

**Szegedi Tudományegyetem**  
**Informatikai Intézet**

# **SZAKDOLGOZAT**

**Kun Réka Bianka**

**2023**

**Szegedi Tudományegyetem  
Informatikai Intézet**

**Rendezvény szervező multiplatform  
mobilalkalmazás**

**Szakdolgozat**

Készítette:  
**Kun Réka Bianka**  
programtervező  
informatikus szakos  
hallgató

Témavezető:  
**Kiss-Vetráb Mercedes**  
PhD Hallgató

**Szeged  
2023**

## **Feladatkírás**

A szakdolgozat témája a multiplatform mobilalkalmazás fejlesztés. A dolgozat célja, hogy a hallgató megtanulhassa az alapvető fejlesztési folyamatokat egészen a tervezéstől a kész szoftverig. Jelen alkalmazás célja, hogy a felhasználók számára lehetővé tegye saját események létrehozását, megosztását másokkal és a környezetükben megtalálható események megtekintését. Az alkalmazás funkciói között szerepel a regisztráció, az események létrehozása, az eseményekre való feliratkozás, valamint a kedvenc események elmentése. Az alkalmazás célja, hogy megkönnyítse az emberek számára az események megtalálását és az azokon való részvételt. A dolgozat megírása során az alábbi munkafolyamatok elsajátítására van lehetőség: UX/UI tervezés a letisztult és felhasználóbarát végeredményhez, Trello használata, Flutter-Firebase adatbázis kezelése, Flutter alapok elsajátítása, MVVM fejlesztési minta megértése, Git használata.

## Tartalmi összefoglaló

- **A téma megnevezése:**

Rendezvény szervező mobilalkalmazás.

- **A megadott feladat megfogalmazása:**

Egy olyan mobilalkalmazás megvalósítása, amelynek használata egyszerű és könnyen átlátható a felhasználók számára. Az alkalmazás lehetőséget biztosít a felhasználóknak, hogy könnyedén hozzanak létre és testre szabjanak saját eseményeket, és megosszák azokat a többi felhasználóval. Az alkalmazás lehetővé teszi a felhasználók számára, hogy jelezzék részvételi szándékukat az eseményeken, amelyeken részt kívánnak venni. Továbbá, az alkalmazásban lehetőség van azon események böngészésére is, amelyek a közelben kerülnek megrendezésre. A felhasználói élmény fokozása és az alkalmazás egyszerű kezelhetősége kiemelt fontosságú volt az alkalmazás tervezése során.

- **A megoldási mód:**

Az alkalmazás fejlesztése során fontos volt számomra, hogy az alkalmazás felhasználói számára elérhetővé tegyem az aktuális helyzet meghatározását, így az alkalmazásban be kellett építenem egy olyan funkciót, amely lehetővé teszi a felhasználó számára, hogy meghatározza a pontos helyzetét. Ennek érdekében implementáltam egy helymeghatározó modult, amelynek segítségével a felhasználó pontos helyzete meghatározható és az alkalmazás használata ezzel még kényelmesebbé válik. Ezen funkció beépítése további lehetőségeket nyújt az alkalmazás felhasználóinak, például megkönnyíti az események keresését, amelyek a felhasználó közelében kerülnek megrendezésre.

- **Alkalmazott eszközök, módszerek:**

Az alkalmazás elkészítéséhez a Flutter multiplatform mobilalkalmazás fejlesztési keretrendszert használtam, amely a Dart programozási nyelven alapul. Az adatok kezelésére a Google Firebase adatbázis-szolgáltatását alkalmaztam a Firestore-t.

- **Elért eredmények:**

Egy olyan mobilalkalmazás, amely felhasználóbarát és intuitív. Könnyen integrálódik az eszközökkel.

- **Kulcsszavak:**

Firebase, Firestore, Dart, Flutter, design, mobilalkalmazás.

## Tartalomjegyzék

Feladatkiírás .....	3
Tartalmi összefoglaló .....	4
Tartalomjegyzék .....	5
Motiváció .....	6
1.Mobilalkalmazás fejlesztés .....	6
1.1. Natív és multiplatform fejlesztői eszközök története .....	7
1.2. Flutter és Dart .....	8
2. UX és UA design .....	11
2.1. Történeti áttekintő .....	11
2.2. Főbb alkalmazott tervezési szabályok .....	12
3. Moodboard és logo .....	13
3.1. Moodboard .....	13
3.2. Logo .....	14
4. Képernyőtervek .....	16
5. Fejlesztési folyamat .....	18
5.1. Trello .....	18
5.2. Github .....	19
6. Alkalmazás szerkezeti felépítése .....	20
6.1. MVVM .....	20
6.2. Adatbázis .....	22
6.3. Üzleti logika .....	23
6.4. Az alkalmazásban használt csomagok.....	25
6.5. Kész alkalmazás .....	27
Irodalomjegyzék .....	37
Nyilatkozat .....	39
Köszönetnyilvánítás.....	40

## Motiváció

Az események szervezésére szolgáló alkalmazások széles körben elterjedtek, mivel hatékonyabbá teszik a rendezvények, találkozók és események tervezését és lebonyolítását. Ezek az alkalmazások lehetővé teszik a felhasználók számára, hogy programokat tervezzenek, információkat osszanak meg és nyomon követhessék az aktuális rendezvényeket. Az események szervezésére szolgáló alkalmazások időt és energiát takarítanak meg a felhasználók számára, lehetővé téve a rendezvények zökkenőmentes lebonyolítását.

Egy olyan mobilalkalmazás elkészítése volt a célom, amelyben ezek a tulajdonságok megtalálhatóak, valamint modern és a felhasználók számára könnyen kezelhető. Az alkalmazás megtervezésénél fontos szempont volt, hogy a felhasználói felülete vonzó, átlátható és könnyen kezelhető legyen.

### 1. Mobilalkalmazás fejlesztés

Az okostelefonokra vagy táblagépekre szánt mobilalkalmazások fejlesztése során a cél az, hogy egy hatékony és felhasználóbarát alkalmazást hozzanak létre. Az Android és iOS rendszerekre az alkalmazások platform-specifikusan készülnek, azaz külön-külön. Az informatika fejlődésének köszönhetően jelenleg már rendelkezésre állnak olyan keretrendszerek, mint például a React Native [21], Xamarin Forms [22], Ionic [23] és a Flutter [1], amelyek lehetővé teszik az alkalmazások egyszeri fejlesztését, majd több platformon történő futtatását.

A mobilalkalmazásunk sikeréhez nélkülözhetetlen a piackutatás és azon alkalmazások tanulmányozása, amelyek azonos piaci rést céloznak meg. Fontos, hogy az alkalmazásunk olyan egyedi funkciókkal rendelkezzen, amelyek megkülönböztetik azt a hasonló alkalmazásoktól.

A mobilalkalmazás fejlesztés több fontos lépésből tevődik össze. Ezek közé tartozik a tervezés, prototípus készítése, felhasználói felület tervezése és implementálása, felhasználói visszajelzések figyelembevétele, az alkalmazás optimalizálása és tesztelése. Az eredményes mobilalkalmazás fejlesztéséhez ezek a lépések nélkülözhetetlenek.

Az alkalmazásfejlesztés egy összetett folyamat, amely az felhasználói igényeket és elvárásokat egyaránt figyelembe veszi annak érdekében, hogy hatékony, használható és felhasználóbarát alkalmazások jöjjenek létre.[15]

## 1.1. Natív és multiplatform fejlesztői eszközök története

A fejlesztés során alkalmazott technológiák és környezetek alapján három nagy kategóriába sorolhatók a mobilalkalmazások, ezek a webalkalmazások, natív alkalmazások és a multiplatform alkalmazások.

A natív fejlesztés során minden platformra külön-külön kell írni az alkalmazást, mivel a platformok különböző nyelveket és eszközöket használnak. Az Android [24] alkalmazásokat Kotlin [25] vagy Java [26] nyelven, az iOS [27] alkalmazásokat pedig Objective C [28] vagy Swift [29] nyelven lehet írni. A Swift az iOS által kifejlesztett programozási nyelv. A natív fejlesztés lehetővé teszi a fejlesztők számára, hogy teljes mértékbe kihasználhassák a platform előnyeit, így az alkalmazások stabilak, azonban az alkalmazások karbantartása, valamint fejlesztése időigényes.

A natív alkalmazások előnye, hogy egyetlen platformra vannak optimalizálva, ezáltal gyorsan és könnyedén futnak.

A multiplatform fejlesztési környezetek megjelenésével lehetővé vált az alkalmazások egyszerűbb és költséghatékonyabb fejlesztése több platformra egyszerre. A multiplatform megközelítés lehetővé teszi, hogy az alkalmazást egyszer írják meg, majd a platform-specifikus fordítóprogramok segítségével átalakítsák az alkalmazást az összes platformra. Az ilyen keretrendszerek közé tartoznak a React Native, Xamarin Forms, Flutter és sok más.

A multiplatform fejlesztés előnyei közé tartozik, hogy az alkalmazások egyszerre írhatók meg több platformra, így időt és pénzt takarítanak meg a fejlesztők számára. A hátránya pedig, hogy a platformok specifikus funkcióinak kivételezése nehezebb, így az alkalmazások teljesítménye és felhasználói élménye nem mindig olyan jó, mint a natív alkalmazásoké. [12, 13]

Az alkalmazásom fejlesztése során a multiplatform fejlesztési folyamatot választottam, mivel a technológia rohamos fejlődését követően egy alkalmazásnak képesnek kell lennie arra, hogy különböző platformokon, például iOS-en, Androidon, vagy akár weben is fusson, anélkül, hogy minden egyes platformra külön-külön keljen lefejleszteni az alkalmazást, ezáltal hatékonyabb fejlesztési és karbantartási folyamatot eredményezve, valamint időt és pénzt takarít meg.

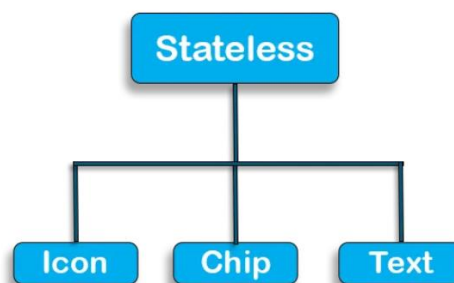
## 1.2. Flutter és Dart

Az alkalmazásomat a Flutter keretrendszerben valósítottam meg. A Flutter egy nyílt forráskódú mobilalkalmazás-keretrendszer, amely a Google által fejlesztett és fenntartott. Segítségével gyorsan és hatékonyan lehet platformfüggetlen mobilalkalmazásokat készíteni.

