

**SAPIENTIA ERDÉLYI MAGYAR TUDOMÁNYEGYETEM  
MAROSVÁSÁRHELYI KAR,  
INFORMATIKA SZAK**



**SAPIENTIA  
ERDÉLYI MAGYAR  
TUDOMÁNYEGYETEM**

EventsApp

**DIPLOMADOLGOZAT**

Témavezetők:  
Olteán-Péter Boróka,  
Tanársegéd  
Dr. Farkas Csaba,  
Egyetemi tanár

Végzős hallgató:  
Páll Arnold-Barna

**2023**

**UNIVERSITATEA SAPIENTIA DIN CLUJ-NAPOCA  
FACULTATEA DE ȘTIINȚE TEHNICE ȘI UMANISTE,  
INFORMATICĂ**



**UNIVERSITATEA  
SAPIENTIA**

EventsApp

**LUCRARE DE DIPLOMĂ**

Coordonator științific:  
Olteán-Péter Boróka,  
Asistent Universitar  
Dr. Farkas Csaba,  
Profesor universitar

Absolvent:  
Páll Arnold-Barna

**2023**

**SAPIENTIA HUNGARIAN UNIVERSITY OF  
TRANSYLVANIA**  
**FACULTY OF TECHNICAL AND HUMAN SCIENCES**  
**COMPUTER SCIENCE SPECIALIZATION**



**SAPIENTIA**  
HUNGARIAN UNIVERSITY  
OF TRANSYLVANIA

EventsApp

**BACHELOR THESIS**

Scientific advisor:  
Olteán-Péter Boróka,  
Assistant Professor  
Dr. Farkas Csaba,  
Full Professor

Student:  
Páll Arnold-Barna

**2023**

UNIVERSITATEA „SAPIENTIA” din CLUJ-NAPOCA  
Facultatea de Științe Tehnice și Umaniste din Târgu Mureș  
Programul de studii: Informatică

Viza facultății:

## LUCRARE DE DIPLOMĂ

Coordonator științific: <b>Dr. Farkas Csaba</b> Îndrumător: <b>Oltean-Péter Boróka</b>	Candidat: Pál Arnold-Barna Anul absolvirii: 2020
---	---

**a) Tema lucrării de licență:**

EventsApp

**b) Problemele principale tratate:**

Scopul aplicațiilor de urmărire a evenimentelor este de a permite utilizatorilor să-și urmărească evenimentele preferate pe baza informațiilor relevante, precum oră, locație și descriere.

Aplicația utilizează limbajul de programare Swift, cadrul SwiftUI pentru crearea interfețelor cu utilizatorul și platforma de dezvoltare Firebase pentru gestionarea bazelor de date.

Documentarea adecvată a software-lui

**c) Desene obligatorii:**

UML Use Case Diagram

Grafice despre structura bazei de date

Grafice despre structura codului

Desene despre design-ul aplicației

**d) Softuri obligatorii:**

Implementarea backend-ului software-ului este în Swift, implementarea interfeței utilizatorului este în SwiftUI, iar dezvoltarea bazei de date se face cu Firebase.

**e) Bibliografia recomandată:**

Moroney, L., & Moroney, L. (2017). The firebase realtime database. The Definitive Guide to Firebase: Build Android Apps on Google's Mobile Platform, 51-71.

Nekrasov, A. (2022). SwiftUI. In Swift Recipes for iOS Developers: Real-Life Code from App Store Apps (pp. 319-341). Berkeley, CA: Apress.

**f) Termene obligatorii de consultații:** săptămânal

**g) Locul și durata practicii:** Universitatea „Sapientia” din Cluj-Napoca,  
Facultatea de Științe Tehnice și Umaniste din Târgu Mureș, sala / laboratorul 416

Primit tema la data de: 20.06.2022

Termen de predare: 07.07.2023

Semnătura Director Departament

Semnătura responsabilului  
programului de studiu

Semnătura coordonatorului

Semnătura candidatului

### **Declarație**

Subsemnatul/a Paul Amador-Bârba, absolvent(ă) al/a specializării  
Informatică, promoția 2020, cunoscând prevederile Legii Educației Naționale 1/2011 și a Codului de etică și deontologie profesională a Universității Sapientia cu privire la furt intelectual declar pe propria răspundere că prezenta lucrare de licență/proiect de diplomă/disertație se bazează pe activitatea personală, cercetarea/proiectarea este efectuată de mine, informațiile și datele preluate din literatura de specialitate sunt citate în mod corespunzător.

