalmazás főbb menüpontjairól készült, melyek bemutatják, hogy sötét módba váltáskor hogyan néz ki. A váltást a mobiltelefonunk rendszerében található Világos/Sötét módváltóval lehet megtenni, ami általában az értesítési panelen látszik egy gomb formájában.

2.8. Egyéb jelzések

34. ábra: Nagyított kép

Ezek a jelzések különböző interakcióknál jelentkeznek, úgy nevezik, hogy Dialógusok (Dialogue).

Egyik ilyen interakció például, amikor a kvízek menüpontok közül az egyiknél járunk, akkor minden egyes kvíznek, ha rákattintunk a képre (34.ábra), akkor van lehetőség a kinagyított változatának a megtekintésére.

35. ábra: Lejárt az idő

36. ábra: Kvíz megkezdése sikertelen

Az alsó két ábrán (35. és 36.) látható balról jobbra haladva, hogy az első képen egy olyan dialógus jelenik meg, amin a „Lejárt az idő!” szöveg van. Ez akkor jön elő, amikor egy kvízt megkezdünk, de az idő lejár, és még nem fejeztük be. A második képen meg az az eset van, amikor nincs regisztrált felhasználó, ezáltal nem tudja megkezdeni a kvízeket. Ekkor az OK megnyomásával eljutunk a bejelentkezési platformra.

3. Fejlesztői dokumentáció

3.1. Fejlesztői környezet

A fejezetben bemutatom azokat az eszközöket, fejlesztői környezeteket, illetve komponenseket, melyeket az alkalmazás fejlesztése során felhasználtam.

3.1.1. Programozási nyelv, - felület

- **Flutter** [9]**:** A Flutter egy nyílt forráskódú mobilalkalmazás-fejlesztési keretrendszer, amelyet a Google hozott létre. A Dart programozási nyelvet használja, és lehetővé teszi nagy teljesítményű, vizuálisan vonzó alkalmazások létrehozását iOS és Android eszközökre egyaránt. Ezen felül képes Windows, Linux operációs rendszerekre is készíteni alkalmazásokat, szoftvereket.
- **Visual Studio Code** [10]**:** A Visual Studio Code (VS Code) egy ingyenes, nyílt forráskódú kódszerkesztő, amelyet a Microsoft fejlesztett ki. Modern webes és felhőalkalmazások építésére, hibakeresésére tervezték, és számos programozási nyelvet és keretrendszert támogat, beleértve a Flutter (Dart), C++, C#, Java, JavaScript, Python és még sok más nyelvet. Különböző kiegészítőkkel (Extensions) vagy már alapból beépített funkciókkal könnyedén személyre szabható a szerkesztőnek a felülete, ezen felül nagyon sok olyan funkció is van, mellyel a fejlesztés gyorsaságát és hatékonyságát lehet növelni, ilyen például:
    - IntelliSense: Kódkiegészítő eszköz, amely gépelés közben javaslatokat ad kódrészletekre, ezáltal gyorsítja a fejlesztési folyamatot.
- **Android Studio** [11] **/ XCode** [12]**:** Ezek a hivatalos kódszerkesztők, amiket a Google és az Apple fejleszt(ett), kifejezetten Androidra és/vagy iOS (macOS,watchOS, …) – re. Mindkét platformra ugyanazokat lehet elmondani, mint a Visual Studio Code-ra funkcionalitást tekintve. A kapcsolat a 3 platform között az, hogy az Android Studio-t és/vagy az XCode-ot legalább telepíteni kell, hogy utána össze lehessen kapcsolni a VSCode-dal, hogy aztán abban fejleszthessünk. Ez nem feltétlen szükséges, csak azért lehet érdemes megcsinálni, hogy ne 2 külön szerkesztőt használjunk, hanem csak 1-et és azt szokjuk meg.

3.1.2. Háttéradatbázis

- **Firebase** [13]**:** A Firebase egy mobil- és webes alkalmazásfejlesztési platform, amelyet a Firebase Incorporated (Inc.) hozott létre, amely ma a Google leányvállalata. Számos szolgáltatást nyújt mobil- és webes alkalmazások építéséhez és kezeléséhez, beleértve a valós idejű adatbázisokat, a hitelesítést és a tárhelyet. Maga a platform a Google Cloudra épül, ami azt jelenti, hogy a Google által kínál felhőalapú szolgáltatásokat használja (biztonság, adattárolás, hálózat, stb…).

Egy – két szolgáltatás, amit a Firebase nyújt (fent említettek):

- *Valós idejű adatbázis:* A Firebase valós idejű NoSQL (not only SQL) felhőalapú adatbázist biztosít, amely képes valós időben tárolni és szinkronizálni az adatokat több ügyfél között. Ez azt jelenti, hogy az adatbázisban az egyik kliensen végrehajtott változtatások azonnal megjelennek a hozzá csatlakoztatott összes többi kliensen.
- *Hitelesítés:* A Firebase számos hitelesítési módszert biztosít a felhasználók hitelesítésére, többek között e-mail és jelszó, telefonszám és közösségi média bejelentkezés (Google, Facebook, Twitter stb.).
- *Tárhelyszolgáltatás:* A Firebase olyan tárhelyszolgáltatást nyújt, amely lehetővé teszi a fejlesztők számára, hogy webes alkalmazásaikat egyetlen paranccsal egy globális tartalomszolgáltató hálózatra (CDN – Content Delivery Network) telepítsék.

3.1.3. Programozási nyelv – Háttéradatbázis kapcsolata

- **Flutter – Firebase kapcsolata** [14]**:** A kettő könnyedén összekapcsolható, hogy a fejlesztők a Firebase szolgáltatásait használhassák a Flutter-alkalmazásaikban. Ezt a Firebase pluginek Flutterhez történő telepítésével lehet megtenni, amelyek hozzáférést biztosítanak a Firebase szolgáltatásaihoz, mint például a Firestore, a Firebase Authentication és a Firebase Storage. A bővítmények telepítése után a fejlesztők a rendelkezésre bocsátott API-k segítségével csatlakozhatnak a Firebase szolgáltatásaihoz, és olyan funkciókat építhetnek be alkalmazásaikba, mint például a felhasználói hitelesítés, az adattárolás és a valós idejű frissítések, anélkül, hogy a fejlesztőknek kellene kezelniük a szerveroldali logikát.

3.2. Dart programozási nyelv

A Dart [15] egy programozási nyelv, amelyet a Google fejlesztett ki, és 2011-ben jelent meg. Úgy tervezték, hogy könnyen tanulható és használható legyen, ugyanakkor olyan funkciókat is biztosít, amelyek a modern programozási nyelvekben általában megtalálhatóak.

A Dart statikusan tipizált nyelv, ami azt jelenti, hogy a változók típusait nem futásidőben, hanem fordításkor ellenőrzik. Ez segíthet abban, hogy a hibákat a fejlesztési folyamat során minél korábban megtaláljuk, és a kód megbízhatóbbá váljon.

A Dart számos modern nyelvi jellemzővel is rendelkezik, többek között:

- **Aszinkron programozás (Asynchronous programming)**: A Dart beépített támogatást nyújt az aszinkron programozáshoz, ami megkönnyíti a gyors reagálású és hatékony kód írását.
- Típuskövetkeztetés: A Dart támogatja a típuskövetkeztetést, ami azt jelenti, hogy a fordító automatikusan le tudja következtetni a változók típusait a használatuk alapján.
- **Mixins**: A Dart támogatja a mixineket, amelyek lehetővé teszik a kód újrafelhasználását több osztályhierarchiában.
- **Bővítési módszerek (Extension methods)**: A Dart lehetővé teszi kiterjesztési metódusok definiálását, amelyekkel új metódusokat adhatunk hozzá meglévő osztályokhoz anélkül, hogy azok forráskódját módosítanánk.
- **Elkülönítések (Isolates)**: A Dart támogatást nyújt az izolátumokhoz, amelyek olyan könnyített szálak, amelyek párhuzamosan futhatnak egymással. Ez megkönnyítheti a skálázható és hatékony párhuzamos kód írását.

A Dart a Flutter-alkalmazások fejlesztéséhez használt elsődleges programozási nyelv, és gyors és reszponzív mobilalkalmazások készítésére optimalizált. Emellett webes alkalmazások, parancssori eszközök és szerveroldali alkalmazások fejlesztésére is használható.

37. ábra: StatelessWidget

38. ábra: StatefulWidget

A Dart nyelv felépítése alapvetően Widget-ekből tevődik össze, melyek a nyelv specialitása. A Widgetek összetehető, fa-szerű felépítése lehetővé teszi a fejlesztők számára, hogy könnyen építsenek összetett és reszponzív felhasználói felületeket, míg a reaktív programozási modell biztosítja, hogy a felhasználói felület változásai automatikusan tükröződjenek az alkalmazás állapotában.

Két alaposztály létezik, amelyek a **StatelessWidget** (37. ábra) és a **StatefulWidget** (38. ábra). Az úgynevezett Widget-ek, ezeknek az alaposztályoknak a kibővítésére szolgálnak. Minden Widget paraméter-argument párokkal rendelkeznek, s ezek egymásba ágyazhatóak, amivel egy szerkezet jön létre, ami a 37. ábrán látható.

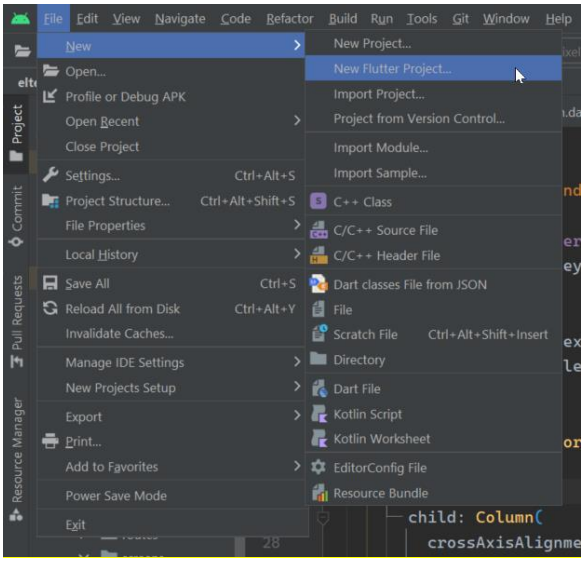
Például az első sorban látható **Scaffold**() egy Widget, melynek vannak paraméterei, ilyen a **body**, amelynek lehet argumentumokat megadni. Ezek az argumentumok lehetnek másik Widget-ek is, mint például a **Container**(), amelynek ugyancsak vannak paraméter-argumentum párjai.

```
Scaffold(
        body: Container(
                decoration: …
                color: …
        )
)
```

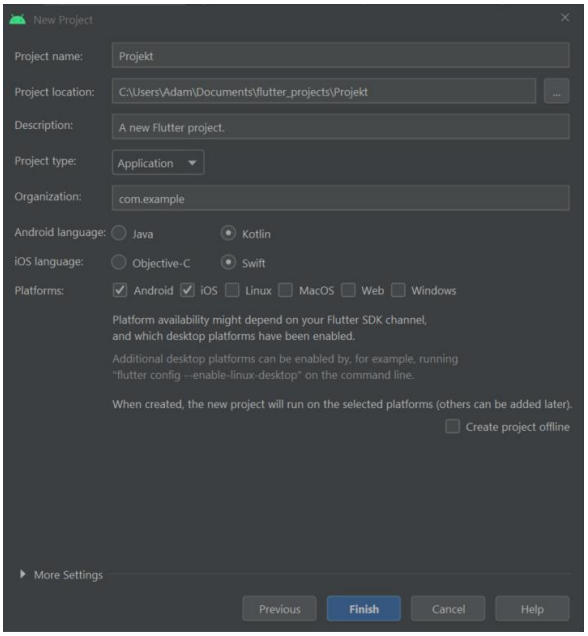
És lényegében ilyen elemekből épül fel egy alkalmazás kinézete (UI).