A Flutter alkalmazások fejlesztése egyre népszerűbbé válik, mivel az egyik legmodernebb és leginkább támogatott keretrendszer a mobil alkalmazások fejlesztéséhez. A Flutter egyetlen kódbázisból képes létrehozni alkalmazásokat, amelyek egyaránt működnek iOS-en, Androidon és a weben is. Lehetővé teszi az alkalmazások különböző eszközökön való egységes megjelenítését. A Flutterben készített alkalmazások általában gyorsabbak és hatékonyabbak, mint a hozzá hasonló natív alkalmazások. Széles körű könyvtárral rendelkezik, amely segítségével az alkalmazáshoz különböző funkciók és szolgáltatások adhatók.

A Flutter keretrendszerben az alkalmazás felhasználói felülete widgetekből épül fel. A widgetek használatával az alkalmazás fejlesztése egyszerűbbé és hatékonyabbá válik. Könnyen testreszabhatóak, kombinálhatóak, mivel a Flutter keretrendszer sok kész widgetet kínál a grafikus felhasználói felület készítéséhez. Egy Flutter alkalmazás képernyő nézete attól függ, hogy az alkalmazás felépítéséhez milyen widgeteket használtak fel. A widgetek állapota és működése a StatelessWidget és a StatefulWidget widgetek alkalmazásával kezelhető.

A StatelessWidget, egy Flutterban használt widget típus, amely nem változtatja állapotát az alkalmazás futása közben. StatelessWidgetek lehetnek szövegek, ikonok, és más olyan UI elemek, amelyek állapota független más widgetektől. [5]

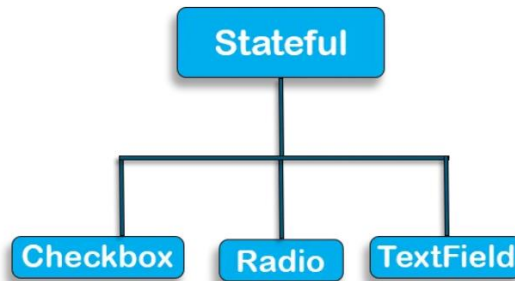


1.1 ábra – StatelessWidget

A StatefulWidget, egy olyan widget, amely változtatható állapottal rendelkezik. Mivel az állapot változása miatt újraépíti magát, így több erőforrást igényelnek, mint a



StatelessWidgetek. Általában az alkalmazás interaktív vagy animált részeit írják le, amelyek állapota folyamatosan változik. A StatefulWidgetek olyan widgetek lehetnek, mint például a checkboxok, beviteli mezők és még sok más. [10]



*1.2 ábra – StatefulWidget*

Funkcionalitásuk alapján két kategóriába sorolhatóak a Flutter widgetek, ezek az elrendezési widgetek (layout widgets) és a vezérlő widgetek (control widgets). Az elrendezési widgetek a Flutterben az elrendezés kialakításáért felelősek, meghatározzák, hogy a többi widget hol helyezkedjen el a felhasználói felületen. Ezek közé az alábbiak tartoznak: [11]

- Row: egy olyan widget, amely lehetővé teszi a gyerekek vízszintes elrendezését. A gyerekek tetszőleges Flutter widgetek lehetnek, mint például szövegek, képek.
- Column: a gyerekek függőleges elrendezését teszi lehetővé.
- Stack: a Stack widget, egy olyan elrendezési widget, amely lehetővé teszi, hogy több widgetet egymásra helyezhessünk, és szabadon elhelyezhessük őket a koordinátarendszerben. Az alkalmazásakor a widgetek egymásra kerülnek, így előtérbe hozhatjuk, átlátszóvá tehetjük vagy elrejtethetjük őket. Gyakran alkalmazzák animálásakor, valamint komplexebb felhasználói felületek kialakításakor.
- Container: lehetővé teszi a gyerekek méretének és pozíciójának beállítását, valamint a szegély, háttérszín és árnyékolás testreszabását is. A Container widgetet gyakran használják arra, hogy elrendezési konténereket hozzanak létre más widgeteknek.
- SizedBox: egy alapvető widget, amely egy üres téglalapot jelenít meg a felhasználói felületen. A widget segítségével megadható a widget magassága és szélessége, valamint a widget és annak gyerekei közötti térköz.

Az elrendezési widgetekkel ellentétben a vezérlő widgetek a felhasználói interakciókat kezelik, mint például az adatok bevitelét, az érintésre történő kattintást. Flutterben a vezérlő widgetek közé a következők tartoznak: [30]

- Button: gombok megjelenítését teszi lehetővé, amelyet a felhasználó érintésre kattinthat.
- Radion button: lehetővé teszi a felhasználó számára, hogy például kiválasszon egy elemet egy listából.
- Slider: segítségével egy csúszka jelenik meg a felhasználó számára, amellyel egy adott tartományon belül állíthat be értékeket.
- Switch: lehetővé teszi a felhasználó számára, hogy be- vagy kikapcsoljon egy adott beállítást.
- Checkbox: engedélyezi a felhasználónak, hogy kiválassza vagy törölje a jelölést az adott opció mellett.

A Dart [2] egy nyílt forráskódú, a Flutter alkalmazások kódolásához használt modern, objektumorientált programozási nyelv. Ideális modern, komplex, hatékony alkalmazások fejlesztéséhez. A Dart nyelv egy modern és rugalmas programozási nyelv, amely könnyen tanulható. A nyelv szintaxisa hasonló más C-szerű nyelvekhez, de magasabb szintű funkciókkal rendelkezik, amelyek segítségével az alkalmazások fejlesztése gyorsabb és hatékonyabb lehet. A nyelv nagyon hasznos a web- és mobilalkalmazások fejlesztése során. A Dart és a Flutter jellemzői:

- Dart egy objektumorientált programozási nyelv. A Flutter miatt vált népszerűvé. Nem csak a Flutter keretrendszerrel használható.
- Flutter egy UI keretrendszer, amely főként a felhasználói felületekre fókuszál, de támogatja az üzleti logikát is. Különböző platformra készíthető vele alkalmazás.

## **2. UX és UI design**

A mobilalkalmazások fejlesztésénél fontos az alkalmazás felhasználói felületének megtervezése.

A UX design a tervezési folyamatok összessége, amelynek célja a felhasználói élmény növelése az alkalmazás használata során. Az alkalmazás sikerének egyik legfontosabb eleme a megfelelő UX design. Az UX design segít a felhasználóknak abban, hogy az alkalmazás könnyen kezelhető, érthető legyen.

A UI design, az alkalmazások felhasználói felületének tervezését jelenti. Eszközei közé tartoznak a színek, betűtípusok, ikonok, gombok és más vizuális elemek, amelyek segítenek a felhasználóknak megérteni az alkalmazás funkcióit. Célja az egységes és összhangban lévő felhasználói felület kialakítása az alkalmazás minden részére. [9, 14]

Az alkalmazásom tervezése során nagy hangsúlyt fektettem a UX és UI designok tervezésére, mivel a felhasználói élmény minősége fontos az alkalmazás sikeréhez. Ennek érdekében olyan megközelítést alkalmaztam, amely kiemelt figyelmet fordított az alkalmazás használhatóságára, a navigációra, a letisztult és egységes designra, valamint az esztétikus megjelenésre. Céлом az volt, hogy az alkalmazás egyszerűen és kellemesen használható legyen, ezzel növelve a felhasználók érdeklődését.

### **2.1. Történeti áttekintő**

A UX és UI tervezésének története régmúlta nyúlik vissza. Már az ókori Görögországban is felfedezhetők voltak olyan tervezési elemek, amelyek később a modern UX és UI tervezésben is felbukkannak. Azonban az igazi áttörés az ipari forradalommal és a technológiai fejlődéssel kezdődött, amikor az emberek elkezdtek felfedezni a tervezési elemek hatását a felhasználói élményre. Az első olyan eszközök, amelyek segítették a tervezőket a felhasználói élmény javításában, a 20. század elején jelentek meg. Az utóbbi évtizedekben az informatika fejlődésével a UX és UI tervezés kulcsfontosságúvá vált az alkalmazások, weboldalak és más digitális felületek tervezése során. Ma már számos eszköz és módszer áll rendelkezésre a UX és UI tervezés hatékony és eredményes megvalósításához. [6, 8]

## 2.2. Főbb, alkalmazott tervezési szabályok

A UX és UI tervezés területén számos szabály és elv létezik, amelyek hozzájárulnak az alkalmazás hatékony és felhasználóbarát felületének kialakításához. A következők tartoznak a UX design főbb tervezési szabályai közé:

- **Felhasználóközpontú tervezés:** A tervezés felhasználóközpontú legyen, azaz a felhasználók igényeire és szokásaira koncentráljunk.
- **Konzisztencia:** Az alkalmazás minden eleme hasonló stílust, elrendezést és funkciókat kell használjon.
- **Felhasználói tesztek:** A felhasználói tesztek és visszajelzések figyelembevétele elengedhetetlen az alkalmazás tervezése során, hogy a felhasználók igényeire és visszajelzéseire reagáljunk és a lehető legjobb felhasználói élményt nyújtsuk.
- **Használhatóság:** Az alkalmazás használhatósága legyen a fókuszban, az interakciók logikusak legyenek.

A UI design szabályai közé pedig az alábbiak tartoznak:

- **Betűtípus:** Fontos a megfelelő betűtípus kiválasztása, hogy az információk és funkciók szövege könnyen olvasható és érthető legyen.
- **Egyszerűség:** Az egyszerűség azért fontos, hogy az alkalmazás ne legyen túlszűfolt és a felhasználók könnyen megtalálják az információkat és funkciókat.
- **Vizuális hierarchia:** Az információk vizuális hierarchiájának kialakítása segít a felhasználóknak az információk megértésében és könnyebb észrevételében.
- **Színhasználat:** A megfelelő színhasználat segítségével a felhasználók könnyebben észreveszik a fontos információkat és a funkciókat.

Az alkalmazásom tervezése és fejlesztése során kiemelten fontosnak tartottam az UX és UI tervezési szabályok betartását, hogy a felhasználók számára egy intuitív és pozitív felhasználói élményt biztosítsak. Ehhez számos UX és UI tervezési alapelvet alkalmaztam, mint például a felhasználóközpontú tervezést, az egyszerűséget, a felhasználóbarát navigációt és az egységes stílushasználatot. Céлом az volt, hogy az alkalmazás használata egyszerű és hatékony legyen, és a felhasználók pozitív élményt szerezzenek vele kapcsolatban.