Localitatea, Tg. MUREŞ  
Data: 2023.06.16

Absolvent

Semnătura.....Paul Amador-Bârba.....

# Kivonat

A mai felgyorsult világban egyre több minden történik körülöttünk és ez miatt az események száma is csak egyre jobban nő, így ezeket napról napra nehezebb követni. Számos alkalmazásnak van esemény követő funkciója, az egyik legnagyobb és talán a legismertebb az a Facebook, viszont ott a hangsúly inkább az emberek közötti interakcióra kerül, így az esemény követő funkciónalitása talán nem a legjobb. A mi hazánkban a hasonló célra tervezett alkalmazások használata talán nem olyan elterjedt, mint a külföldi országokban, viszont kihívást is jelenthet megtalálni azt az alkalmazást amelyik kitudja elégíteni a felhasználó összes elvárásait.

Az esemény követő alkalmazások célja, hogy az általunk preferált eseményeket tudjuk követni, időpont, helyszín meg egyéb fontos információ alapján, mint például a leírás.

Dolgozatomban egy ilyen tipusú aplikációt hozok létre, amelyet a telefonunk segítségével elérünk. Manapság már szinte mindenki rendelkezik egy okos mobillal, amely mindenkorban van bárhol is megyünk. Ezek alapján egy ilyen alkalmazás csak segíteni tud a minden nap felgyorsult világban.

A projektem célkitűzése egy olyan iOS applikáció létrehozása, amely segítségével könnyebben tudjuk követni a számunkra előnyben részesített eseményeket, jutunk információkhöz az adott eseményekről, illetve a részvétel tervezésében is segítséget nyújthat. Az applikáció elkészítéséhez Swift programozási nyelv van használva, SwiftUI keretrendszer a felhasználói felületek létrehozásához, illetve Firebase fejlesztési platform az adatbázis kezeléséhez.

**Kulcsszavak:** esemény, esemény követés, információk megszerzése, tervezés, Swift, SwiftUI, Firebase, adatbázis.

# Rezumat

În lumea rapidă de astăzi, tot mai multe lucruri se întâmplă în jurul nostru și, din acest motiv, numărul evenimentelor este în creștere, deci este mai dificil să le urmăm zi de zi. Multe aplicații au funcții de urmărire a evenimentelor, una dintre cele mai mari și poate cea mai cunoscută este Facebook, dar acolo accentul se pune mai mult pe interacțiunea dintre oameni, astfel încât funcționalitatea trackerului de evenimente poate să nu fie cea mai bună. În țara noastră, utilizarea aplicațiilor concepute în scopuri similare poate să nu fie la fel de răspândită ca în țările străine, dar poate fi o provocare să găsești o aplicație care să satisfacă toate așteptările utilizatorului.

Scopul aplicațiilor de urmărire a evenimentelor este de a putea urmări evenimentele preferate în funcție de oră, locație și alte informații importante, cum ar fi descrierea.

În teza mea creez o aplicație de acest tip, pe care o putem accesa cu ajutorul telefonului nostru. În zilele noastre, aproape totă lumea are un mobil inteligent care este întotdeauna cu noi oriunde mergem. Pe baza acestui fapt, o astfel de aplicație poate ajuta doar în lumea rapidă de zi cu zi.

Obiectivul proiectului meu este de a crea o aplicație iOS care să ne ajute să urmărim evenimentele preferate, să obținem informații despre evenimentele date și să ne ajute, de asemenea, să ne planificăm participarea. Aplicația utilizează limbajul de programare Swift, cadrul SwiftUI pentru crearea interfețelor cu utilizatorul și platforma de dezvoltare Firebase pentru gestionarea bazelor de date.

**Cuvinte de cheie:** eveniment, urmărire evenimentelor, obținerea de informații, planificare, Swift, SwiftUI, Firebase, bază de date.