Összességében a Dart egy modern és sokoldalú programozási nyelv, amely a mobilalkalmazásoktól a szerveroldali kódokig számos alkalmazáshoz jól használható. Könnyű kezelhetősége és modern funkciói miatt népszerű választás a fejlesztők körében.

3.3. Projekt létrehozása



39. ábra: Projekt létrehozása



40. ábra: Projekt beállításai

Android Studioban Flutter projektek létrehozása a File -> New -> New Flutter Project… menüponttal vagy terminálban a flutter create <projekt_név> paranccsal lehetséges (39. ábra).

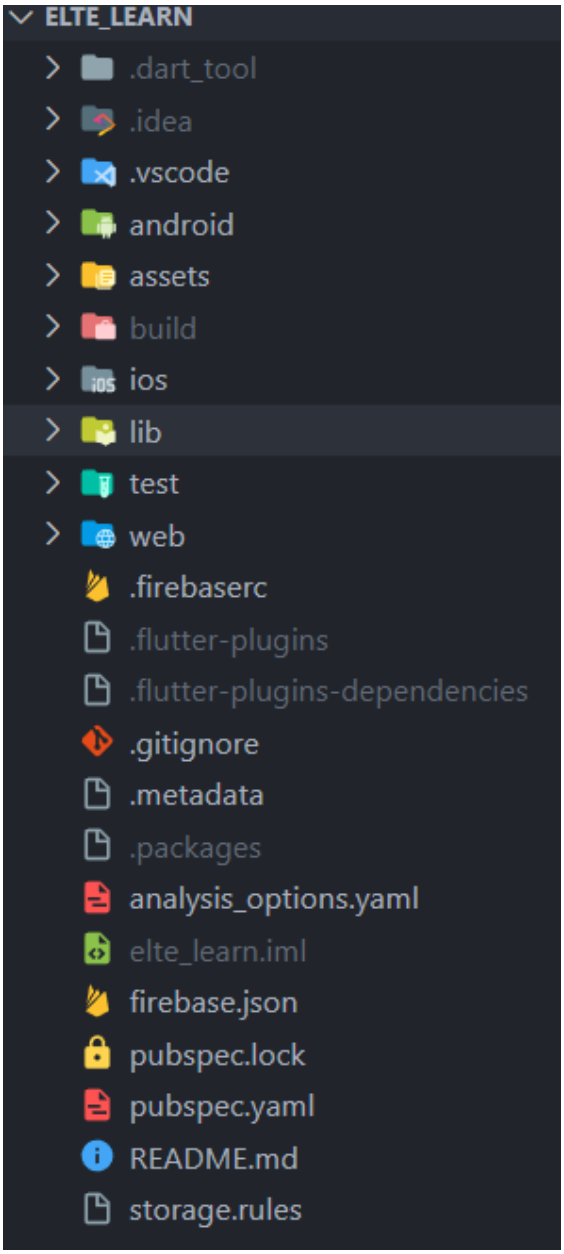
Ezek után van lehetőség beállítani a projekt fő paramétereit, melyek közül kiemelném az Android és IOS nyelveknél látható opciókat, amelyekből a fejlesztő a számára szimpatikusabb nyelvet ki tudja választani (40. ábra).

Amikor a Flutter segítségével Android és/vagy IOS-alkalmazást készítünk, a Dart kódot compiler-ek segítségével fordítja natív kóddá. És ezeket a natív nyelveket lehet kiválasztani, hogy melyikre fordítsa át a keretrendszer. De a Kotlin, Java, Swift, Objective-C nem maga a Dart programozási nyelv, ez annak a lefordított natív változata.

Ez azért fontos, mert ha platformspecifikus kódot kell írni, akkor a Dart kód és a platformspecifikus kód közötti kommunikációhoz platformcsatornákat használhat. Ez lehetővé teszi, hogy kihasználjuk az egyes nyelvek erősségeit, miközben a Flutterrel mégis platformok közötti alkalmazást építhetünk.

3.4. Projekt felépítése

A 41. ábrán egy alap kiindulási projekt tekinthető meg Flutter keretrendszerben, ezek között található olyan mappa vagy fájl, amit automatán legenerált, de nem használjuk. Ebben az esetben ilyen például az **ios** és a **web** mappa. Ezeket azért nem használjuk, mivel csak Androidos készülékekre lett tervezve az alkalmazás.
Ezek a mappák számunkra a legfontosabbak:



41. ábra: Projekt felépítése

.dart_tool: Ez a mappa a különböző Dart eszközök által használt fájlokat tartalmazza.

android: Ez a mappa tartalmazza az összes releváns adatot a fordításhoz és egy működő Android alkalmazás (apk vagy appbundle) létrehozásához.

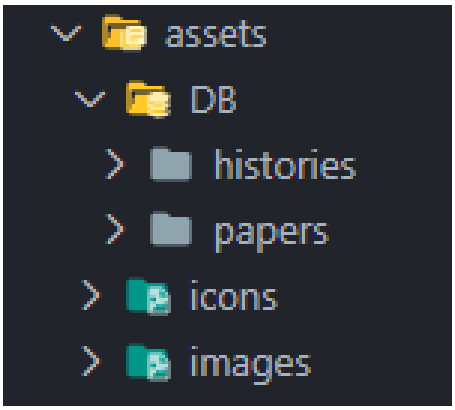
assets: Itt tárolódnak az alkalmazáshoz felhasznált források.

build: Akkor jön létre, amikor a Flutter build eszközei először kerülnek futtatásra. Ez tartalmazza az alkalmazás különböző platformokon történő futtatásához szükséges generált fájlokat. Minden platformnak saját almappája van.

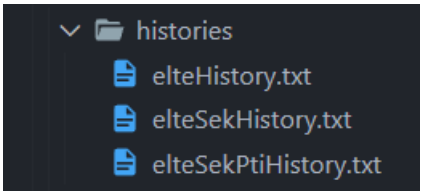
ios: Hasonló az android mappához, csak az ios-szel kapcsolatos fájlokat tartalmazza.

lib: Ez számunkra a legfontosabb mappa, mivel ebben található maga az alkalmazás kódja.

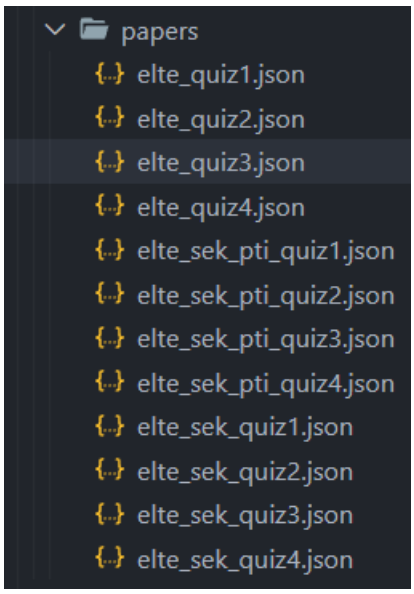
pubspec.lock/pubspec.yaml: A pubspec tartalmazza a projekt leírását, a korlátozásokat, a függőségeket és az eszközöket. Az összes 3rd party csomagot itt kell definiálni, hogy a „Flutter tools”-ok tudják, hogy melyik verziót kell lekérni.



42. ábra: Források



43. ábra: Történetek forrásai

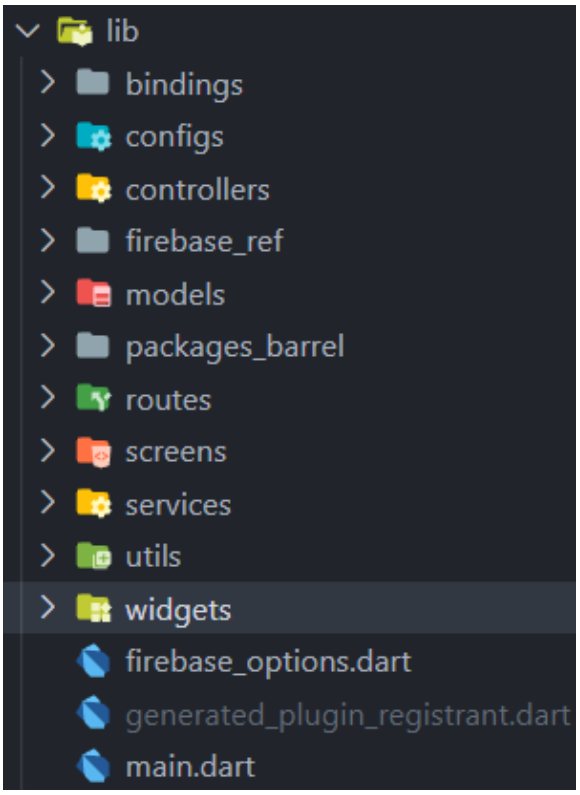


44. ábra: Kvízek forrásai

assets: Itt tárolódnak az alkalmazáshoz felhasznált források.

Az assets mappa 3 fő almappára van bontva, mely közül a DB is almappákra van osztva:

- **DB:** Az adatok tárolására szolgáló mappa, ahonnan feltöltődnek az fájlok a Firebase-be.
  - **histories:** Ebben a mappában találhatóak meg a „Történetek” menüpontban olvasható források. Ezek .txt fájlok, és lokálisan tárolódnak, hogy ha felhasználóknak nincs hozzáférésük internethez, akkor legalább tudjanak olvasni történeteket, mivel minden máshoz szükséges az internet kapcsolat.
  - **papers:** Itt találhatóak a kvízek, melyek a „Kvízek” menüpontokban vannak. Azért esett arra választás, hogy json fájlokban legyenek tárolva a kvízekhez szükséges adatok, mivel jól strukturálhatóak, ezáltal könnyen kezelhetőek és a Firebase ezt a fájlformátumot tudja kezelni.
- **icons:** Az alkalmazásban megtalálható ikonok vannak ide helyezve.
- **images:** Az alkalmazásban megtalálható képek vannak itt elhelyezve.



45. ábra: Alkalmazás kódja

- **screens:** Az alkalmazás felhasználói felületének (UI) képernyőit (screens) tartalmazza. Minden képernyő (screens) az alkalmazás egy-egy különböző nézetét képviseli, és tartalmazza az adott nézet megjelenítésének stílusát.
- **services:** Itt történik a Firebase Storage-ban tárolt adatok letöltésének implementációja.
- **utils:** Általában olyan segédosztályok és funkciók tárolására használják, amelyek nem kötődnek az alkalmazás egy adott Widget-jéhez vagy képernyőjéhez. Jelen esetben a konstans szövegek tárolódnak itt.
- **widgets:** Különböző egyedi, akár többször újra használható Widget-ek találhatóak meg ebben a mappában.