### 3. Moodboard és logo

#### 3.1. Moodboard

A moodboard egy vizuális összeállítás, amelynek célja, hogy különböző vizuális elemeket, mint például színeket, mintákat, textúrákat mutasson be. A moodboard azért hasznos, mert segít összeállítani az elképzeléseket egy konkrét látványvilág megteremtése érdekében, és lehetővé teszi, hogy a különböző vizuális elemek egymással harmonizáljanak a végeredményben. [32]

A moodboardot gyakran alkalmazzák a UX designer-ek. Ennek az előnye, hogy lehetővé teszi a designer-ek számára, hogy ötleteiket szervezett és áttekinthető módon mutassák be, és segítségével kreatív munkát hozzanak létre.

A moodboard elemei:

- **Képek:** A moodboardnak jelentős részét képezik. Ezek lehetnek, fényképek, rajzok, minták vagy bármilyen más vizuális elem.
- **Színek:** A színek nagyon fontosak a moodboard készítésekor. Ha jól választjuk ki őket, akkor az segít abban, hogy a látványvilág egységes és hangulatos legyen.
- **Betűtípusok:** Az üzenet könnyű olvashatósága és érthetősége érdekében fontos a megfelelő betűtípusok kiválasztása és elrendezése.
- **Szövegek:** A szövegek alkalmazásával az alkotók részletesebben és pontosabban fejezhetik ki céljaikat.
- **Hangulatjelek:** Az alkotók a hangulatjelek használatával érzelmileg hatékonyabbá tehetik a végeredményüket.

Számos online alkalmazás létezik, amelyek lehetőséget biztosítanak moodboard készítésére, mint például a Canva [33], Milanote [34], Pinterest [35], Niice [36] és sok más. Az alkalmazásomban a Canva grafikai tervezőt és szerkesztő eszközt alkalmaztam a moodboard elkészítésére. A moodboard elkészítésekor számomra fontos volt, hogy olyan színek és ábrák szerepeljenek rajta, amelyek a felhasználók számára egy letisztult és modern felhasználói felületet biztosítanak majd az alkalmazás használata során. (3.1. ábra)



3.1 ábra – Az alkalmazás moodboardja

Az alkalmazásom megvalósítása során az ábrán látható színeket választottam, amelyeket hexadecimális kód formájában adtam meg. Az ábrán jól láthatóan megjelennek a kiválasztott színek, amelyek mindegyike fontos szerepet játszik az alkalmazás felépítésében és használhatóságában. A fehér színt alkalmaztam az alkalmazás háttérszínének, ami segít a felhasználónak a szemek fáradásának csökkentésében. A fekete szín pedig a betűk színének beállítására szolgált, amelyek jól láthatóak és könnyen olvashatóak maradnak. A sötétebb kék szín szerepe a gombok és az események nevének színeként jelenik meg, amely segít a könnyű navigációban és a kiválasztásban. A világosabb kék színt pedig az appbarnál használtam fel, ami segíti a felhasználói felület letisztultságát és modernitását.

### 3.2. Logo

Az alkalmazásban való logó használata fontos, mivel az az egyik első dolog, amit a felhasználók látnak, amikor letöltik az alkalmazást. A logó az alkalmazás arculatának kiemelkedő eleme, amely segít a felhasználóknak azonosítani és megkülönböztetni az alkalmazást más applikációktól.

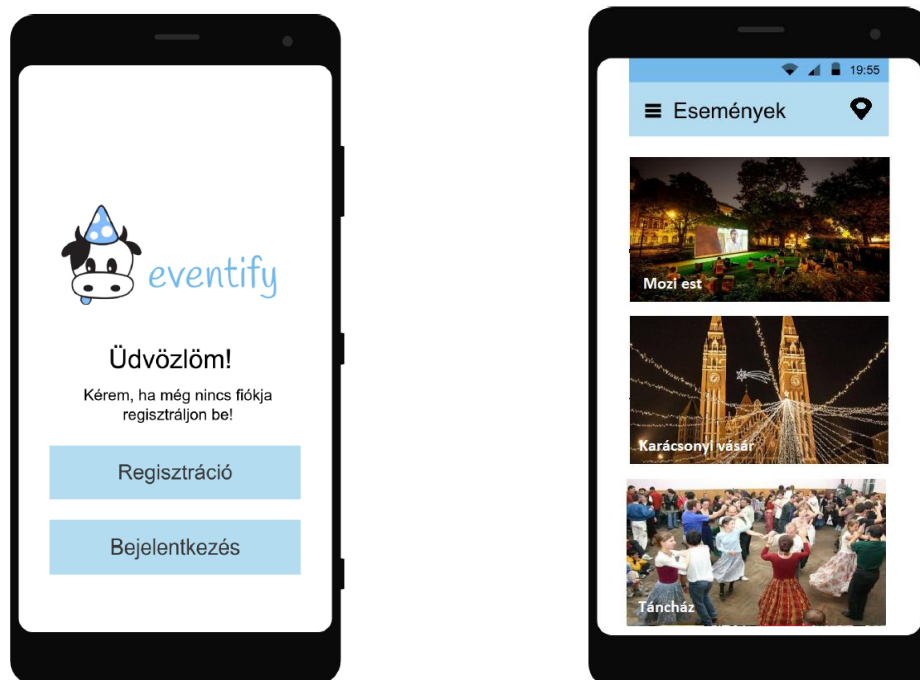


*3.2 ábra – Az alkalmazás logója*

Az alkalmazás logójának elkészítésekor számomra az volt a fontos szempont, hogy olyan designet hozzak létre, amely megfelel az alkalmazás témájának. A logó tervezése során figyelembe vettem az alkalmazás célját, funkcióit, valamint az általa nyújtott élményt, és az ehhez passzoló színeket, formákat és betűtípusokat használtam. A cél az volt, hogy a logóval vonzóvá tegyem az alkalmazást a felhasználók számára. (3.2 ábra)

## 4. Képernyőtervek

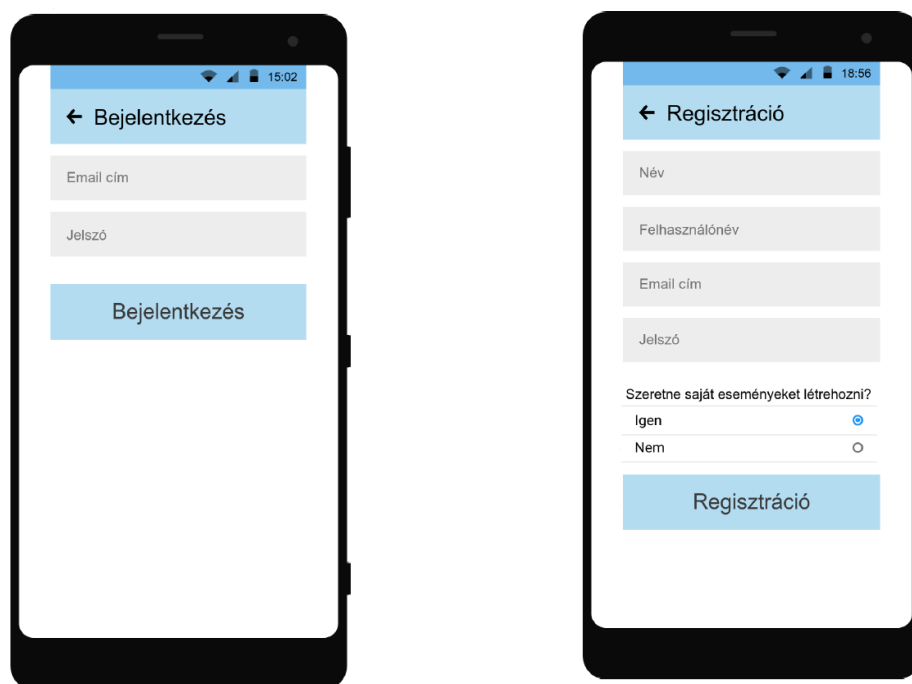
Az alkalmazás tervezése és fejlesztése során kiemelten fontos volt számomra, hogy megfelelő képernyőtervek készüljenek az alkalmazásról, amelyek segítették a fejlesztés folyamatát. Az előre elkészített tervek lehetővé tették számomra, hogy már a fejlesztés kezdetén világos elképzelésem legyen az alkalmazás kialakításáról, funkcionalitásáról és felhasználói felületéről.



4.1 ábra – Az alkalmazás képernyőtervei

Az alkalmazásomhoz készített képernyőterveket (4.1. és 4.2. ábrák) a Proto.io szolgáltatás felhasználásával terveztem meg. A Proto.io [37] egy online platform és eszközkészlet, amely lehetővé teszi a felhasználók számára a prototípusok és vizuális tervek létrehozását és tesztelését. Számos platformon működik, beleértve a webet, az iOS-t és az Androidot is. A Proto.io-t választottam, mert könnyen használható, így lehetővé tette számomra, hogy a tervezési folyamatot gyorsan és hatékonyan végezzem el. A Proto.io segítségével vizuálisan tervezhettem meg az alkalmazás különböző képernyőit, és áttekintést kaphattam az alkalmazás funkcióiról és felépítéséről.





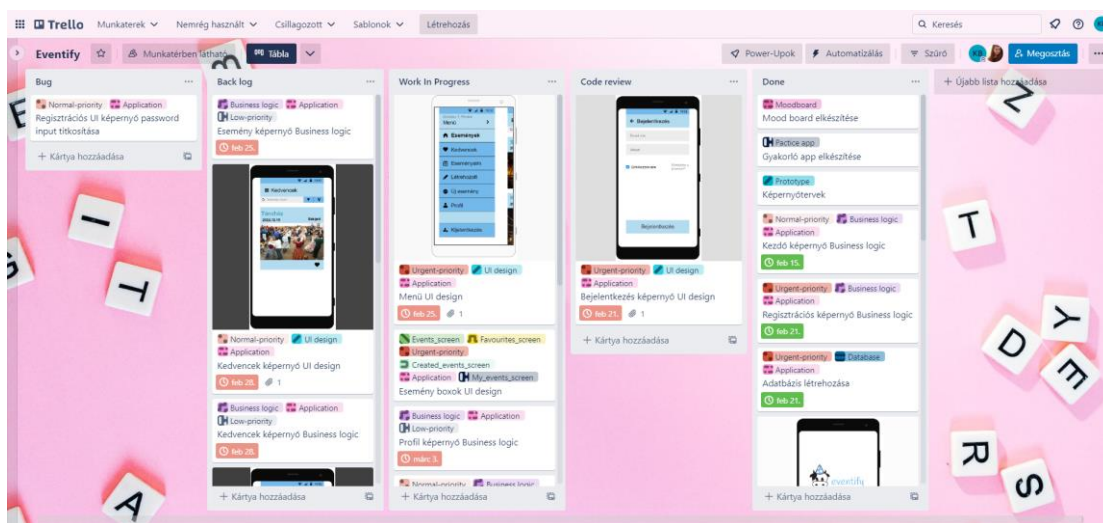
*4.2 ábra – Az alkalmazás képernyőtervei*

Az alkalmazás képernyőinek tervezésekor számomra kulcsfontosságú volt, hogy egy olyan felhasználói felületet tervezzek, amely képes lekötni a felhasználók figyelmét. Ezen felül fontos volt számomra, hogy az alkalmazás témájának megfelelő design elemeket alkalmazzam a képernyőkön, ami az alkalmazás megjelenését és felismerhetőségét is erősíti.