# Abstract

In today's fast-paced world, more and more things are happening around us, and because of this, the number of events is only increasing, so it is more difficult to follow them day by day. Many apps have event tracking features, one of the biggest and perhaps the most well-known is Facebook, but there the emphasis is more on interaction between people, so the functionality of event tracker may not be the best. In our country, the use of applications designed for similar purposes may not be as widespread as in foreign countries, but it can be challenging to find an application that can satisfy all the expectations of the user.

The purpose of event tracking applications is to be able to track your preferred events based on time, location and other important information such as description.

In my thesis I create an application of this type, which we can access with the help of our phone. Nowadays, almost everyone has a smart mobile that is always with us wherever we go. Based on this, such an application can only help in the everyday fast-paced world.

The objective of my project is to create an iOS application that can help us follow our preferred events, get information about the given events, and can also help us plan our participation. The application uses Swift programming language, SwiftUI framework for creating user interfaces, and Firebase development platform for database management.

**Keywords:** event, event tracking, obtaining information, planning, Swift, SwiftUI, Firebase, database.

# Tartalomjegyzék

<b>1. Bevezető</b>	<b>12</b>
1.1. Eventbrite . . . . .	13
1.2. Eventful . . . . .	13
1.3. Meetup . . . . .	13
1.4. Bandsintown . . . . .	13
<b>2. Alkalmazás funkcionalitásai</b>	<b>14</b>
2.1. Bejelentkezés . . . . .	14
2.2. Regisztrálás . . . . .	14
2.3. Események listájának megtekintése . . . . .	15
2.4. Esemény részletes megtekintése . . . . .	15
2.5. Esemény lementése a kedvencek közé . . . . .	15
2.6. Kedvenc események listájának megtekintése . . . . .	15
2.7. Kedvenc esemény részletes megtekintése . . . . .	15
2.8. Esemény törlése a kedvencek közül . . . . .	16
2.9. Kedvenc események megtekintése az aktuális nap alapján . . . . .	16
2.10. Felhasználó fiókjának adatainak a megváltoztatása . . . . .	16
2.11. Alkalmazás témajának megváltoztatása . . . . .	16
2.12. Kijelentkezés . . . . .	16
2.13. Új esemény hozzáadása (admin felhasználó) . . . . .	16
2.14. Esemény módosítása/törlése (admin felhasználó) . . . . .	16
<b>3. Technológiai áttekintés</b>	<b>17</b>
3.1. Xcode . . . . .	17
3.2. Swift . . . . .	17
3.3. SwiftUI . . . . .	18
3.4. iOS . . . . .	18
3.5. Git . . . . .	18
3.6. GitHub . . . . .	18
3.7. Sourcetree . . . . .	18
3.8. Firebase . . . . .	18
<b>4. Követelmények</b>	<b>19</b>
4.1. Felhasználói követelmények . . . . .	19
4.1.1. Hagyományos felhasználói követelmények . . . . .	19
4.1.2. Admin felhasználói követelmények . . . . .	20

4.2. Nem funkcionális követelmények . . . . .	21
4.3. Funkcionális követelmények . . . . .	22
<b>5. A projekt architektúrája</b>	<b>23</b>
5.1. Az MVVM architektúra . . . . .	23
5.1.1. Az MVVM és MVC közötti különbség . . . . .	24
5.1.2. Modell komponens feladata . . . . .	24
5.1.3. Nézet komponens feladata . . . . .	24
5.1.4. Nézetmodell komponens feladata . . . . .	24
5.1.5. MVVM előnyei . . . . .	25
5.2. Adatbázis . . . . .	25
5.2.1. Firebase Realtime Database . . . . .	25
5.2.2. Adatbázis szerkezete . . . . .	25
5.3. Osztályok . . . . .	27
<b>6. Felhasználói kézikönyv</b>	<b>29</b>
6.1. Splash képernyő . . . . .	29
6.2. Bejelentkezés . . . . .	30
6.3. Regisztrálás . . . . .	30
6.4. Főoldal . . . . .	31
6.5. Oldalsó menü . . . . .	32
6.6. Beállítások . . . . .	33
6.6.1. Felhasználó adatainak a módosítása . . . . .	34
6.6.2. Kijelentkezés . . . . .	35
6.7. Események listája . . . . .	36
6.8. Esemény részletei és hozzáadás a kedvencekhez . . . . .	37
6.9. Kedvenc események megtekintése . . . . .	38
6.10. Kedvenc esemény részletei és törlés a kedvencek közül . . . . .	39
6.11. Admin oldal . . . . .	40
<b>7. Megvalósítás</b>	<b>42</b>
7.1. Bejelentkezés . . . . .	42
7.2. Regisztrálás . . . . .	43
7.3. Kijelentkezés . . . . .	43
7.4. Események lekérése . . . . .	44
7.5. Esemény hozzáadása a kedvencekhez . . . . .	44
7.6. Kedvenc események lekérése . . . . .	45
7.7. Esemény törlése a kedvencek közül . . . . .	45
7.8. Felhasználó fiók adatainak a megváltoztatása . . . . .	46
7.8.1. Felhasználó név megváltoztatása . . . . .	46
7.8.2. Felhasználó jelszavának megváltoztatása . . . . .	46
7.9. Új esemény hozzáadása (admin felhasználó) . . . . .	46
7.10. Esemény módosítása/törlése (admin felhasználó) . . . . .	47
7.10.1. Esemény módosítása . . . . .	47
7.10.2. Esemény törlése . . . . .	48
7.11. Verziókezelés . . . . .	48