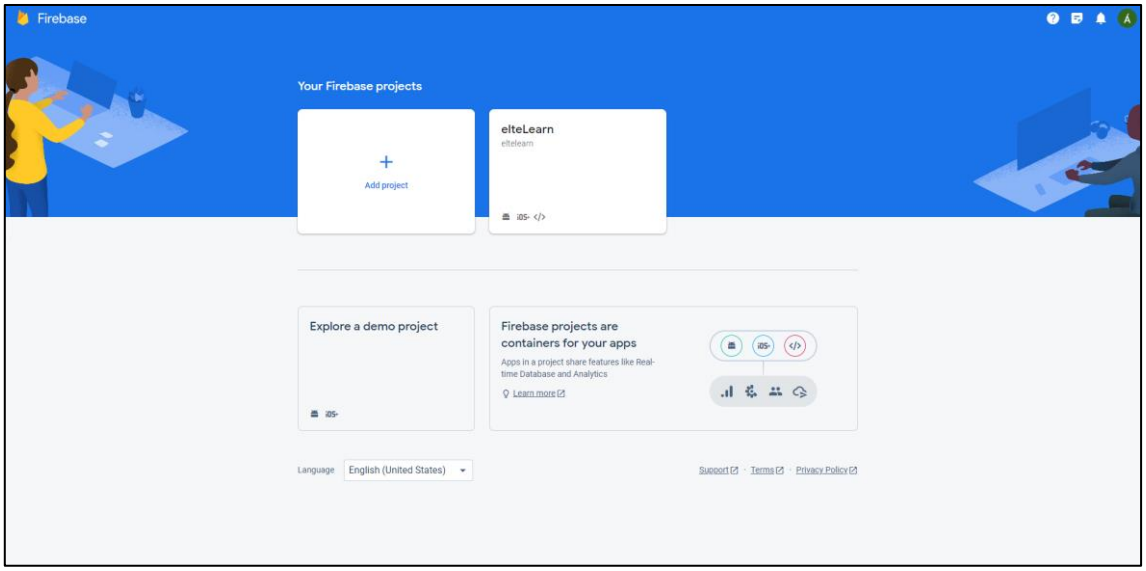
lib: Az egyik, ha nem a legfontosabb mappa, mivel ebben található maga az alkalmazás kódja.

- **bindings:** Ebben a mappában van egy olyan osztály, amelynek célja, hogy deklarálja azokat a függőségeket, amelyeket inicializálni kell, amikor egy adott útvonal vagy osztály meghívásra kerül a Flutter alkalmazásban.
- **configs:** Alapvető konfigurációk találhatóak meg, mint például az applikáció témája, vagy rendszerkonfigurációk.
- **controllers** A „screens” mappában található képernyőknek a funkcionalitásait tartalmazzák az itt lévő fájlok.
- **firebase_ref:** Firebase referenciák találhatóak meg, hogy ne kelljen mindig ugyanazt a sok sort ismételni.
- **models:** A bejövő adatok modelljei és/vagy ezek megfelelő formátumába konvertálásai találhatóak meg.
- **packages_barrell:** Használt „package” -k találhatóak meg egy fájlban, melyet ezek után csak 1 sorral elég importálni.
- **routes:** Az alkalmazás „útvonalait” tartalmazza, melyek különböző „screen”-ekre navigálnak.

3.5. Adatbázis

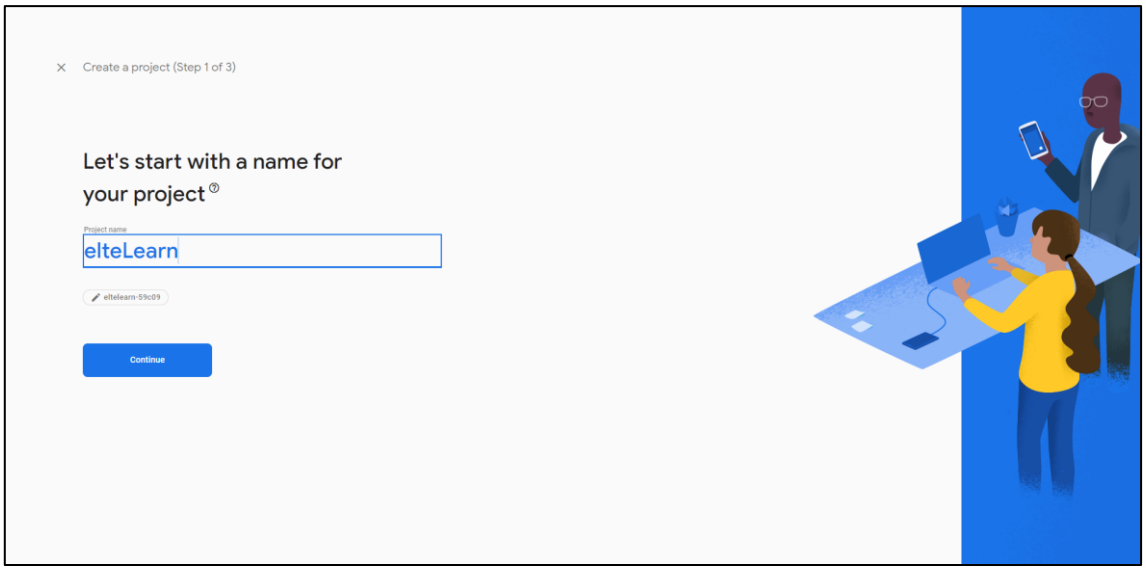
Ahogy korábban már a 3.1.2 Háttéradatbázisnál említettem a Firebase egy felhőalapú platform, amely különféle dologban segíthet az alkalmazások fejlesztése során és a fejlesztő számára gyorsabbá és könnyebbé teheti a fejlesztési folyamatot. Segít olyan dolgokban, mint például a hitelesítés, adatbázis-kezelés, autentikáció, üzenetküldés és még sok más. Ezek miatt a szolgáltatások miatt döntöttem úgy, hogy ez a környezet megfelelő számomra. Az alkalmazás adatbázisában a felhasználók és a hozzájuk tartozó adatok (email, jelszó, kitöltött kvízek, pontok), a kvízekhez tartozó képek, és a kvízek kérdései tárolódnak, melyeket könnyedén lehet módosítani.

3.5.1. Kezdőlépések



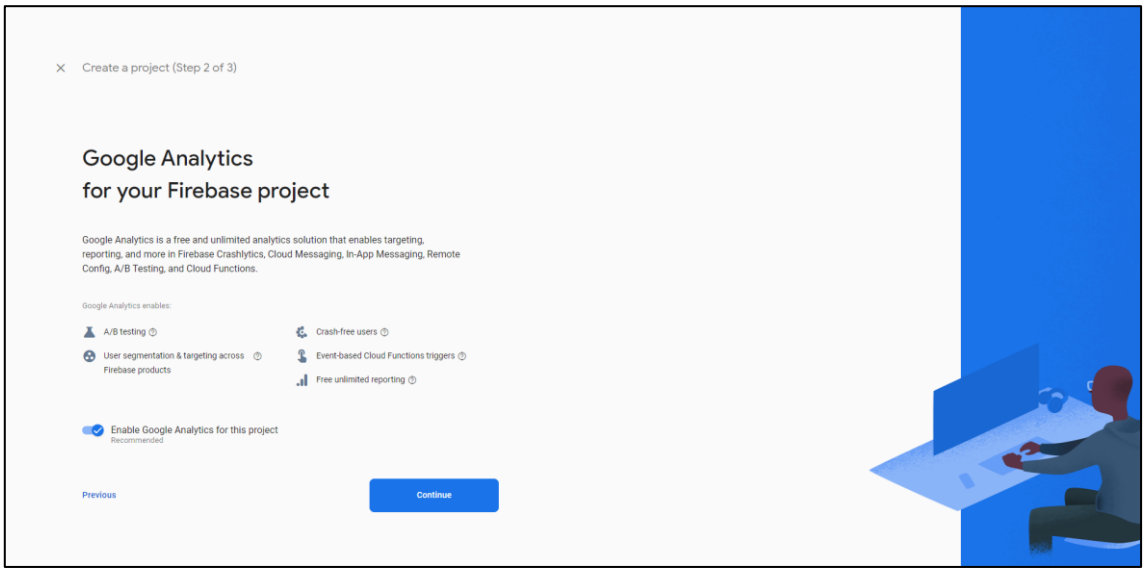
46. ábra: Kezdőlépések

1. lépés: Kezdésképpen ezen a felületen (Firebase Console [16]) kellett létrehozni egy projektet az **Add Project** megnyomásával.



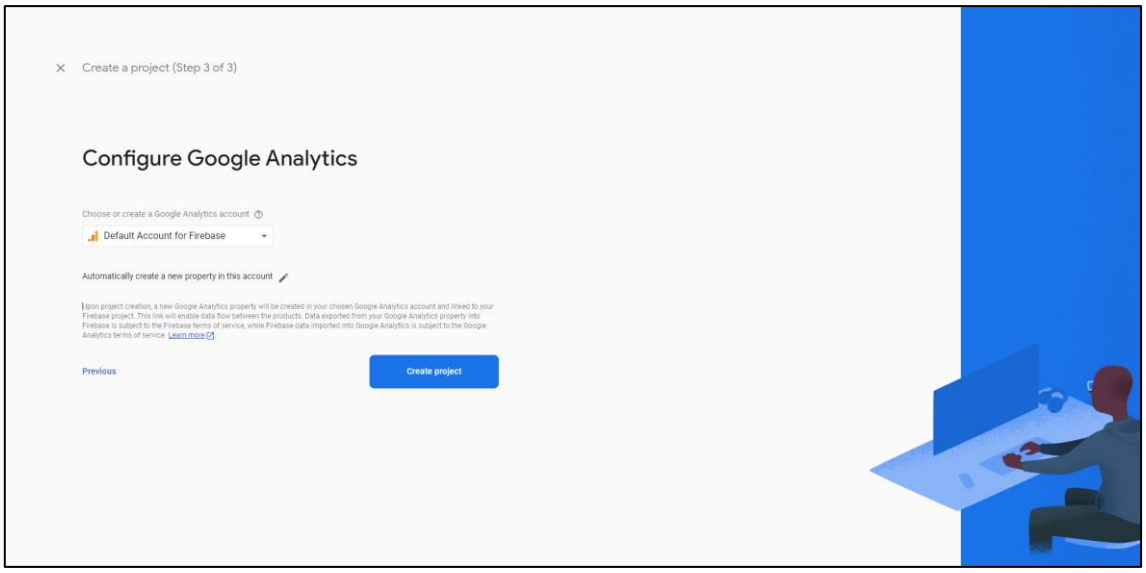
47. ábra: Project elnevezése

2. lépés: Project elnevezése



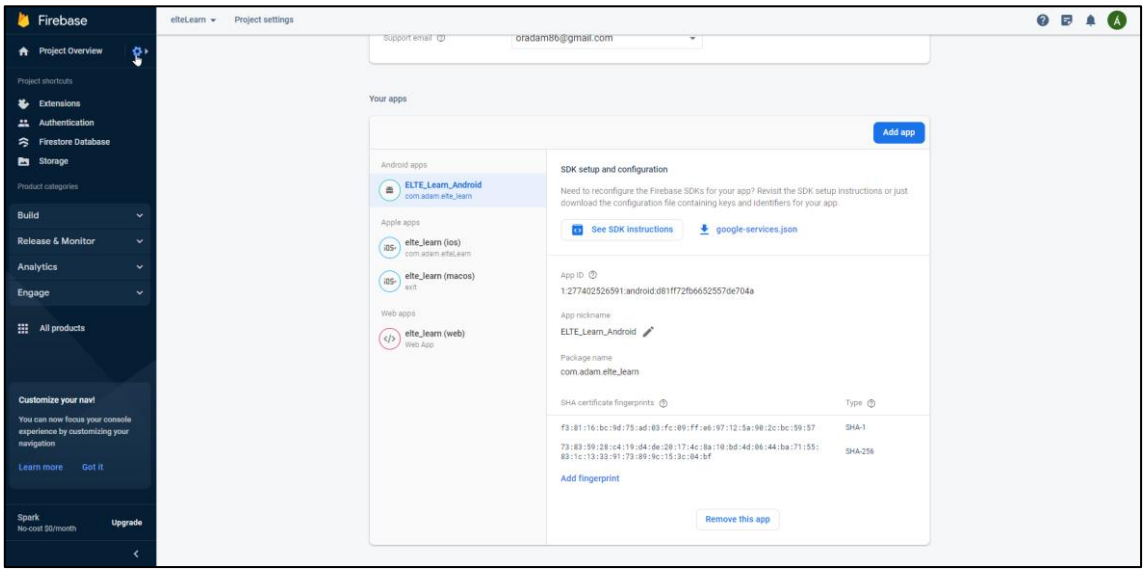
48. ábra: Google Analytics

3. lépés: Itt eldönthetjük, hogy szeretnénk-e a **Google Analytics** nevű szolgáltatás használni, amely a kezdetektől fogva különböző adatokat nyer ki a felhasználóktól (mikor lépett be a felhasználó, naponta hány felhasználó van, stb…)