## 5. Fejlesztési folyamat

### 5.1. Trello

A Trello [20] egy olyan online feladatkezelő alkalmazás, amely lehetővé teszi a felhasználók számára, hogy hatékonyan szervezzék és kezeljék az egyéni és csapatprojektjeiket. Az alkalmazás egy táblarendszert használ, ahol a felhasználók létrehozhatnak különböző listákat, majd ezeket a listákat egyszerűen mozgathatják, címkézhetik, és a kártyákat rendezhetik a projektek vagy feladatok állapotától függően.



5.1 ábra – Az alkalmazáshoz tartozó Trello felület

Az alkalmazásom fejlesztése során a hatékony és átlátható fejlesztés érdekében felhasználtam a Trello (5.1. ábra) szolgáltatásait. Létrehoztam az szükséges oszlopokat, mint a „Backlog”, „Work in progress”, „Code review” és „Done”. A „Backlog” oszlop tartalmazza az összes megvalósítani kívánt feladatot, amelyeknek az előrehaladását rendszeresen frissítettem. Az „Work in progress” oszlopban a jelenleg folyamatban lévő feladatok kaptak helyet. A „Code review” oszlop szolgált arra, hogy a már elkészült kódok ellenőrzésre kerüljenek, mielőtt a „Done” oszlopba kerülnek. A „Done” oszlop pedig az összes elkészült feladat található.

A Trelloban a kártyáinkat személyre is szabhatjuk, adhatunk hozzá címkéket, csatolmányt csatolhatunk hozzá és akár határidőt is állíthatunk be az egyes feladatoknak.

A fejlesztési folyamat könnyebb nyomonkövethetőségének érdekében a Trelloban található feladataimhoz, címkéket, csatolmányokat, illetve határidőket rendeltem hozzá.

A Trello lehetővé teszi, hogy egyedi címkéket hozzunk létre, amelyek például jelölhetik a feladatok állapotát, fontosságát. A címkék segítségével könnyedén tudtam csoportosítani és megkülönböztetni az egyes feladataimat.

A Trelloban található feladataimnak határidőket is állítottam, amelyek segítségével hatékonyan tudtam beosztani az időben, és biztosítani tudtam azt, hogy az elvégzendő feladatok a megfelelő időben és sorrendben végezhessem el.

## **5.2. GitHub**

A GitHub [31] egy webes alapú verziókezelő platform, amely lehetővé teszi a fejlesztők számára, hogy nyomon kövessék a forráskód változásait, és megoszthassák a kódjukat másokkal.

Az alkalmazásom fejlesztése során a GitHubot használtam, ahol létrehoztam egy repositoryt az alkalmazásom számára „Szakdolgozat” néven. Az egyszerűbb nyomonkövetés és az átlátható fejlesztési folyamat érdekében külön brancheket hoztam létre minden egyes képernyőnek és az azokhoz tartozó üzleti logikának. A fejlesztési folyamat során gyakran alkalmaztam a git rebase parancsot is, amely két branch változtatásait egyesíti, és általa könnyebb követni és megérteni a változtatásokat a repositoryban.

A GitHub review folyamatot is biztosít. Ez a funkció lehetővé tette számomra, hogy az alkalmazáshoz tartozó kódokat és egyéb fájlokat ellenőrizzek, mielőtt azok belekerülnének a main branchre. Ezáltal megakadályozva azt is, hogy helytelen, nem odaillő kódok kerüljenek bele az alkalmazásba. Lehetőség van a kódokhoz megjegyzéseket, észrevételeket fűzni, amelyek az adott hibára utalnak, így segítve a hibák javításának folyamatát.

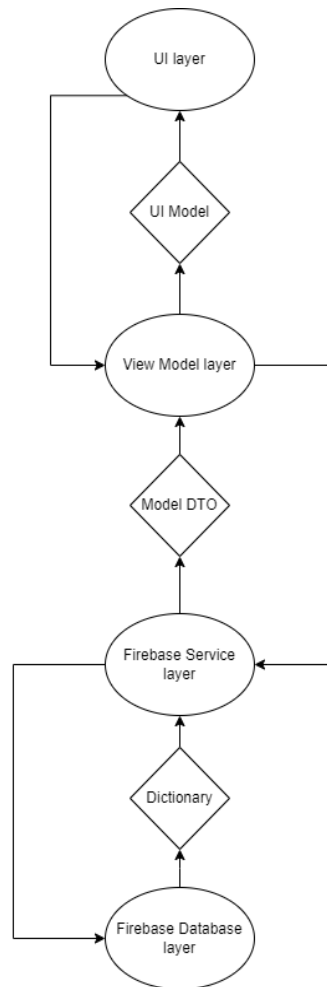
## 6. Alkalmazás szerkezeti felépítése

### 6.1. MVVM

Az MVVM (Model-View-ViewModel) architektúra egy olyan tervezési minta, amely hatékonyan támogatja a felhasználói felülettel rendelkező alkalmazások tervezését és fejlesztését. Az MVVM architektúra legfontosabb elve az üzleti logika és a felhasználói felület elkülönítése, ami lehetővé teszi a két réteg egymástól független fejlesztését és tesztelését. Az MVVM architektúra három fő részből áll: a modellből (Model), a nézetből (View) és a nézetmodellből (ViewModel). A modell a megjelenítendő statikus információkat tartalmazza, a nézet az adatok megjelenítéséért és a felhasználói interakciók kezeléséért felelős, míg a nézetmodell összeköti a modellt és a nézetet, valamint itt található az üzleti logika is.

Az MVVM architektúra használatának előnyei közé tartozik a hatékonyabb és strukturáltabb kódolás, valamint az alkalmazások könnyebb tesztelhetősége, mivel a különböző rétegeket függetlenül lehet tesztelni. [16, 17]

Az általam fejlesztett alkalmazásban az MVVM architektúrát (6.1. ábra) alkalmazom, amely lehetővé teszi számomra, hogy egy jól strukturált és átlátható kódbázist hozzak létre. Az MVVM segítségével sikerült elkülönítenem az alkalmazás üzleti logikáját és a felhasználói felületet, így ezeket függetlenül lehet tesztelni és karbantartani. A jól elkülönített rétegek lehetővé teszik az egyszerűbb cserélhetőséget, ha például egy adott réteget ki kell cserélni egy másik megoldásra.



6.1 ábra – Az alkalmazás MVVM diagrammja

A View vagy más néven UI réteg a felhasználói felületet jelenti, ahol a felhasználók az alkalmazással interakcióba lépnek. A View réteg a felhasználói eseményeket kezeli és továbbítja őket a ViewModel rétegnek.

A ViewModel felelős a felhasználói interakciók feldolgozásáért, a megjelenítési logika irányításáért és az alkalmazásállapot kezeléséért.

A Model réteg az alkalmazás adatmodelljéért felelős, amely az adatokat tartalmazza.

A Model DTO (Data Transfer Object) az adatok átvitele során használt objektumokat jelenti. Ezek az objektumok a View Model és a Firebase [4] Service rétegek közötti adatátvitelre szolgálnak.

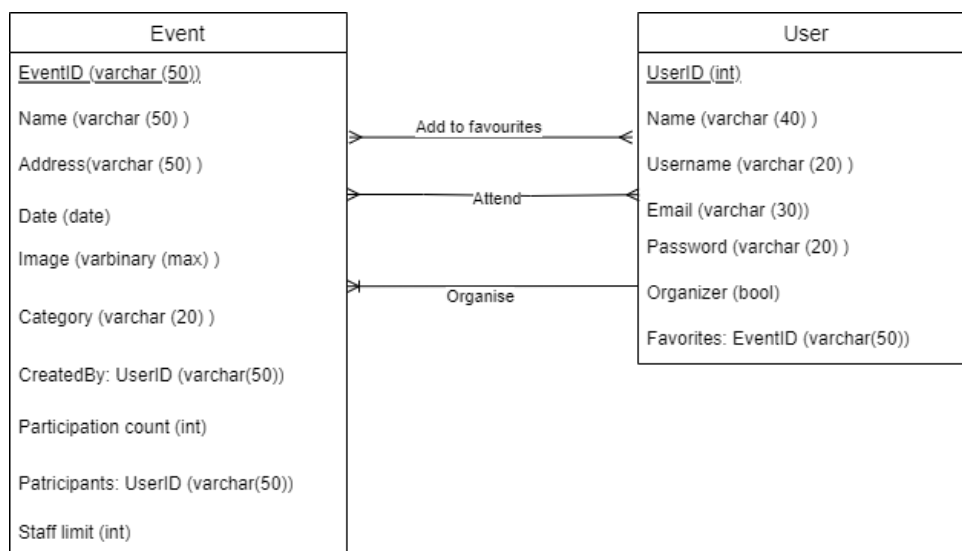
A Firebase Service réteg az alkalmazás által használt Firebase szolgáltatásokat kezeli, például a Firebase Auth-t, Firebase Storage-t, Firebase Cloud Messaginget, és a Firebase Realtime Database-t.

A Dictionary két réteg kommunikációjára használt objektum.

A Firebase Database réteg az adatbázis adatok tárolására szolgál. A Firebase Database használatával az alkalmazások biztonságosan képesek tárolni, lekérdezni és frissíteni az adatokat.

## 6.2. Adatbázis

Az alkalmazásom adatbázisaként a Firebase és Firestore szolgáltatásokat vettem igénybe. A Firebase és Firestore adatbázisok biztonságosan tárolják és kezelik az adatokat, és egyszerűen integrálhatóak az alkalmazás kódjába. Emellett számos egyéb funkciót is biztosítanak, például az autentikációt és a felhőalapú tárolást, amelyek előnyöket nyújtanak az alkalmazásom számára, és lehetővé teszik a felhasználók biztonságos és hatékony kezelését.