Összefoglaló	50
Köszönetnyilvánítás	51
Ábrák jegyzéke	52
Irodalomjegyzék	53

# 1. fejezet

## Bevezető

Manapság egyre nehezebb találni egy olyan esemény követő alkalmazást amely kitudja elégíteni minden személy vagy felhasználó igényeit és még ezek mellett ne is legyen bonyolult a használata. Az egyre csak jobban felgyorsuló mai világban minden idő fontos az ember számára, akár percekre lebontva is, így ezt egyre többször nehezebb beosztani a körülöttünk lévő eseményekre. Ebből kifolyólag egyre csak több az olyan eset, hogy lemaradunk egy eseményről vagy bizonyos információ marad ki az eseményről ami miatt a részvétel szempontjából nem érünk oda időben vagy akár túl korán oda érünk vagy olyan is megtörténhet, hogy akár a helyszínre utaló információ marad ki ami miatt az aki részt szeretne venni az adott eseményen nem tudja, hogy az hol lesz megtartva és, hogy hogyan jusszon el oda.

Dolgozatom célja egy ilyen alkalmazás elkészítése, mint mobil app. Egy olyan alkalmazás, amely segítségével tudomást lehet szerezni a körülöttünk történő eseményekről, segíteni tud betervezni a részvételt egy adott eseményhez, illetve lehetőséget ad arra a felhasználónak, hogy külön helyen szerepeljenek azok az események amelyeken részt szeretne venni vagy akár csak jobban szeretné követni őket. Ennek megvalósítására készítettem egy olyan mobil alkalmazást, amely használata nem okoz nehézséget bármilyen korosztályú felhasználó számára, legyen az idős vagy fiatal.

Az app használásához bejelentkezés szükséges a már meglévő fiókba vagy létrehozni egyet, hogyha a személy aki használni szeretné még nem tette meg. Miután ez a lépés közül valamelyik megtörtént, utána meglehet tekinteni, hogy milyen alkalmak, rendezvények voltak, vannak és lesznek. Az események típusok, jellegük szerint kategorizálva vannak és ezek alapján lehet kiválasztani egyet amelyet a felhasználó szeretne megtekinteni. Meglehet tekinteni ezeknek az eseményeknek az információt, mint például a leírását, időkeretét amiben zajlik, a helyszínnel kapcsolatos információkat stb. Lehetőség adatik arra is, hogy kedvencek közé tároljuk az eseményeket, így könnyebb megtalálni azokat amelyekbe érdekeltek vagyunk.

Az applikáció megvalósításához egy iOS alkalmazás mellett döntöttem, mely Xcode fejlesztői környezetben jött létre, ahol a kód implementálásához Swift programozási nyelv van használva, SwiftUI keretrendszer van használva a felhasználói felületek létrehozásához, illetve Firebase fejlesztési platform van használva az adatbázis kezeléséhez. A dolgozatban bemutatásra kerül a felhasznált technológiák, eszközök amely segítségevel létrejöhetett az alkalmazás. Bemutatásra kerül a rendszer elképzelése, specifikációja,

funkcionalitásai, struktúrája, valamint milyen lehetőségek létezhetnek a továbbfejlesztéshez.