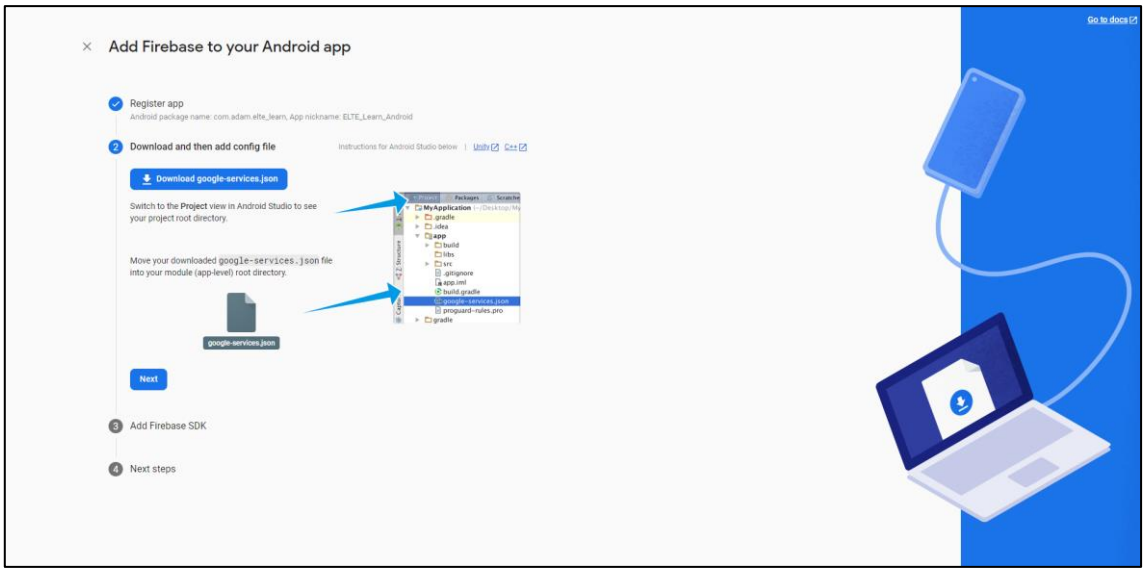
49. ábra: Google Analytics Konfiguráció

4. lépés: Ha a 3. lépésben beállítottuk a Google Analytics szolgáltatást, akkor ebben a lépésben be kell állítani a **Default Account for Firebase** felhasználót, és megnyomjuk a **Create Project** gombot.



50. ábra: App - Firebase kapcsolat

5. lépés: Be kell lépni a **Project Settings** menüpontba, ahol különböző adatokat (projekt neve, ID, stb…) láthatunk fentebb. Számunkra viszont a lentebbi rész a fontos, ahol be kell állítani, hogy milyen platform(ok)ra szeretnénk használni a Firebase szolgáltatásait.



51. ábra: Kapcsolat beállítása(i)

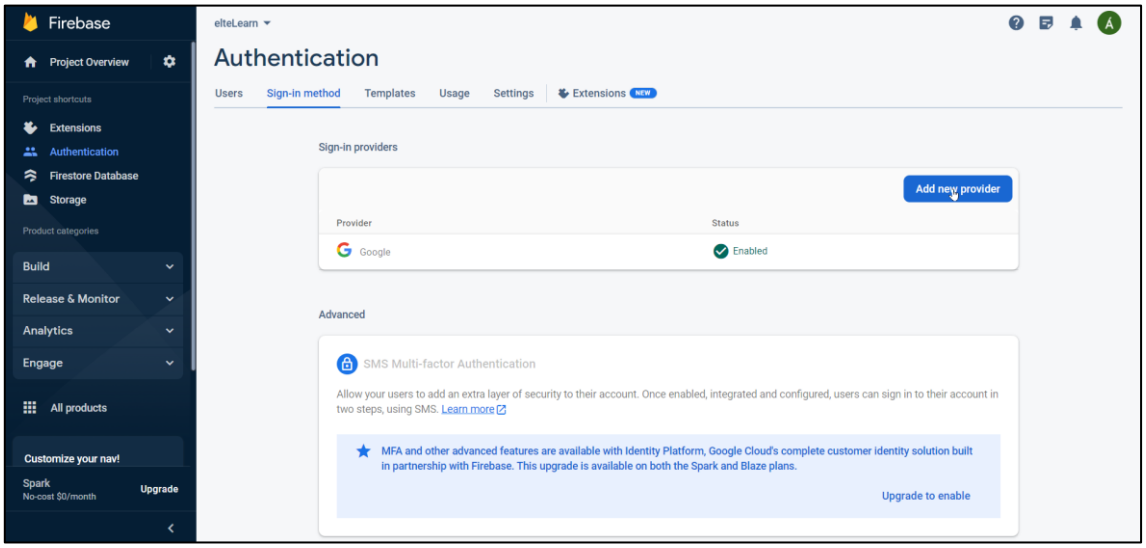
6. lépés: A **See SDK instructions** gombra kattintva megnyílik ez az oldal, amely lépésről-lépésre segít a szolgáltatást összekapcsolni az alkalmazásunkkal, programunkkal. Minden lépés sikeres végrehajtása után létrejött a kapcsolat, és ezáltal tudjuk majd használni az alkalmazásunkhoz a Firebase által nyújtott szolgáltatásokat.

3.5.2. Autentikáció (Authentication)

A Google Firebase autentikáció egy olyan háttértár-szolgáltatás, amely a fejlesztők számára biztonságos módot biztosít a felhasználók hitelesítésére webes és mobilalkalmazásaikban. Számos hitelesítési módszert kínál, többek között e-mail/jelszó, telefonszám, Google, Facebook, Twitter és más OAuth szolgáltatókat, ezenfelül kezelni tudja a felhasználókat, ami azt jelenti, hogy a regisztrációt, a bejelentkezést, a jelszó visszaállítását és a fiók törlését is eltudja végezni egyszerűen.

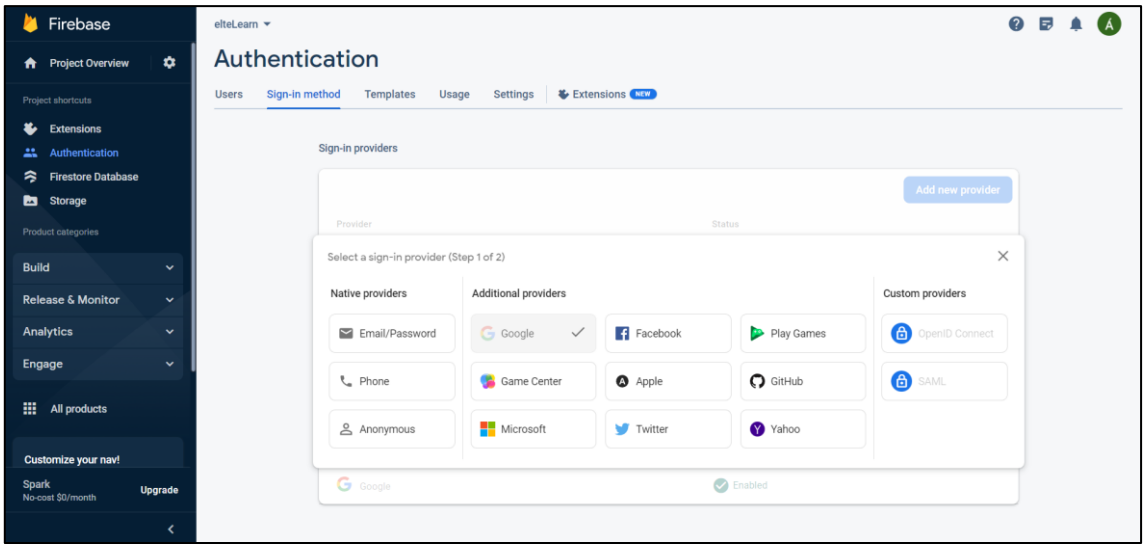
Használatának egyik jelentős előnye, hogy lehetővé teszi a fejlesztők számára, hogy hitelesítési funkciókat adjanak hozzá alkalmazásaikhoz anélkül, hogy szerveroldali kódot kellene írniuk.

3.5.2.1. Beállítás



52. ábra: Autentikáció

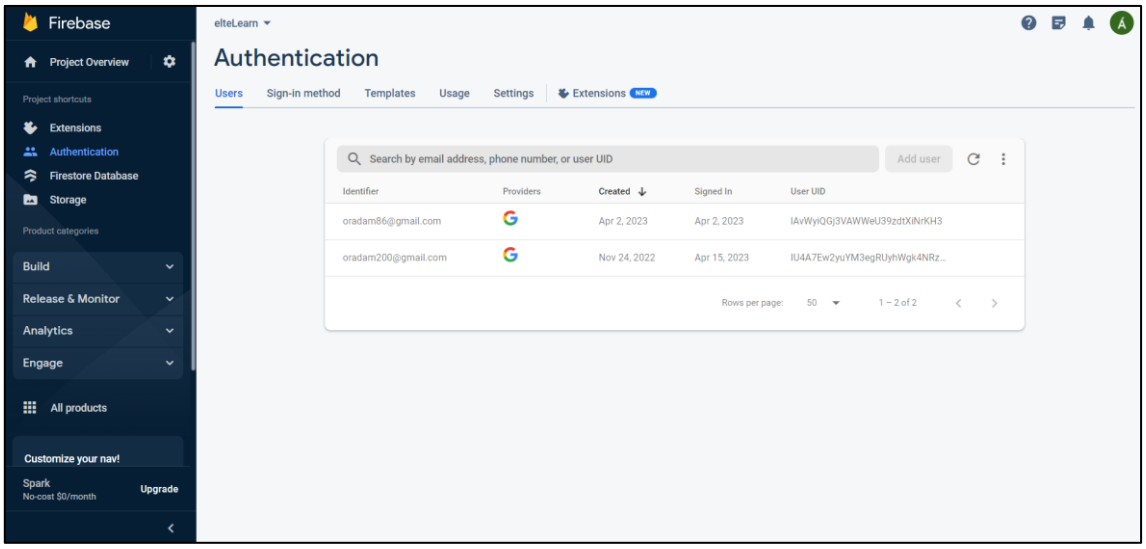
1. lépés: Az **Authentication** menüpont alatt a **Sign in Method** fülön belül az **Add new provider** gomb megnyomásával lehet hozzáadni a kiválasztott bejelentkezési formát.