6.2 ábra – Az alkalmazáshoz tartozó adatbázis diagrammja

Az adatbázisban két különböző tábla található: (6.2. ábra) az "Event" és "User" táblák. Az "Event" tábla az eseményekre vonatkozó adatokat tartalmaz, beleértve a nevet, helyszínt, dátumot, képet, kategóriát és a létszámkorlátot. Minden eseménynek saját egyedi azonosítója van. A "User" tábla a felhasználók adatait tartalmazza, beleértve az azonosítót, a nevet, a felhasználónevet, az e-mail címet, a jelszót és azt, hogy szervező-e vagy sem. A táblák között a részvétel és a kedvencekhez adás esetében több a többhöz kapcsolatok állnak fenn, mivel egy felhasználó részt vehet több eseményen is, és egy esemény több felhasználó részvételét is tartalmazhatja, ugyanez vonatkozik a kedvencekhez adáshoz is.

Az események képeinek tárolásához a Firebase Storage-t használtam. A Firebase Storage egy könnyen használható felhőalapú tárhely, amely lehetővé teszi a képek

feltöltését és letöltését a felhasználók számára. Az események képeinek tárolása a Firebase Storage-ban kényelmes és biztonságos megoldás volt az alkalmazásom számára.

### **6.3. Üzleti logika**

Az üzleti logika az alkalmazás azon része, amely felelős az adatok kezeléséért, az adatbázis interakciókért, folyamatok irányításáért. [18]

Az alkalmazásomban az üzleti logika réteg rendkívül fontos szerepet tölt be. Ez a réteg felelős a felhasználók autentikálásáért és autorizálásáért, a bemenetek ellenőrzéséért és a hibakezelésért, valamint az adatfeldolgozásért.

A felhasználó autentikálása az a folyamat, amikor a felhasználó azonosítja magát egy rendszerben, például egy alkalmazásban vagy webhelyen. Az autentikáció általában a felhasználónév és jelszó, email cím és jelszó, vagy valamilyen biometrikus azonosítással történik. Az alkalmazásomban a felhasználók autentikálása a Firebase Authentication segítségével történik, amelyhez az email cím és jelszó megadása szükséges. Az autentikáció során a Firebase minden felhasználó számára generál egy egyedi azonosítót, amely a felhasználók könnyebb beazonosítását teszik lehetővé. [19]

Az autorizáció a felhasználó jogosultságainak ellenőrzése, amely lehetővé teszi, hogy a felhasználó csak azokhoz az adatokhoz férjen hozzá, amelyekhez engedélyezve van a hozzáférése. Az alkalmazásomban kétféle szerepkör betöltésére van lehetősége a felhasználónak. A jogosultság attól függ, hogy a regisztrációt követően a felhasználó szeretne-e az alkalmazásban eseményeket létrehozni. A regisztrációs felületen található választógombok segítségével két különböző szerepkör közül lehet választani az alkalmazásban. Az egyik szerepkör a szervező, míg a másik a normál felhasználót jelöli. Ez az alkalmazásban az „isOranizer” változóban tárolódik, amely értéke igaz vagy hamis lehet, a regisztrációkor kiválasztott értéktől függően.

Az alkalmazásomban az üzleti logika réteg ellenőrzi a bemeneti adatokat, hogy azok megfelelnek-e az elvárt típusnak vagy formátumnak. Ez az ellenőrzés azért fontos, hogy az alkalmazás ne dolgozzon fel érvénytelen adatokat. Az alkalmazásban a bemenetek ellenőrzése a regisztráció, bejelentkezés (6.3 ábra) és új esemény felvételénél történik.

```

String? validateEmail(String value) {
    if (value.isEmpty) {
        return mustEnterEmailErrorMessage;
    } else if (!value.contains("@")) {
        return wrongEmailErrorMessage;
    }
    return null;
}

String? validatePassword(String value) {
    if (value.isEmpty) {
        return mustEnterPasswordErrorMessage;
    } else if (value.length < 6) {
        return validatePasswordErrorMessage;
    }
    return null;
}

```

6.3 ábra – Az alkalmazásban használt bejelentkezés képernyőhöz tartozó validáció

A hibakezelés elengedhetetlen részét képezi az alkalmazásnak, mivel ez kezeli a váratlan interakciókat az alkalmazás működése során. Az üzleti logika rész figyeli az alkalmazás állapotát, és ha egy hiba történik akkor azt jelzi a felhasználó számára. A hibakezelés minden képernyőhöz megvan valósítva az alkalmazásban.

Az alkalmazásban található különböző függvények végzik az adatok feldolgozását, melyek az alkalmazás különböző részeinél alkalmazódnak, például a regisztrációnál, bejelentkezésnél, események létrehozásánál és lekérdezésénél. Az adatfeldolgozás részét képezi az adatok átalakítása is a megfelelő formátumba. Ilyen átalakítás az alkalmazás `fehMarkersFromEventData()` függvényben található, ahol adatbázisban tárolt helyszín adatok átalakítása történik a térkép által értelmezhető szélességi és hosszúsági koordináták átalakításává. (6.4 ábra)



```

Future<void> fetchMarkersFromEventData(BuildContext context) async {
  try {
    final List<EventDTO> eventDTO = await service.getEventsForToday();
    final List<EventModel> eventModels =
      eventDTO.map((dto) => EventModel.fromDTO(dto)).toList();
    Iterable<Future<Marker>> futureList =
      eventModels.map((EventModel eventData) async {
        List<Location> locations = await locationFromAddress(eventData.address);
        LatLng latLng =
          LatLng(locations.first.latitude, locations.first.longitude);
        return Marker(
          onTap: () {
            Navigator.push(
              context,
              MaterialPageRoute(
                builder: (context) => EventScreen(eventData),
              ),
            );
          },
          markerId: MarkerId(eventData.id),
          position: latLng,
          infoWindow: InfoWindow(
            title: eventData.name,
          ),
        );
      }).toSet();
    _markers = (await Future.wait(futureList)).toSet();
    errorMessages = [];
  } catch (e) {
    if (e.toString().isNotEmpty) {
      errorMessages = [e.toString()];
    } else {
      errorMessages = [standardErrorMessage];
    }
  }
  notifyListeners();
}

```

6.4 ábra – Az alkalmazás térképen történő események megjelenítéséért felelős függvény

## 6.4. Az alkalmazásban használt csomagok

Alkalmazásaink hatékonyabb és gyorsabb fejlesztése érdekében használhatjuk a Flutter által biztosított csomagokat. Ezek a csomagok elérhetőek a pub.dev [3] weboldalon. A Flutter projektünkben a csomagok telepítése és használata könnyedén megvalósítható a "pubspec.yaml" fájl segítségével.

Az általam fejlesztett alkalmazásban igénybe vettem néhány ilyen csomagot. Ezek a csomagok a 6.5 ábrán láthatóak.

```
dependencies:
  flutter:
    sdk: flutter
  firebase_core: ^2.10.0
  cloud_firestore: ^4.5.2
  provider: ^6.0.5
  firebase_storage: ^11.1.1
  form_builder_image_picker: ^3.1.0
  google_maps_flutter: ^2.2.5
  geocoding: ^2.1.0
  geolocator: ^9.0.2
  permission_handler: ^10.2.0
  firebase_core_web: ^2.3.0
  firebase_auth_web: ^5.3.2
  google_maps_flutter_web: ^0.4.0+8
  cupertino_icons: ^1.0.5
  firebase_database: ^10.1.1
  firebase_auth: ^4.4.2
  fluttertoast: ^8.2.1
  intl: ^0.17.0
```

6.5 ábra – Az alkalmazásban használt csomagok

Az általam használt csomagok rövid leírása:

- `firebase_core`: Segítségével inicializálható és konfigurálható a Firebase projekt, amely lehetővé teszi a többi Firebase szolgáltatás, például az adatbázis, az autentikáció, a felhő tárolás stb. használatát.
- `cloud_firestore`: Ez az adatbázis használható a Flutter alkalmazásban adatok tárolására, lekérdezésére és szinkronizálására a felhővel.
- `provider`: Egy állapotkezelési megoldás a Flutter alkalmazásokban, amely lehetővé teszi az alkalmazás állapotának kezelését és az állapot megosztását a widget fában.
- `firebase_storage`: Ez használható a Flutter alkalmazásban fájlok (például képek, hangfájlok stb.) feltöltésére, letöltésére és kezelésére a Firebase felhő tárolójában.
- `form_builder_image_picker`: Ezzel a csomaggal egyszerűen lehetőség van képfeltöltő űrlapok és űrlapelemek létrehozására az alkalmazásban.
- `google_maps_flutter`: Ezzel a csomaggal lehetőség van térképek megjelenítésére, interaktív térképi funkciók használatára és térképes alkalmazások fejlesztésére.
- `geocoding`: A csomag lehetővé teszi a címek és koordináták közötti konvertálást a Flutter alkalmazásban.

- geolocator: Ezzel a csomaggal lehetőség van az eszköz aktuális helyzetének lekérdezésére.
- permission\_handler: Ezzel a csomaggal lehetőség van az engedélykérésekre, az engedélyek státuszának ellenőrzésére és az engedélykérések eredményének kezelésére az alkalmazásban.
- firebase\_core\_web: Ennek a csomagnak az alkalmazásával az alkalmazásban a Firebase SDK inicializálható és konfigurálható web platformra.
- firebase\_auth\_web: Lehetővé teszi a Firebase Authentication használatát az alkalmazásban a webes platformon.
- google\_maps\_flutter\_web: Ez a csomag pedig a Google Maps integrációját biztosítja az alkalmazásban webes platformon.
- cupertino\_icons: Ezzel a csomaggal lehetőség van az iOS stílusú ikonok használatára a Flutter alkalmazásban.
- firebase\_database: Ezzel a csomaggal lehetőség van az adatok tárolására, lekérdezésére és szinkronizálására a Firebase felhő adatbázisában a Flutter alkalmazásban.
- firebase\_auth: Ezzel a csomaggal lehetőség van felhasználók regisztrálására, bejelentkezésére, kijelentkezésére, jelszó visszaállítására és más azonosítási feladatokra a Flutter alkalmazásban.
- fluttertoast: Ezzel a csomaggal lehetőség van rövid üzenetek, figyelmeztetések vagy hibaüzenetek megjelenítésére a felhasználónak rövid időre a képernyőn.
- intl: Segítségével lehetőség van dátumok, pénznemek formátumának beállítására.