Hasonló funkcionálitású esemény követő alkalmazások már léteznek és ezekből egy pár bemutatásra kerül az alábbiakban:

### **1.1. Eventbrite**

Az Eventbrite egy széles körben használt platform események felfedezésére, szervezésére és nyomon követésére. Lehetővé teszi helyi események megtalálását, jegyek vásárlását és az események látogatottságának nyomon követését. Az Eventbrite rendelkezik egy iOS alkalmazással is amely megtalálható az AppStore-ban. [2]

### **1.2. Eventful**

Az Eventful egy átfogó eseményfelfedező és követő alkalmazás. Események széles skáláját kínálja különböző kategóriákban, és nyomon követheti kedvenc eseményeit, emlékeztetőket kaphat, és megtekintheti eseményelőzményeit. Ez rendelkezik egy iOS alkalmazással is amely ingyenesen letölthető az AppStore-ból. [3]

### **1.3. Meetup**

A Meetup egy közösségi hálózati alkalmazás, amely segít megtalálni és csatlakozni az érdeklődési körének megfelelő csoportokhoz. Lehetővé teszi a csoportok által szervezett események és tevékenységek nyomon követését, valamint válaszolhat és értesítéseket kaphat a közelgő eseményekről. Letölthető az AppStore-ból. [10]

### **1.4. Bandsintown**

A Bandsintown egy nagyszerű alkalmazás koncertek és élő zenei események nyomon követésére. Megvizsgálja a zenei könyvtárat, és eseményeket javasol az Ön preferenciái alapján. Nyomon követheti kedvenc előadói is, és értesítéseket kaphat közelgő műsorairól. Ez az alkalmazás megtalálható az AppStore-ban, ahol ingyen letölthető. [1]

## 2. fejezet

# Alkalmazás funkcionalitásai

Ebben a fejezetben bemutatásra kerülnek az alkalmazás funkcionalitásai. Az alkalmazás megnyitása és betöltése után a felhasználó előtt 2 oldal jelenhet meg különböző esetekben. Az első esetben a bejelentkezés oldal jelenik meg, abban az esetben ha a felhasználó még nem volt bejelentkezve. Ezen az oldalon lehetőség adatik arra is, hogy átnavigáljon a regisztrálás oldalra, abban az esetben ha a felhasználó nem rendelkezik fiókkal. Második esetben pedig az alkalmazás főoldala jelenik meg, abban az esetben, hogyha a felhasználó bejelentkezve maradott és újra megnyitotta az applikációt. A főoldalon lehetőség adatik arra, hogy tovább lehessen navigálni az események listájához, ahol meglehet tekinteni az összes eseményt időtartam alapján szűrve vagy akár csak egy fajta esemény típus és időtartam alapján szűrve, majd bármelyik esetet választva, ha kiválasztunk egy eseményt amit megszeretnénk tekinteni, akkor átkerülnk egy olyan oldalra ahol megtudjuk tekinteni az adott kiválasztott esemény összes részletes információját, eltudjuk menteni a kedvencek közé, illetve megtudjuk nyitni az esemény helyszínét az Apple beépített térképe segítségével. Továbbá ezen az oldalon lehetőség kerül arra, hogy meglehessen tekinteni azokat a kedvenc eseményeket amelyek aznap fognak történni, persze, hogyha a felhasználó mentett le eseményeket a kedvenc események listába, ha nem akkor ez üres marad. Az oldalon még megjelenik a bal felső sarokban egy gomb amely segítségével egy oldalsó menüt lehet megnyitni, ahol tovább lehet navigálni a beállítások oldalra, a kedvenc események lista oldalra, illetve ha admin típusú felhasználóba vagyunk bejelentkezve akkor megjelenik egy extra oldal, admin oldalnak nevezve, ahol az admin külön fajta műveleteket tud végrehajtani.

### 2.1. Bejelentkezés

Az alkalmazás egyik induló képernyője. Ha a felhasználó használni szeretné az applikációt, akkor kötelező módon kell rendelkeznie egy fiókkal, majd ebbe be a fiókba be kell jelentkezzen, hogy elérje az applikáció funkcionalitásait.