53. ábra: Autentikáció típusai

2. lépés: Miután megnyomtuk az **Add new provider** gombot, megjelennek a felhasználható bejelentkezési formátumokat, és ezek közül választva egy könnyen és gyorsan megoldható bejelentkezést tudunk létrehozni alkalmazásunkhoz.

3.5.2.2. Felhasználók



54. ábra: Felhasználók

A felhasználók megtekintésére szolgál az **Authentication** menüpont alatt a **Users** fül, ahol minden felhasználónak az adatait tudjuk megtekinteni.

- **Identifier:** A felhasználónak a regisztrációnál megadott email címe vagy felhasználó neve látható.
- **Provider:** Azt a szolgáltatót láthatjuk, amivel a felhasználó bejelentkezett, jelen esetben a Google Authentication.
- **Created:** A felhasználó létrejöttének ideje, amikor a felhasználó regisztrált.
- **Signed In:** A legutolsó bejelentkezés időpontja.

3.5.2.3. Bejelentkezés

lib\screens\login**login_screen.dart**: A bejelentkezési felület kinézetének implementációja lelhető fel a **LoginScreen** osztályban.

```
Button(
  onTap: () => controller.signInWithGoogle(),
  child: Stack(
    children: [
      Align(
        alignment: Alignment.centerLeft,
        child: SvgPicture.asset(
          googleSvg,
        ),
      ),
      Center(
        child: AutoSizeText(
          googleSignInTitle,
          minFontSize: 18,
          style: TextStyle(
            color: Theme.of(context).primaryColor,
            fontWeight: FontWeight.bold,
          ),
        ),
      ),
    ],
  ),
),
```

55. ábra: Google bejelentkezés gomb

A **LoginScreen** osztályban található egy **Button** nevű egyedi Widget, amely a „Google bejelentkezés” gomb kénzetéért és funkcionalitásáért felel. A **funkcionalitást** ami a Google-el való bejelentkezést teszi lehetővé, a **controller.signInWithGoogle()** metódus végzi el.

lib\controllers**auth_controller.dart**: Az **AuthController** osztályban található meg minden olyan funkcionalitás, ami a felhasználó kezelésére szolgál, jelen esetben ez a **signInWithGoogle()** függvény.

```
Future<void> signInWithGoogle() async {
  final GoogleSignIn googleSignIn = GoogleSignIn();
  try {
    GoogleSignInAccount? googleSignInAccount = await googleSignIn.signIn();
    if (googleSignInAccount != null) {
      final googleAuthAccount = await googleSignInAccount.authentication;
      final googleAuthAccountCredential = GoogleAuthProvider.credential(
        idToken: googleAuthAccount.idToken,
        accessToken: googleAuthAccount.accessToken,
      );
      await _firebaseAuth.signInWithCredential(googleAuthAccountCredential);
      await saveUser(googleSignInAccount);
      navigateToHomePage();
    }
  } on Exception catch (e) {
    throw Exception("signInWithGoogle ERROR: $e");
  }
}
```

56. ábra: Google bejelentkezés funkcionalitása

A **signInWithGoogle()** egy aszinkron függvény, amely a Google Sign-In-t használja a felhasználó hitelesítésére, ha ez sikeres, a függvény elmenti a felhasználó adatait és ezután a kezdőlapra irányít.

3.5.2.4. Kijelentkezés

lib\controllers**auth_controller.dart**: Az **AuthController** osztályban található meg minden olyan funkcionalitás, ami a felhasználó kezelésére szolgál, jelen esetben ez a **signOut()** metódus.

57. ábra: Kijelentkezés funkcionalitása

58. ábra: Kijelentkezést segítő metódus

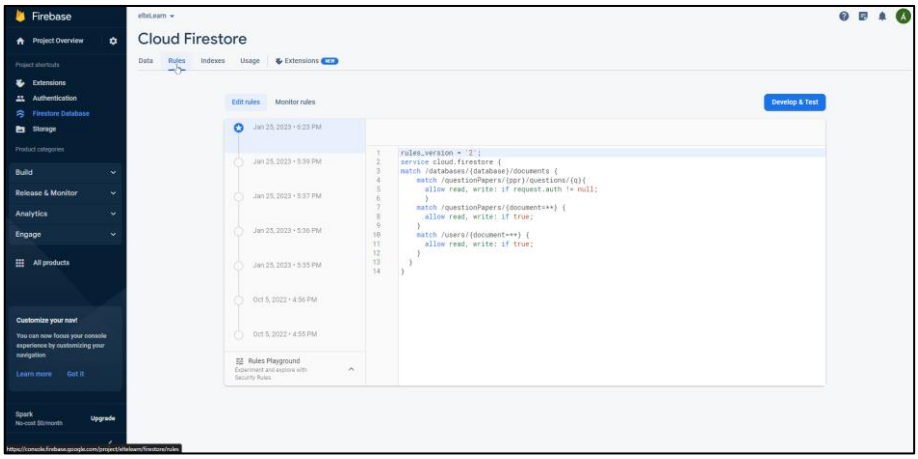
A **signOut (külső)** nevű metódus, arra szolgál, hogy kijelentkeztesse a felhasználót, ha szeretne kilépni. A **firebase_auth package** segítségével történhet meg a kijelentkeztetés, mely biztosít egy **signOut (belső)** metódust, ami egyszerűen és gyorsan kilépteti a felhasználót. Ha ez sikeresen megtörtént, akkor ugyan úgy vissza navigál a kezdőoldalra. Ha bármi hiba történne, akkor egy hiba üzenetet dob vissza.

3.5.3. Firestore Adatbázis (Firestore Database)

A Firestore egy felhőalapú NoSQL dokumentumadatbázis [17], amelyet a Google a Firebase platform részeként biztosít. Úgy tervezték, hogy valós időben tárolja és szinkronizálja az adatokat az ügyfelek és a szerverek között. A Firestore-t mobil- és webfejlesztők használják olyan skálázható és reaktív alkalmazások készítésére, amelyek valós idejű adatszinkronizálást igényelnek. A Firestore az adatokat JSON fájlformátumú dokumentumok gyűjteményeiként tárolja, amelyek egymásba ágyazhatók és hierarchikusan strukturálhatók. Minden dokumentum kulcs-érték párokat tartalmaz, ahol a kulcsok karakterláncok, az értékek pedig bármilyen támogatott adattípus, például karakterláncok, számok, tömbök vagy leképezések lehetnek. A dokumentumok gyűjteményekbe szerveződnek, amelyek hasonlóak a relációs adatbázisok táblázataihoz.

3.5.3.1. Rules

Firestore Database -> Rules:

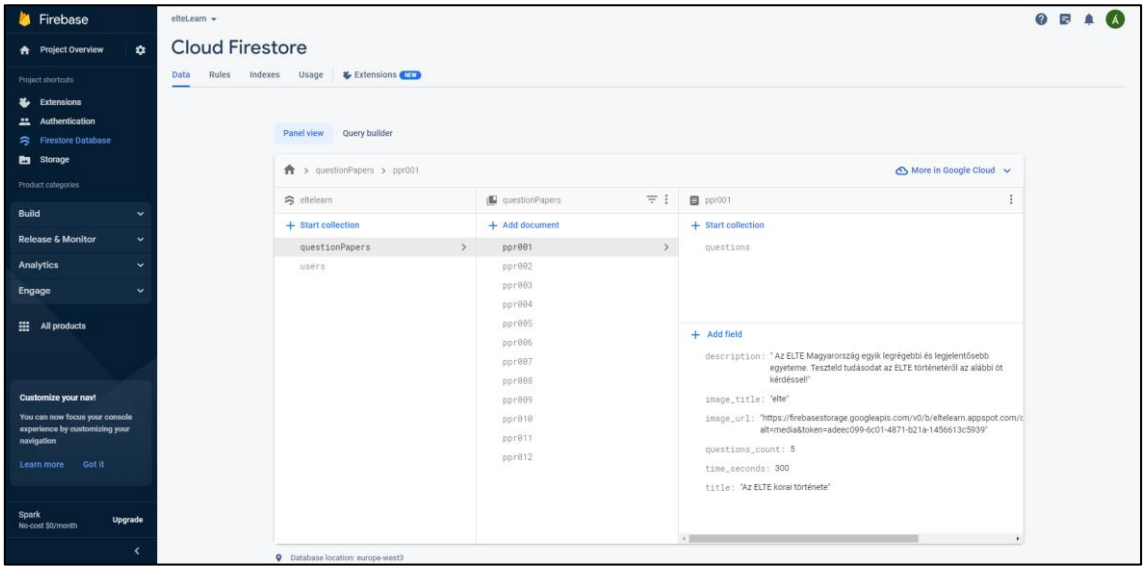


59. ábra: Firestore Database Rules

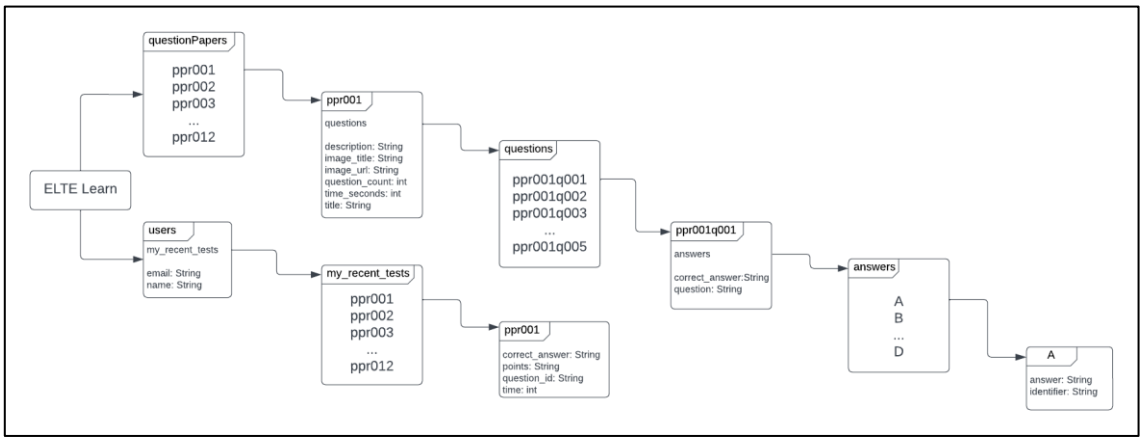
Itt jelenik meg a Cloud Firestore biztonsági szabályok nyelvén (Cloud Firestore security rules language) írt biztonsági szabálykészlet. Ezek a szabályok határozzák meg, hogy ki milyen adatokhoz férhet hozzá az adatbázisban. Bizonyos szintű védelmet nyújtanak azáltal, hogy csak a hitelesített felhasználók számára engedélyezik az adatok olvasását és írását a meghatározott útvonalakon. A szabályokat a megfelelő felhasználóval lehet módosítani, ahogy az 59-es ábrán látható.