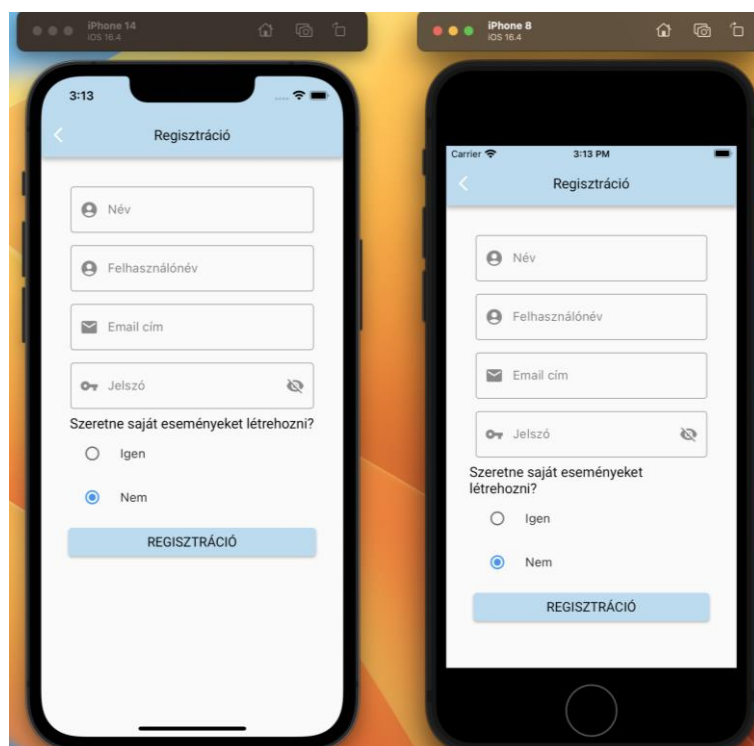
#### **6.4.Kész alkalmazás**

Az alkalmazás indulásakor a kezdő képernyő (6.6. ábra) fogad minket, ahol gombok segítségével választhatunk, hogy szeretnénk-e regisztrálni vagy bejelentkezni az alkalmazásba. Az alkalmazás rezponzivitásának köszönhetően különböző készülékeken is könnyedén és zökkenőmentesen használható.



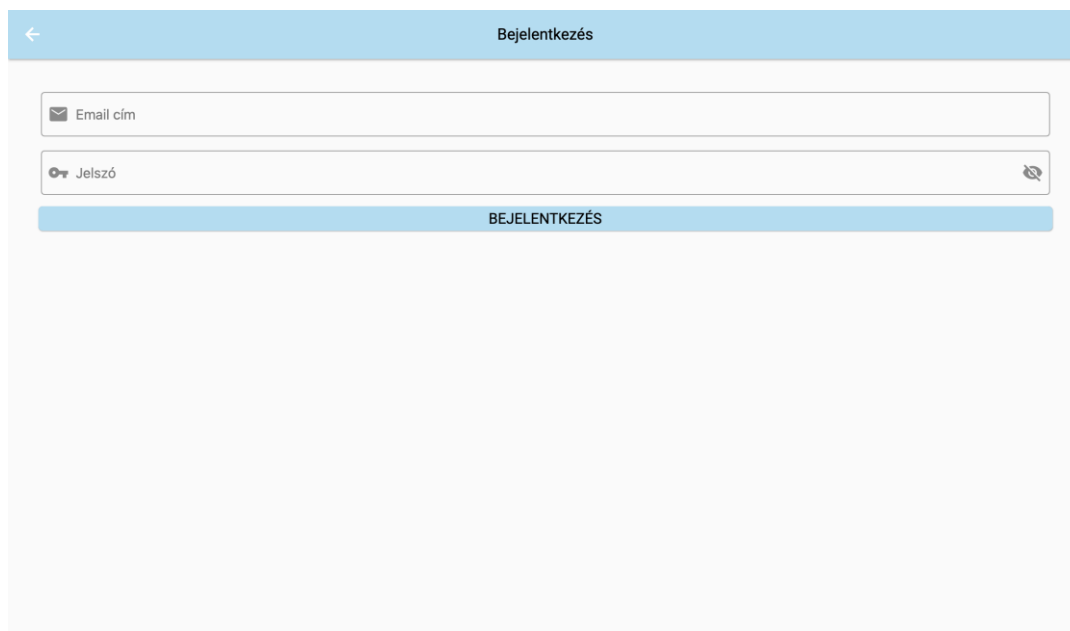
6.6 ábra – Az alkalmazás kezdőképernyője webes felületen

A regisztráció opcióval az új felhasználók könnyedén létrehozhatnak egy fiókot az alkalmazásban, míg a bejelentkezés lehetőséggel az már regisztrált felhasználók azonosíthatják magukat és hozzáférhetnek a fiókjukhoz. Ez az egyszerű és felhasználóbarát kezdő képernyő lehetővé teszi a felhasználók számára, hogy gyorsan és egyszerűen elkezdjék használni az alkalmazást.



6.7 ábra – Az alkalmazás regisztráció képernyője iOS készüléken

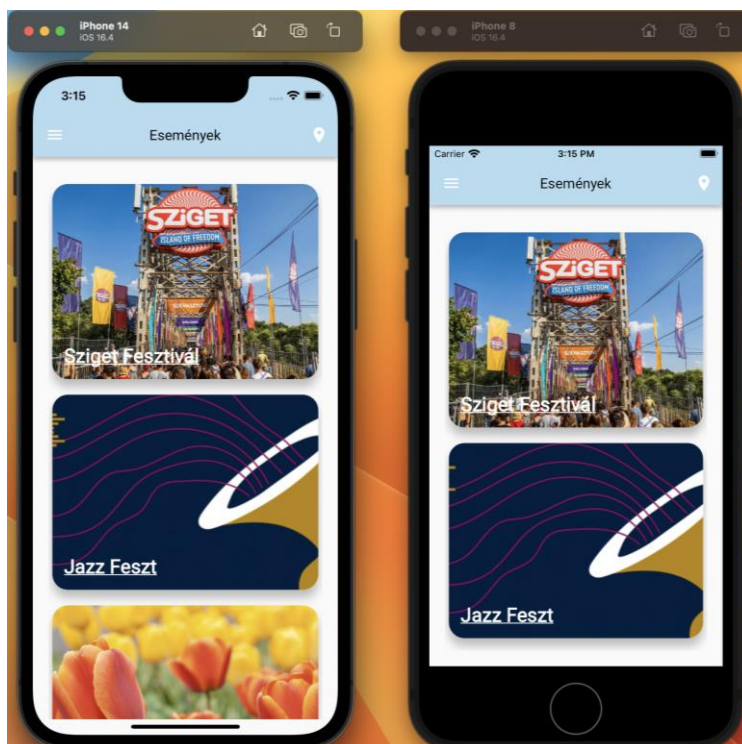
Amennyiben még nem rendelkezünk fiókkal az alkalmazásban, úgy a regisztrációs képernyőn (6.7. ábra) a szükséges adatok megadásával létrehozhatunk egy fiókot. Ha a felhasználó szeretne saját eseményeket létrehozni az alkalmazásban, akkor a regisztráció során ezt is jelölheti a választógombok segítségével. Amennyiben a regisztrációs űrlapon hiányosan vagy hibásan töltjük ki a szükséges mezőket, például elhagyjuk az email-címünket, vagy nem felel meg a jelszavunk a biztonsági követelményeknek, az alkalmazás hibaüzenetet küld és nem enged minket bejelentkezni. A hibaüzenetek tartalmazzák az okot, ami miatt a regisztráció sikertelen volt, és javaslatot tesznek arra, hogy miként lehetne orvosolni a problémát. Ha a hibaüzeneteket figyelmen kívül hagyjuk, nem tudunk belépni az alkalmazásba, és nem tudjuk élvezni az események és programok széles kínálatát.



6.8 ábra – Az alkalmazás bejelentkezés képernyője webes felületen

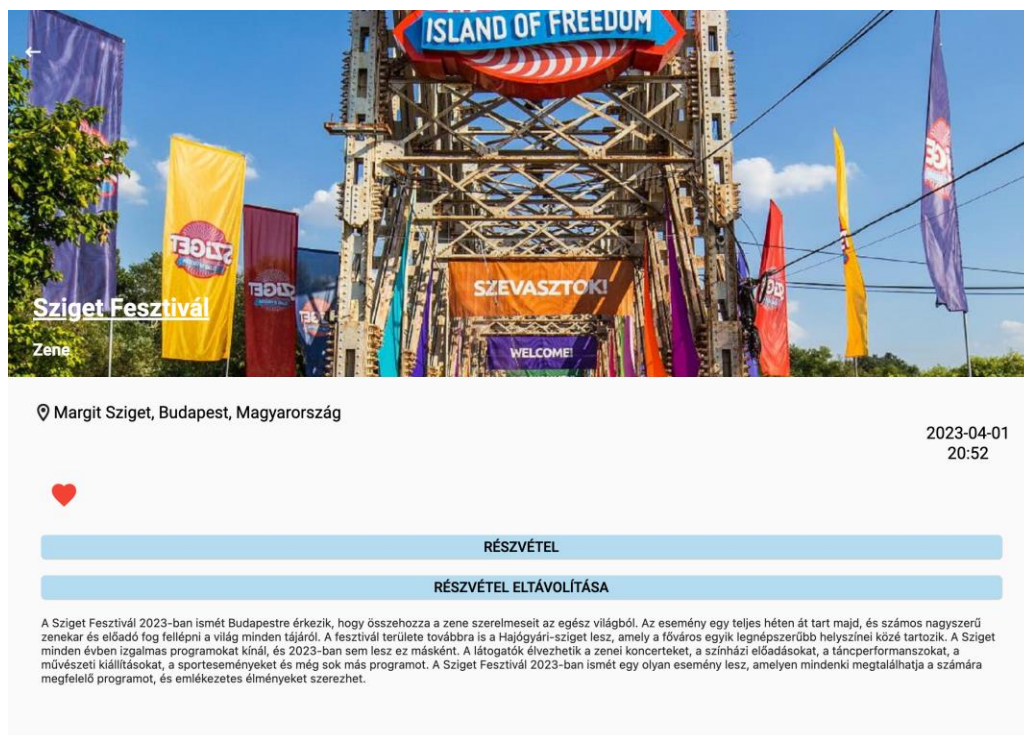
Amennyiben már korábban regisztráltunk az alkalmazásban, akkor egyszerűen és gyorsan beléphetünk a fiókunkba az email címünk és a jelszavunk megadásával. (6.8. ábra)

A bejelentkezés és regisztráció az alkalmazás használatának előfeltétele. A bejelentkezést, illetve a regisztrációt követően a főképernyőre (6.9. ábra) navigál át minket az alkalmazás. A főképernyő appbarja tartalmazza a képernyő nevét, a menü ikont és a térkép ikont. A képernyő közepén esemény kártyák jelennek meg, amelyek az esemény képét, illetve a nevét tartalmazzák. Amennyiben érdekel minket az egyik esemény, akkor az esemény képére kattintva az esemény képernyőjére jutunk.



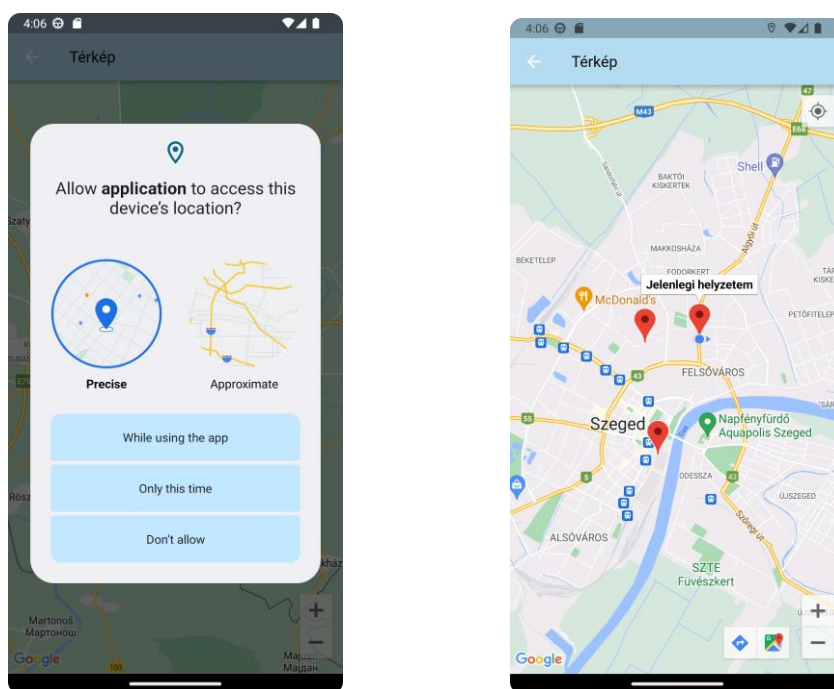
6.9 ábra – Az alkalmazás főképernyője iOS készüléken

Az „Esemény” képernyő (6.10. ábra) tartalmazza az esemény pontos időpontját, a helyszínt, valamint egy rövid leírást az eseményről. A képernyőn található szív ikon segítségével hozzáadhatjuk az adott eseményt a kedvenceink listájához, vagy amennyiben már korábban hozzáadtuk el is távolíthatjuk azt. A sikeres hozzáadásról és eltávolításról értesítést kapunk. Az alkalmazásban lehetőség van arra, hogy jelezzük részvételi szándékunkat egy eseményen, illetve korábban jelzett részvételünket el is távolíthatjuk. Az alkalmazás értesít minket a sikeres részvételről vagy eltávolításról. Ha szeretnénk egy eseményt eltávolítani a részvételi listánkról, a "Részvétel eltávolítása" gomb csak akkor fogja ezt megtenni, ha már korábban hozzáadtuk az eseményt. Ha nem jeleztük részvételi szándékunkat, és szeretnénk az eseményt eltávolítani, az alkalmazás visszajelzést ad az "Ön még nem vesz részt az eseményen!" üzenettel. Ha az esemény létszámkorlátja megtelt, az alkalmazás arról is tájékoztat bennünket, hogy "A részvétel meghaladja a szabad helyek számát!". Ezáltal az alkalmazás átlátható és informatív módon segíti a felhasználókat az eseményeken való részvétel kezelésében. Az esemény képernyőről könnyedén visszavigázhatunk a főképernyőre az appbar segítségével.



6.10 ábra – Az alkalmazás esemény képernyője webes felületen

Ha a térkép ikonra kattintunk az alkalmazásban, átnavigál minket a "Térkép" képernyőre (6.11. ábra). Itt, miután engedélyeztük a helymeghatározást, láthatjuk a térképen az eseményeket jelölő marker ikonokat, valamint a saját helyzetünket. Ha egy esemény marker ikonjára kattintunk, az alkalmazás automatikusan átnavigál minket az adott esemény képernyőjére.

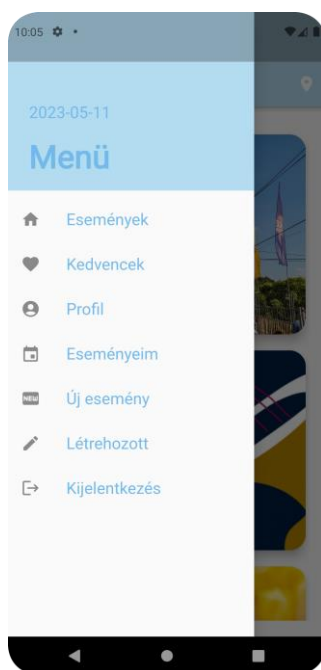


6.11 ábra – Az alkalmazás térkép képernyője android készüléken



A menü ikonra kattintva megjelenik számunkra az alkalmazás menüje. Az alkalmazás kétféle „Menü” képernyővel (6.12. ábra) rendelkezik, ami attól függ, hogy a regisztrációkor a „Szeretne saját eseményt létrehozni?” lehetőségkor melyik lehetőséget választottuk. Amennyiben az „Igen” opciót választottuk, akkor a „Menü” az alábbi lehetőségeket tartalmazza:

- Események: Az alkalmazás visszavigál minket a főképernyőre.
- Kedvencek: Az alkalmazás a kedvencek képernyőre navigál minket.
- Profil: Az alkalmazás a profil képernyőre navigál minket.
- Eseményeim: Az alkalmazás az eseményeim képernyőre navigál minket.
- Új esemény: Az alkalmazás az új esemény létrehozásához szükséges képernyőre navigál minket.
- Létrehozott: Az alkalmazás az általunk létrehozott események képernyőjére navigál minket.
- Kijelentkezés: Az alkalmazásból való kijelentkezés után a kezdőképernyőre navigál minket az alkalmazás.

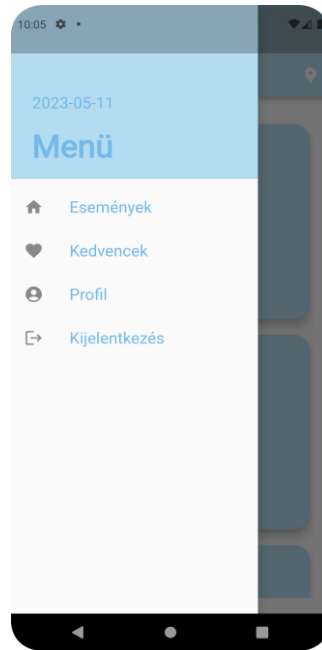


6.12 ábra – Az alkalmazás menü képernyője android készüléken

Ha viszont a „Nem” opciót választottuk a regisztráció során, akkor nincs lehetőségünk saját eseményt létrehozni. Ebben az esetben a „Menü” az alábbiakat tartalmazza: (6.13. ábra)



- Események: Az alkalmazás visszanavigál minket a főképernyőre.
- Kedvencek: Az alkalmazás a kedvencek képernyőre navigál minket.
- Profil: Az alkalmazás a profil képernyőre navigál minket.
- Eseményeim: Az alkalmazás az eseményeim képernyőre navigál minket.
- Kijelentkezés: Az alkalmazásból való kijelentkezés után a kezdőképernyőre navigál minket az alkalmazás.



*6.13 ábra – Az alkalmazás menü képernyője android készüléken*

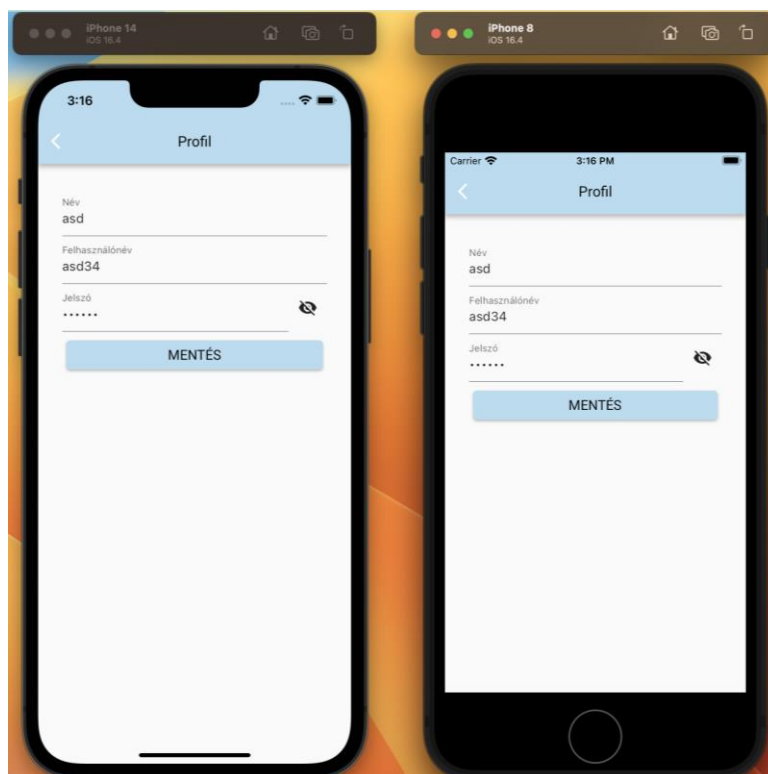
Az alkalmazás mindkét "Menü" képernyőjén az aktuális dátum is megjelenik, így könnyen nyomon követhető, hogy az adott napon milyen eseményeken lehet részt venni.

Az alkalmazás „Kedvencek” képernyőjére (6.14. ábra) való navigálást követően megtekinthetjük az általunk korábban kedvencként jelölt eseményeket és programokat. Itt rendezetten láthatjuk a kedvencek közé felvett eseményeket, amelyekre kattintva az esemény oldalára jutunk. Emellett lehetőség van arra is, hogy eltávolítsuk az adott eseményt a kedvenceink közül, ha már nem érdekel minket. A kedvencek képernyője egy praktikus funkció, amely segít nekünk abban, hogy mindig naprakész legyünk azokkal az eseményekkel és programokkal kapcsolatban, amelyek iránt érdeklődünk, és hogy könnyen visszataláljunk azokhoz.