3.5.3.2. Data

Firestore Database -> Data:



60. ábra: Firestore Database Data



61. ábra: Adatbázis Diagram

Ezen a két képen látszódik, ami már fentebb a 3.5.2 Firestore Adatbázis (Firestore Database) fejezetnél olvasható volt, hogy a Firestore az adatokat JSON fájlformátumú dokumentumok gyűjteményeiként tárolja, amelyek egymásba ágyazhatók és hierarchikusan strukturálhatók. A hierarchia struktúrájának részletesebb leírását a következő oldaltól kezdve lehet megtekinteni.

A hierarchia felépítése (2 fő részre van bontva):

- **questionPapers**: Kvízek és hozzájuk tartozó adatok, leírások találhatóak meg a struktúra ezen pontjában.
    - **ppr001**:
        - **description**: Egy rövid leírás az adott kvízről.
        - **image_title**: A képnek a címe, ami hozzárendelődik a kvízhez.
        - **image_url**: A képeknek az címei (URL) látszódnak.
        - **questions_count**: Kérdések száma, kvízenként akár eltérhet.
        - **time_seconds**: Kvíz kitöltésére szánt időllimit.
        - **title**: Kvíz címe.
        - **questions**: Kvíz kérdései, lehetséges -, és helyes válaszai
            - **correct_answer:** Helyes válasz
            - **question:** Kérdés
            - **answers:** Lehetséges válaszok.
                - **A:** Egyik lehetséges válasz a megadottak közül.
                    - **answer: „**A” jelű válasznak a tartalma.
                    - **identifier:** A lehetséges válasz azonosítója pl. A, B, C, D, …
                - **B: …**
    - **ppr002**: …
- **users**: Felhasználó adatait tárolja.
    - **email**: Felhasználó email-je
    - **name**: Felhasználó neve
    - **my_recent_test**: Itt tárolódnak a felhasználó által megoldott kvízek.
        - **ppr001**: Egyik megoldott kvíz, melybe belelépve tekinthetőek meg az eredmények.
            - **correct_answer**: Helyes válaszok.
            - **points**: Az adott kvíz megoldása után elért pontszám.
            - **question_id**: A megoldott kvíznek az azonosítója.
            - **time**: Az idő (másodpercben), ami a kvíz megoldásának az ideje.
        - **ppr002: …**

lib\controllers\question_paper**data_uploader.dart**:

```dart
class DataUploader extends GetxController {
  @override
  void onReady() {
    uploadData();
    super.onReady();
  }

  final loadingStatus = LoadingStatus.loading.obs;

  Future<void> uploadData() async {
    loadingStatus.value = LoadingStatus.loading; // value: 0

    final manifestContent = await DefaultAssetBundle.of(Get.context!).loadString("AssetManifest.json");

    final Map<String, dynamic> manifestMap = json.decode(manifestContent);

    final papersInAssets =
        manifestMap.keys.where((path) => path.startsWith("assets/DB/papers") && path.contains(".json")).toList();

    List<QuestionPaperModel> questionPapers = [];

    for (var paper in papersInAssets) {
      String stringPaperContent = await rootBundle.loadString(paper);
      questionPapers.add(QuestionPaperModel.fromJson(json.decode(stringPaperContent)));
    }

    final fireStore = FirebaseFirestore.instance;
    var batch = fireStore.batch();

    for (var paper in questionPapers) {
      batch.set(questionPaperRef.doc(paper.id), {
        "title": paper.title,
        "image_url": paper.imageUrl,
        "image_title": paper.imageTitle,
        "description": paper.description,
        "time_seconds": paper.timeSeconds,
        "questions_count": paper.questions == null ? 0 : paper.questions!.length
      });

      for (var questions in paper.questions!) {
        final questionPath = questionRef(paperId: paper.id, questionId: questions.id);
        batch.set(questionPath, {
          "question": questions.question,
          "correct_answer": questions.correctAnswer,
        });

        for (var answer in questions.answers) {
          batch.set(questionPath.collection("answers").doc(answer.identifier), {
            "identifier": answer.identifier,
            "answer": answer.answer,
          });
        }
      }
    }

    await batch.commit();

    loadingStatus.value = LoadingStatus.completed; // value: 1
  }
}
```

62. ábra: Feltöltés Adatbázisba

A **DataUploader** osztály egy GetxController osztály kiterjesztése, amely egy állapotkezelő könyvtár. E segítségével könnyedén lehet kezelni az alkalmazás dinamikusságát. Az osztály arra szolgál, hogy az adatokat JSON fájlokból feltöltse a Firebase Database-be, ahol tárolódnak és/vagy frissülnek az adatok. Bővebb leírás a következő lapon látszik.

DataUploader osztály bővebben:

1. Mikor a Widget készen áll (onReady) meghívja az uploadData() metódust.
2. Az uploadData() pedig a következőt teszi:
    1. A loadingStatus nevű változó kezdeti értékét a LoadingStatus.loading értékre állítja (ez egy enum érték, amely a loading_status.dart fájlban található).
    2. Betölti az "AssetManifest.json" nevű JSON fájlt az alkalmazás eszközeiből a DefaultAssetBundle.of() segítségével.
    3. A fájl JSON tartalmát egy Map<String, dynamic>-be helyezi, majd dekódolja.
    4. Szűri a Map-ot, hogy megkapja az "assets/DB/papers" könyvtárban található JSON fájlok elérési útvonalainak listáját.
    5. Beolvassa az egyes fájlok tartalmát Stringként, és a fromJson() metódus segítségével QuestionPaperModel objektumként dekódolja. Az így kapott QuestionPaperModels listát a questionPapers-ben tárolja.
    6. Az egymásba ágyazott műveletek a QuestionPaperModellek és a hozzájuk tartozó kérdések és válaszok Firestore Database adatbázisba való feltöltésére szolgál.
    7. Ezeket a műveleteket a batch.commit() utasítással végrehajtja, és felkerül az adatbázisba (Firestore Database).
    8. A loadingStatus értékét LoadingStatus.completed értékre állítja, hogy tudjuk végzett a feltöltéssel.

Összességében a kód helyi JSON fájlokból tölt fel adatokat egy távoli adatbázisba (Firestore Database), és a GetX segítségével kezeli a feltöltési folyamat betöltési állapotát.

3.5.3.3. Modellek

lib\models**question_paper_model.dart**:

A modellek az adatok szervezett és strukturált módon történő ábrázolására szolgálnak. A modelleket jellemzően egy adatforrásból, például adatbázisból vagy API-ból lekért adatok tárolására használják, majd az alkalmazásban az adatok megjelenítésére vagy kezelésére.

```dart
class QuestionPaperModel {
  String id;
  String title;
  String imageTitle;
  String? imageUrl;
  String description;
  int timeSeconds;
  List<Questions>? questions;
  int questionsCount;

  QuestionPaperModel({
    required this.id,
    required this.title,
    required this.imageTitle,
    this.imageUrl,
    required this.description,
    required this.timeSeconds,
    this.questions,
    required this.questionsCount,
  });

  QuestionPaperModel.fromJson(Map<String, dynamic> json)
      : id = json['id'] as String,
        title = json['title'] as String,
        imageTitle = json['image_title'] as String,
        imageUrl = json['image_url'] as String,
        description = json['description'] as String,
        timeSeconds = json['time_seconds'],
        questions = (json['questions'] as List).map((dynamic e) => Questions.fromJson(e as Map<String, dynamic>)).toList
(),
        questionsCount = 0;

  QuestionPaperModel.fromSnapshot(DocumentSnapshot<Map<String, dynamic>> json)
      : id = json.id,
        title = json['title'],
        imageTitle = json['image_title'],
        imageUrl = json['image_url'],
        description = json['description'],
        timeSeconds = json['time_seconds'],
        questions = [],
        questionsCount = json['questions_count'] as int;

  String timeInMinutes() => "${(timeSeconds / 60).ceil()}";
}
```

63. ábra: Kvízek modellje

QuestionPaperModel nevű osztály, egy -egy kérdőív/kvíz modelljét reprezentálja.

Az osztály számos tulajdonsággal rendelkezik, beleértve az ID-t, a címet, a kép címét, a kép URL-címét, a leírást, az időt másodpercben, a kérdések listáját és a kérdések számát.

Ebben az osztályban három konstruktor van. Az első konstruktor (QuestionPaperModel) az osztály új példányának létrehozására szolgál, és a fent látható paramétereket veszi fel.

A második konstruktor (fromJson) az osztály új példányának létrehozására szolgál egy JSON objektumból. Ez a konstruktor egy Map<String, dynamic> objektumot vesz paraméterként, és az osztály tulajdonságait a JSON objektumból inicializálja.

A harmadik (fromSnapshot) konstruktor paraméterként egy DocumentSnapshot<Map<String, dynamic>> objektumot vesz fel, amely a karakterlánc kulcsok dinamikus értékekhez tartozó leképezését tartalmazza. A konstruktor a QuestionPaperModel objektum tulajdonságait a DocumentSnapshotból kapott értékekkel inicializálja.

```
class Questions {
  String id;
  String question;
  List<Answers> answers;
  String? correctAnswer;
  String? selectedAnswer;

  Questions({required this.id, required this.question, required this.answers, this.correctAnswer});

  Questions.fromSnapshot(QueryDocumentSnapshot<Map<String, dynamic>> snapshot)
      : id = snapshot.id,
        question = snapshot['question'],
        answers = [],
        correctAnswer = snapshot['correct_answer'];

  Questions.fromJson(Map<String, dynamic> json)
      : id = json['id'],
        question = json['question'],
        answers = (json['answers'] as List).map((e) => Answers.fromJson(e)).toList(),
        correctAnswer = json['correct_answer'];
}
```

64. ábra: Kérdések modellje

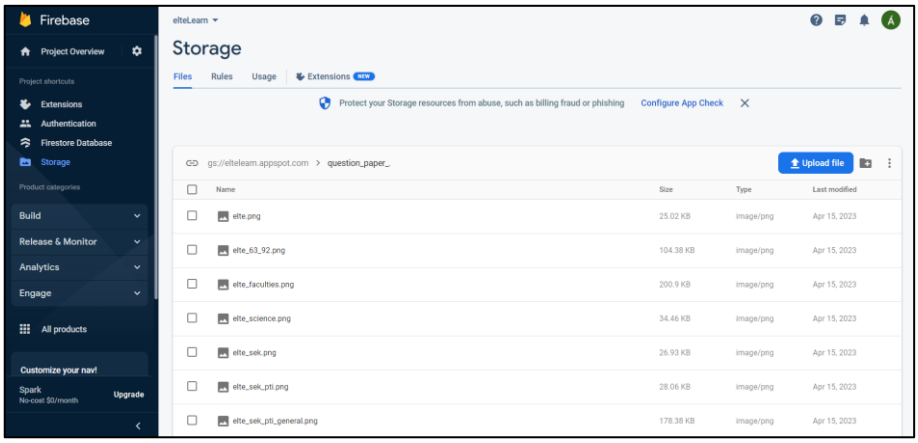
```
class Answers {
  String? identifier;
  String? answer;

  Answers({this.identifier, this.answer});

  Answers.fromJson(Map<String, dynamic> json)
      : identifier = json['identifier'],
        answer = json['answer'];

  Answers.fromSnapshot(QueryDocumentSnapshot<Map<String, dynamic>> snapshot)
      : identifier = snapshot['identifier'] as String?,
        answer = snapshot['answer'] as String?;
}
```

65. ábra: Válaszok modellje

A Questions modell a kvízekben megtalálható kérdéseknek a modellje, az Answers pedig a kvízek válaszainak modellje. Mindkét modellnek ugyanaz a célja és működési elve, mint a **QuestionPaperModel** modellnek, csak más paraméterekkel van felruházva**.**

3.5.4. Tárhely (Storage)

A Firebase Storage a Google Firebase platformja által nyújtott felhőalapú tárolási szolgáltatás. Lehetővé teszi a fejlesztők számára, hogy biztonságosan feltöltsék és letöltsék a felhasználók és/vagy adminisztrátorok által létrehozott tartalmakat, például képeket, hang- és videófájlokat a Firebase-alapú alkalmazásaikból.