6.14 ábra – Az alkalmazás kedvencek képernyője android készüléken

A „Profil” képernyőre (6.15. ábra) navigálva az alkalmazás mindig az aktuális felhasználó adatait jeleníti meg, ideértve a nevét, felhasználónevét és jelszavát. Ezen a képernyőn a felhasználónak lehetősége van az adatai módosítására is, például a felhasználónevének, jelszavának vagy nevének megváltoztatására.



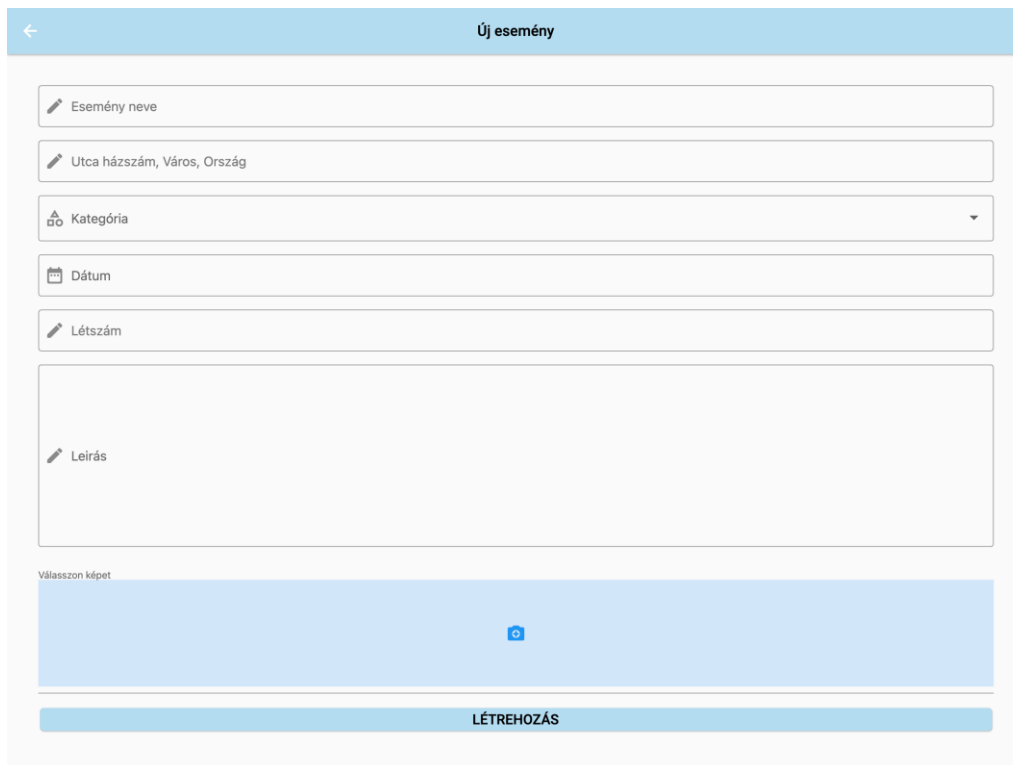
6.15 ábra – Az alkalmazás profil képernyője iOS készüléken

Az "Eseményeim" képernyőn (6.16. ábra) az alkalmazás azon eseményeket listázza, amelyekre a felhasználó részvételt jelzett a "Részvétel" gomb megnyomásával. Ez az opció lehetővé teszi a felhasználók számára, hogy könnyedén nyomon követhessék azokat az eseményeket, amelyekre részt vesznek, és hogy egy helyen láthassák az összes eseményt, amelyen részt vesznek vagy részt kívánnak venni. Ha egy felhasználó részt vesz egy eseményen, az esemény információi automatikusan megjelennek az "Eseményeim" képernyőn.



6.16 ábra – Az alkalmazás eseményeim képernyője android készüléken

Az "Új esemény" képernyő (6.17. ábra) célja az, hogy a felhasználók saját eseményeket hozhassanak létre az alkalmazásban. Az esemény létrehozása során számos adatot meg kell adni, mint például az esemény neve, helyszíne és kategóriája. A felhasználóknak továbbá ki kell választaniuk a dátumot, amikor az esemény megrendezésre kerül, és rövid leírást kell írniuk az eseményről. Az esemény látogatottságának korlátozásához létszámkorlátot kell beállítani, és az eseményhez feltétlenül mellékelni kell egy képet, amely segíti a felhasználókat az esemény könnyebb azonosításában. Amennyiben a felhasználó elmulasztja kitölteni valamelyik szükséges mezőt az „Új esemény” oldalon, akkor az alkalmazás hibaüzenettel fogja tájékoztatni erről és nem engedi a hozzáadást. Az üres mezőkre az alkalmazás felhívja a figyelmet, hogy azokat ki kell tölteni a sikeres közzététel érdekében. Ezzel biztosítva van, hogy csak teljes és pontos információkkal rendelkező események kerüljenek közzétételre.



6.17 ábra – Az alkalmazás új esemény képernyője webes felületen

A "Létrehozott" képernyőn (6.18. ábra) azok az események találhatók, amelyeket a felhasználó saját maga hozott létre az alkalmazásban. A felhasználó a képernyőn találja az általa létrehozott eseményeket.



6.18 ábra – Az alkalmazás létrehozott esemény képernyője android készüléken

## Irodalomjegyzék

- [1] Flutter - <https://flutter.dev/>
- [2] Dart - <https://dart.dev/overview>
- [3] Pub.dev - <https://pub.dev>
- [4] Firebase - <https://firebase.google.com/>
- [5] Stateless Widget in Flutter - <https://www.educative.io/answers/what-are-stateless-widgets-in-flutter>
- [6] A brief history of UX design and its evolution – <https://thenextweb.com/news/a-brief-history-of-ux-design-and-its-evolution>
- [7] 13 Best UI/UX Design Practices for Mobile Apps – <https://medium.com/@designgrapes/13-best-ui-ux-design-practices-for-mobile-apps-13b44f8317d7>
- [8] Where UX Comes From - <https://www.uxbooth.com/articles/where-ux-comes-from/>
- [9] UX és a UI design jelentése és használata – <https://creativesite.hu/webdesign-blog/ux-es-az-ui-design-jelentese-es-hasznalata>
- [10] Stateful vs Stateless Widget – <https://medium.com/flutter-community/flutter-stateful-vs-stateless-db325309deae>
- [11] Layout widgets - <https://docs.flutter.dev/ui/widgets/layout>
- [12] What is Native App Development – <https://mdevelopers.com/blog/what-is-a-native-mobile-app-development->
- [13] Natív, hybrid vagy cross-platform app – <https://www.broduction.net/hu/blog/native-vs-hybrid-vs-crossplatform>
- [14] User Experience (UX) Design - <https://www.interaction-design.org/literature/topics/ux-design>
- [15] Mobil app fejlesztés - <https://www.broduction.net/hu/blog/mobile-app-development-2022-guide>
- [16] MVVM – <https://medium.com/upday-devs/android-architecture-patterns-part-3-model-view-viewmodel-e7eeee76b73b>
- [17] The Model-View-ViewModel Pattern – <https://learn.microsoft.com/en-us/xamarin/xamarin-forms/enterprise-application-patterns/mvvm>
- [18] What Is Business Logic? - <https://www.indeed.com/career-advice/career-development/business-logic>
- [19] What is authentication? - <https://www.techtarget.com/searchsecurity/definition/authentication>
- [20] Trello – <https://trello.com/hu>
- [21] React Native - <https://reactnative.dev/>
- [22] Xamarin Forms - <https://dotnet.microsoft.com/en-us/apps/xamarin/xamarin-forms>
- [23] Ionic - <https://ionicframework.com/>
- [24] Android - <https://www.android.com/>
- [25] Kotlin - <https://kotlinlang.org/>
- [26] Java - <https://www.java.com/en/>
- [27] iOS - <https://www.apple.com/hu/>

- [28] Objective C - <https://developer.apple.com/library/archive/documentation/Cocoa/Conceptual/ProgrammingWithObjectiveC/Introduction/Introduction.html>
- [29] Swift - <https://www.swift.com/>
- [30] Control widgets - <https://docs.flutter.dev/ui/widgets/material#buttons>
- [31] GitHub - <https://github.com/>
- [32] Moodboard - <https://milanote.com/guide/create-better-moodboards>
- [33] Canva - <https://www.canva.com/create/mood-boards/>
- [34] Milanote - <https://milanote.com/>
- [35] Pinterest - <https://hu.pinterest.com/>
- [36] Niice - <https://niice.co/>
- [37] Proto.io - <https://proto.io/>

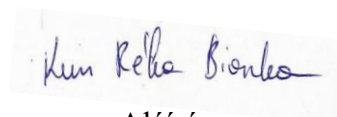
## Nyilatkozat

Alulírott, Kun Réka Bianka programtervező informatikus BSc szakos hallgató, kijelentem, hogy a dolgozatomat a Szegedi Tudományegyetem, Informatikai Intézet Számítógépes Algoritmusok és Mesterséges Intelligencia Tanszékén készítettem, programtervező informatikus BSc diploma megszerzése érdekében.

Kijelentem, hogy a dolgozatot más szakon korábban nem védtem meg, saját munkám eredménye, és csak a hivatkozott forrásokat (szakirodalom, eszközök, stb.) használtam fel.

Tudomásul veszem, hogy szakdolgozatomat / diplomamunkámat a Szegedi Tudományegyetem Diplomamunka Repoitóriumban tárolja.

Dátum 2023. május 12.

A handwritten signature in blue ink, reading "Kun Réka Bianka", is placed on a small, light-colored rectangular piece of paper.

Aláírás

## **Köszönetnyilvánítás**

Szeretnék ezúton köszönetet nyilvánítani mindazoknak, akik támogattak abban, hogy ez a szakdolgozat elkészülhessen. Köszönettel tartozom a barátaimnak és családomnak, akik állandóan bíztattak és támogattak, amikor nehéz pillanatokat éltem át a szakdolgozat írása során.

Külön szeretnék köszönetet mondani, Kiss-Vetráb Mercedesnek, aki időt és energiát szánt rám, valamint hasznos tanácsokkal és javaslatokkal látott el. A szakdolgozat készítésének folyamata alatt sokat tanulhattam tőle és nagyra értékelem a szaktudásáért, és tapasztalatáért.

Végezetül, köszönöm mindenkinek, aki hozzájárult ahhoz, hogy ezt a szakdolgozatot elkészíthettem.