66. ábra: Tárhely

Projektet alapul véve, a Firebase Storage-be az adminisztrátor tud feltölteni fájlokat, amik jelen esetben egy mappába (question_paper_images) lettek feltöltve. Ezek a fájlok PNG formátumúak, melyek a kvízekhez kapcsolódó képeket tartalmazzák. Minden kvízhez, ami az alkalmazásban megtalálható, egy-egy képek van rendelve ezek közül.

lib\services**firebase_storage_service.dart**:

```
class FirebaseStorageService extends GetxService {
  Future<String?> getImage(String? imgName) async {
    if (imgName == null) {
      return null;
    }
    try {
      var urlRef = firebaseStorage.child("question_paper_images").
child("${imgName.toLowerCase()}.png");

      var imgUrl = await urlRef.getDownloadURL();

      return imgUrl;
    } catch (e) {
      return throw Exception("FirebaseStorageService: $e");
    }
  }
}
```

67. ábra: Adatok lekérése a tárhelyből

A FirebaseStorageService egy olyan osztály, amely a GetxService osztály kiterjesztése. Tartalmaz egy getImage() metódust, amely paraméterként egy képnevet fogad el, és egy URL karakterláncot ad vissza a kép letöltéséhez a Firebase Storage-ból.

lib\controllers\question_paper**question_paper_controller.dart:**

```dart
final allPapers = <QuestionPaperModel>[].obs;

@override
void onReady() {
  getAllPapers();
  super.onReady();
}

Future<void> getAllPapers() async {
  try {
    QuerySnapshot<Map<String, dynamic>> data = await questionPaperRef.get();
    final paperList = data.docs.map((paper) => QuestionPaperModel.fromSnapshot(paper)).toList();
    allPapers.assignAll(paperList);

    for (var paper in paperList) {
      final imgUrl = await Get.find<FirebaseStorageService>().getImage(paper.imageTitle);
      paper.imageUrl = imgUrl;
    }
    allPapers.assignAll(paperList);
  } catch (e) {
    throw Exception("getAllPapers ERROR: $e");
  }
}
```

68. ábra: Adatok lekérése

Majd a QuestionPaperController osztály getAllPapers() nevű metódusa -amely a Firebase Cloud Firestore questionPaperRef nevű gyűjteményéből kér adatokat-, a fromSnapshot() metódus segítségével QuestionPaperModel objektumok listájává alakítja, és a paperList nevű változóhoz rendeli. Ezután végigjárja a paperList-et, majd a getImage() metódus segítségével lekérdezi a kép URL-címét a Firebase Storage-ból, és hozzárendeli az URL-címet a megfelelő QuestionPaperModel objektumhoz. Végül a frissített paperList az allPapers változóhoz kerül hozzárendelésre, amely a GetX állapotkezelő könyvtár által biztosított RxList osztály egy példánya, ami arra szolgál, hogy dinamikus változtatható legyen a változó tartalma.

lib\screens\quizzes**quizzes_screen.dart:**

```dart
child: ContentArea(
  addPadding: false,
  child: Obx(
    () => ListView.separated(
      padding: UIParameters.mobileScreenPadding,
      itemBuilder: (BuildContext context, int index) {
        final paper = _questionPaperController.allPapers[index];
        if (paperIds.contains(paper.id)) {
          return Padding(
            padding: EdgeInsets.only(bottom: getHeight * 0.025),
            child: QuestionCard(model: paper),
          );
        } else {
          return const SizedBox(height: 0, width: 0);
        }
      },
      separatorBuilder: (_, int index) {
        return const SizedBox(height: 0, width: 0);
      },
      itemCount: _questionPaperController.allPapers.length,
    ),
  ),
),
```

69. ábra: Kvízek megjelenítés

Legvégül a kvízeket a QuizzesScreen osztályban található ListView.separated builder jeleníti meg, képekkel, leírásokkal és elválasztva egymástól. A kvízek kinézetéért pedig a QuestionCard osztály szolgál.

3.6. Harmadik féltől származó könyvtárak (3rd-party libraries)

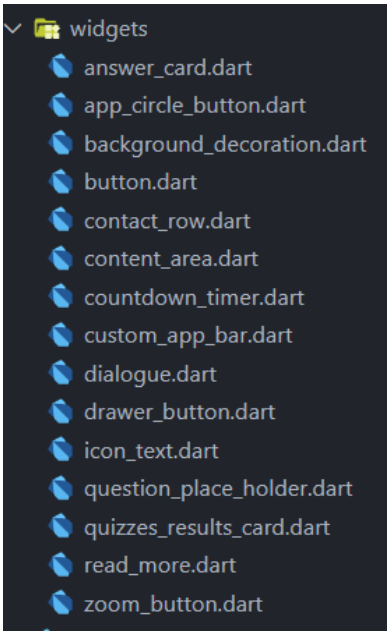
A Flutterben a harmadik féltől származó könyvtár egy olyan csomag vagy csomagkészlet, amelyet nem a Flutter készítői fejlesztettek. Ezek a csomagok kiegészítő funkciókat biztosítanak. A hivatalos oldalon [18] találhatóak, és ezek a könyvtárak könnyen integrálhatók a **pubspec.yaml** fájlba, mely után használható a fejlesztésben.

```
cupertino_icons: ^1.0.2
get: ^4.6.5
cloud_firestore: ^3.4.6
auto_size_text: ^3.0.0
gap: ^2.0.0
google_fonts: ^3.0.1
firebase_storage: ^10.3.7
cached_network_image: ^3.2.1
firebase_core: ^1.24.0
flutter_zoom_drawer: ^3.0.3
firebase_auth: ^3.11.2
url_launcher: ^6.1.6
google_sign_in: ^5.4.2
flutter_svg: ^1.1.6
shimmer: ^2.0.0
easy_separator: ^1.0.1
webview_flutter: ^4.0.7
timeline_tile: ^2.0.0
pull_to_refresh: ^2.0.0
flutter_map: ^2.2.0
flutter_launcher_icons: ^0.11.0
```

70. ábra: 3rd-party libraries

- **flutter_svg** [31]**:** SVG képek megjelenítéséhez.
- **shimmer** [32]**:** Csomag a csillogó betöltési animációk létrehozására.
- **easy_separator** [33]**:** Widgetek közötti elválasztóvonalak létrehozására.
- **webview_flutter** [34]**:** Webes tartalmak megjelenítése.
- **timeline_tile** [35]**:** Idővonal-stílusú UI-elemek létrehozására.
- **pull_to_refresh** [36]**:** Frissítésre szolgáló csomag.
- **flutter_map** [37]**:** Térképek alkalmazásokba való integrálásához.
- **flutter_launcher_icons** [38]**:** Indítóikonok generálásához.
- **cupertino_icons:** Cupertino (iOS-stílusú) ikonok.
- **get** [19]: A függőségek injektáláshoz.
- **cloud_firestore** [20]**:** Firestore-ral való kapcsolathoz szükséges.
- **auto_size_text** [21]**:** Szövegek automatikus átméretezése.
- **gap** [22]**:** Widgetek közötti függőleges és vízszintes rések létrehozására.
- **google_fonts** [23]**:** Google betűtípusok egyszerű használatára szolgál.
- **firebase_storage** [24]**:** A felhőalapú fájltároló szolgáltatással való integrációhoz szükséges.
- **cached_network_image** [25]**:** Internetről származó képek megjelenítése.
- **firebase_core** [26]**:** A Firebase szolgáltatásokkal való integrációhoz szükséges, beleértve a hitelesítést, a Firestore-t és a Storage-t.
- **flutter_zoom_drawe** [27]**r:** Zoomolható menüsáv.
- **firebase_auth** [28]**:** A felhasználói hitelesítés kezelésére szolgáló szolgáltatás integrálásához kell.
- **url_launcher** [29]**:** Segít az URL-ek elindításához az eszköz alapértelmezett böngészőjében.
- **google_sign_in** [30]**:** Felhasználók Google-fiókkal való hitelesítésére szolgál.

3.7. Egyedi Widget-ek

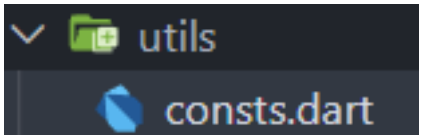


71. ábra: Egyedi Widget-ek

lib/**widgets**:

A már többször említett, úgynevezett Widget-ek, amelyek felhasználói felület alapvető építőkövei, amelyek a felhasználói felület hierarchiájának egy részét képviselik. Ezek egymásba ágyazhatóak, testreszabhatóak az egyedi igényekhez igazodva. A beépített widgetek mellett saját, egyéni Widget-eket is létre lehet hozni. A „widgets" mappában találhatóak az általam készített egyedi Widget-ek.

3.8. Utils



72. ábra: Utils mappa

```dart
// Url Sources
const String urlStartsWith = "https://";
const String facebookUrl = "https://m.facebook.com/elte.sek";
const String websiteUrl = "https://sek.elte.hu/";
const String myWebsite = "https://oradam.web.elte.hu/portfolio-website-1/"
;

// ———————————————— Menu Sources START ————————————————
_
const String logInTitle1 = "Jelentkezz be";
const String logInTitle2 = "Bejelentkezés";
const String logOutTitle = "Kijelentkezés";

const String elte = "ELTE";
const String elteSek = "ELTE SEK";
const String elteSekPti = "ELTE SEK PTI";

// Quizzes
const String quizzesTitle = "Kvízek";
const String newbieTitle = "Hello újonc, kérlek jelentkezz be";
const String elteQuizzesTitle = "ELTE Kvízek";
const String elteSekQuizzesTitle = "ELTE SEK Kvízek";
const String elteSekPtiQuizzesTitle = "ELTE SEK PTI Kvízek";
```

73. ábra: Consts fájl

lib/**utils**:

Az **utils** mappában látható egy **consts.dart** fájl, mely a képernyőkön látható szövegeket, linkeket tartalmazza egy helyen azért, hogy ha a módosítani szeretnénk, akkor ne kelljen megkeresni mindig azt a fájl, ahol meg vannak jelenítve a szövegek, linkek, hanem egyszerűen csak 1 fájl elég módosítani.

4. Összefoglalás és további fejlesztési lehetőségek

2022 nyarán megismertem a Dart programozási nyelvet és a Flutter keretrendszert, és rögtön észrevettem a bennük rejlő potenciált. Majd egyre jobban érdekelni kezdtek azok a lehetőségek, amelyeket ezek a technológiák nyújtanak a mobilalkalmazás-fejlesztés terén.

Mivel ELTELearn mobilalkalmazás célja az ELTE Savaria Egyetemi Központ (SEK) hallgatóinak és jelentkezőinek az egyetemi életének, híreinek és fontos információinak könnyű elérése, ezenfelül kvízek formájában egy játékos felületet biztosít a rendelkezésre álló információk alapján. Ezért úgy találtam, hogy ez a projekt elég erős kihívást jelent számomra. A fejlesztés során értékes tapasztalatokat szereztem a programozás és a szoftverfejlesztés terén, valamint mélyebb megértést szereztem a mobilalkalmazások tervezéséről és a felhasználói élményről.

A projekt során a Dart és a Flutter rendszert hihetetlenül felhasználóbarátnak és egyszerűnek találtam, ami lehetővé tette számomra, hogy könnyedén implementáljak funkciókat az alkalmazáson belül. A keretrendszer egyszerűsége abban is segített, hogy fejlesszem a kódolási készségeimet és jobban megértsem az alkalmazásfejlesztési folyamatot.

A jövőre nézve úgy vélem, hogy a projekt során szerzett tapasztalatok felbecsülhetetlen értékűek lesznek, amikor tovább kutatom a mobilalkalmazás-fejlesztés területét. Erősen fontolgatom, hogy ezen a területen folytatom a karrieremet, és izgatott vagyok a lehetőségek miatt, amelyeket ez a szakma kínál.

Az alkalmazás tovább fejlesztésére ilyen lehetőségeket látok, mint például a(z):

- **Admin felület**: A kvízek létrehozása, képek feltöltése és történetek szerkesztése mellett megfontolhatná a felhasználói fiókok kezelésének és a használati elemzések megtekintésének lehetőségét is. Ez lehetővé tenné az alkalmazás adminisztrátorának, hogy értékes betekintést nyerjen az alkalmazás használatába, és azonosítani tudja a fejlesztendő területeket.
- **Több regisztrációs/bejelentkezési lehetőség**: A felhasználóknak lehetőséget kínálni a közösségi média fiókokkal (például Twitter és/vagy Facebook) vagy e-mailben és jelszóval történő bejelentkezésre.
- **Felhasználói beállítások menü**: A felhasználók testre szabhassák az alkalmazás kinézetét személyes igényük szerint. Ez olyan dolgokat foglalhat magában, mint a preferált nyelv kiválasztása vagy az értesítések beállítása.
- **Többnyelv támogatása**: Annak érdekében, hogy az alkalmazás szélesebb közönség számára is elérhető legyen, a többnyelvű támogatás hozzáadását érdemes lenne létrehozni.
- **Közzététel az összes lehetséges forgalmazónál**: A lehető legszélesebb közönség elérése az alkalmazás közzétételét több alkalmazásboltban, például a Google

Play és az Apple App Store áruházakban. Ez megkönnyíti a felhasználók számára az alkalmazás felfedezését és letöltését, függetlenül attól, hogy melyik platformot használják.

- **Push-értesítések**: Értesítések hozzáadása, hogy a felhasználókat folyamatosan tájékoztassa az új kvízekről, történetekről és friss hírekről.
- **Gamification**: Például jelvények és jutalmak szerzése kvízek kitöltéséért vagy a történetek elolvasásáért. Ez még vonzóbbá teszi az alkalmazást, és arra ösztönzi a felhasználókat, hogy visszatérjenek
- **Közösségi megosztás**: Adjon lehetőséget a felhasználók számára, hogy könnyedén megoszthassák a történeteket és kvízeket a közösségi médiaplatformokon, például a Facebookon, a Twitteren és az Instagramon. Ez segít növelni az alkalmazás ismertségét és több letöltést generálni.

Összességében büszke vagyok az ELTELearn mobilalkalmazásának fejlesztése során elért eredményeimre, elért tudásért, hogy tanulhattam és fejlődhettem a projekt révén. Hiszem, hogy ez a tapasztalat értékes készségekkel és tudással ruházott fel, amelyek jó szolgálatot tesznek majd a szoftverfejlesztés és a mobilalkalmazások tervezése iránti szenvedélyem további folytatásában.

5. Ábrajegyzék

1. ábra: Alkalmazás Ikon 2
2. ábra: Splash Screen 2
3. ábra: Bemutatkozó oldal 2
4. ábra: Kezdőfelület 3
5. ábra: Menüsáv 3
6. ábra: Bejelentkezés 3
7. ábra: Google bejelentkezés 3
8. ábra: Bejelentkezett felhasználó 3
9. ábra: Kvízek menü 4
10. ábra: ELTE kvízek 4
11. ábra: ELTE SEK kvízek 4
12. ábra: ELTE SEK PTI kvízek 4
13. ábra: Első kérdés 5
14. ábra: Utolsó kérdés 5
15. ábra: Kérdések megtekintése 5
16. ábra: Helyes válaszok 5
17. ábra: Válaszok megtekintése 5
18. ábra: Történetek menü 6
19. ábra: Történet 6
20. ábra: Történet kibontása 6
21. ábra: Hírek menü 7
22. ábra: ELTE SEK Weboldal 7
23. ábra: ELTE SEK Facebook 7
24. ábra: Kapcsolat menü 8
25. ábra: Kapcsolatok 8
26. ábra: Helyzet 8
27. ábra: Email 8
28. ábra: Weboldal 8
29. ábra: Sötét mód - Menü 9

30. ábra: Sötét mód - Történetek 9
31. ábra: Sötét mód - Kvízek 9
32. ábra: Sötét mód - Kérdések 9
33. ábra: Sötét mód - Megválaszoltak.....9
34. ábra: Nagyított kép.....10
35. ábra: Lejárt az idő 10
36. ábra: Kvíz megkezdése sikertelen 10
37. ábra: StatelessWidget.....14
38. ábra: StatefulWidget 14
39. ábra: Projekt létrehozása.....15
40. ábra: Projekt beállításai 15
41. ábra: Projekt felépítése.....16
42. ábra: Források 17
43. ábra: Történetek forrásai.....17
44. ábra: Kvízek forrásai 17
45. ábra: Alkalmazás kódja 18
46. ábra: Kezdőlépések.....19
47. ábra: Project elnevezése.....20
48. ábra: Google Analytics 20
49. ábra: Google Analytics Konfiguráció.....21
50. ábra: App - Firebase kapcsolat 21
51. ábra: Kapcsolat beállítása(i).....22
52. ábra: Autentikáció 23
53. ábra: Autentikáció típusai.....24
54. ábra: Felhasználók 24
55. ábra: Google bejelentkezés gomb.....25
56. ábra: Google bejelentkezés funkcionalitása 25
57. ábra: Kijelentkezés funkcionalitása 26
58. ábra: Kijelentkezést segítő metódus 26
59. ábra: Firestore Database Rules 27
60. ábra: Firestore Database Data 28

61. ábra: Adatbázis Diagram 28
62. ábra: Feltöltés Adatbázisba 30
63. ábra: Kvízek modellje 32
64. ábra: Kérdések modellje 33
65. ábra: Válaszok modellje 33
66. ábra: Tárhely 34
67. ábra: Adatok lekérése a tárhelyből 34
68. ábra: Adatok lekérése 35
69. ábra: Kvízek megjelenítés 35
70. ábra: 3rd-party libraries 36
71. ábra: Egyedi Widget-ek 37
72. ábra: Utils mappa 37
73. ábra: Consts fájl 37

6. Irodalomjegyzék

[1] R. Payne: Beginning App Development with Flutter, 2019.12.05., [336], ISBN-13 978-1484251805.

[2] G. Blokdyk: Firebase The Ultimate Step-By-Step Guide, 2022.02.04., [302], ISBN-13 978 0655321477.

[3] https://www.elte.hu/content/az-egyetem-tortenete.t.4?m=18, Utolsó megtekintés: 2023.03.12.

[4] https://sek.elte.hu/about, Utolsó megtekintés: 2023.03.10.

[5] https://sek.elte.hu/jubileum60/tortenet, Utolsó megtekintés: 2023.03.10.

[6] https://sek.elte.hu/kepzesek, Utolsó megtekintés: 2023.03.11.

[7] https://szoftvermernok.inf.elte.hu, Utolsó megtekintés: 2023.03.11.

[8] https://www.openstreetmap.org/, Utolsó megtekintés: 2023.04.07.

[9] https://flutter.dev/, Utolsó megtekintés: 2022.12.28.

[10] https://code.visualstudio.com/, Utolsó megtekintés: 2022.10.27.

[11] https://developer.android.com/studio, Utolsó megtekintés: 2022.07.19.

[12] https://developer.apple.com/xcode/, Utolsó megtekintés: 2022.07.19.

[13] https://firebase.google.com/, Utolsó megtekintés: 2023.04.26.

[14] https://docs.flutter.dev/data-and-backend/firebase, Utolsó megtekintés: 2023.03.03.

[15] https://dart.dev/guides, Utolsó megtekintés: 2022.10.20.

[16] https://console.firebase.google.com/u/0/, Utolsó megtekintés: 2023.04.26.

[17] https://firebase.google.com/docs/database, Utolsó megtekintés: 2023.03.17.

[18] https://pub.dev/, Utolsó megtekintés: 2023.04.02.

[19] https://pub.dev/packages/get, Utolsó megtekintés: 2023.02.09.

[20] https://pub.dev/packages/cloud_firestore, Utolsó megtekintés: 2023.01.18.

[21] https://pub.dev/packages/auto_size_text, Utolsó megtekintés: 2022.11.25.

[22] https://pub.dev/packages/gap, Utolsó megtekintés: 2022.10.27.

[23] https://pub.dev/packages/google_fonts, Utolsó megtekintés: 2022.12.16.

[24] https://pub.dev/packages/firebase_storage, Utolsó megtekintés: 2023.01.23.

[25] https://pub.dev/packages/cached_network_image, Utolsó megtekintés: 2023.01.25.

[26] https://pub.dev/packages/firebase_core, Utolsó megtekintés: 2023.01.24.

[27] https://pub.dev/packages/flutter_zoom_drawer, Utolsó megtekintés: 2023.02.02.

[28] https://pub.dev/packages/firebase_auth, Utolsó megtekintés: 2022.11.14.

[29] https://pub.dev/packages/url_launcher, Utolsó megtekintés: 2023.02.27.

[30] https://pub.dev/packages/google_sign_in, Utolsó megtekintés: 2022.11.14.

[31] https://pub.dev/packages/flutter_svg, Utolsó megtekintés: 2023.04.29.

[32] https://pub.dev/packages/shimmer, Utolsó megtekintés: 2022.12.10.

[33] https://pub.dev/packages/easy_separator, Utolsó megtekintés: 2022.10.28.

[34] https://pub.dev/packages/webview_flutter, Utolsó megtekintés: 2023.04.27.

[35] https://pub.dev/packages/timeline_tile, Utolsó megtekintés: 2023.02.21.

[36] https://pub.dev/packages/pull_to_refresh, Utolsó megtekintés: 2023.04.26.

[37] https://pub.dev/packages/flutter_map, Utolsó megtekintés: 2023.03.07.

[38] https://pub.dev/packages/flutter_launcher_icons, Utolsó megtekintés: 2023.02.04.

Szakdolgozat OneDrive linkje: https://ikelte-my.sharepoint.com/:u:/g/personal/jam5bo_inf_elte_hu/EbetZTJNcLlIpixHLQd904QB3SXK7_2MkSMZbtvTK3jAmQ?e=geb0sS