### 2.2. Regisztrálás

Erre a képernyőre a bejelentkezés oldalról lehet átnavigálni. Az a felhasználó amely szeretné használni az alkalmazást, kötelező módon létre kell hozzon egy fiókot, abban az

esetben, ha még nem rendelkezik egy már meglévővel. Itt meg kell adni a kért adatokat, majd ezek után be lehet regisztrálni és így használni lehet az alkalmazást.

## **2.3. Események listájának megtekintése**

Az alkalmazás másik induló képernyője az a főoldal. Ezen az oldalon lehetőség adatik arra, hogy átlehessen navigálni az események listájának megtekintéséhez, akár megtekintheti az összes eseményt időtartam alapján szűrve vagy esemény típus és időtartam alapján szűrve. Miután ki lett választva valamelyik opción, megjelenik előttünk az események lista, ahol böngészni lehet az események között. Itt megjelenik minden esemény képe, neve, illetve a kezdeti és végzeti időtartama összekötve.

## **2.4. Esemény részletes megtekintése**

Ha az események listája oldalon kiválasztottunk egy eseményt, amelyet megszeretnénk tekinteni részletesebben, akkor átkerülünk az esemény részletes megtekintése oldalra. Ezen az oldalon meglehet tekinteni az összes eseménnyel kapcsolatos információt, azaz a nevét, típusát, kezdeti és végzeti idejét, leírását, illetve a helyszínét az Apple beépített térképnek segítségével.

## **2.5. Esemény lementése a kedvencek közé**

Mikor egy esemény részletes oldalán vagyunk, lehetőség adatik arra, hogy az adott eseményt lementsük a kedvencek közé. Ezt úgy lehet megvalósítani, hogy az esemény neve mellett + (plusz) gombot megnyomjuk, majd megjelenik egy kis felugró ablak ami azt jelzi, hogy a művelet sikeresen végződött.

## **2.6. Kedvenc események listájának megtekintése**

Miután egy adott eseményt lementettünk a kedvencek közé, meglehet tekinteni azt az oldalt, amely tartalmazza az összes kedvencek közé lementett eseményt, lista formájában. Hogy ezt az oldalt megtekintsük, vissza kell navigálni a főoldalra, ott megnyitni az oldalsó menüt, majd kiválasztani a kedvencek elemet, majd azt megnyomva, átkerülünk arra az oldalra ami tartalmazza a kedvencek közé lementett események listáját.

## **2.7. Kedvenc esemény részletes megtekintése**

A kedvenc események oldalon a felhasználónak lehetősége van arra, hogy itt is megtekinse egy általa lementett esemény részletes információit. Ezt ugyan úgy lehet megtenni, mint az összes esemény lista vagy egy fajta típusú esemény lista oldalon való művelet végrehajtásával.

## **2.8. Esemény törlése a kedvencek közül**

Ha a felhasználó kiszeretne törölni egy eseményt a kedvencek közül, akkor ezt úgy teheti meg, hogy megnyissa a törölni kivánt esemény részletes oldalát, majd ott az esemény neve mellett megjelenő - (mínusz) gombot kell megnyomnia. Ha a művelet sikeres volt, akkor megjelenik egy kis felugró ablak.

## **2.9. Kedvenc események megtekintése az aktuális nap alapján**

A főoldalon lehetőség van arra, hogy meglehessen tekinteni azokat a felhasználó által lementett eseményeket a kedvencek közé, amelyeknek az időkeretébe beleesik az aktuális nap amelyen vagyunk. Ha nincsen ilyen, akkor a lista üres marad.

## **2.10. Felhasználó fiókjának adatainak a megváltoztatása**

Ha a felhasználó megszeretné változtatni a felhasználói nevét vagy akár jelszavát, ezt megteheti úgy, hogyha kiválassza az oldalsó menüben a beállítások oldalt, majd ott kiválassza, hogy melyik műveletet szeretné végrehajtani.

## **2.11. Alkalmazás témájának megváltoztatása**

A felhasználónak lehetősége van arra, hogy testre szabhatja az applikáció megjelenési témáját világos és sötét között. Ezt a beállítások oldalon lehet megtenni.

## **2.12. Kijelentkezés**

Ha a felhasználó kiszeretne jelentkezni, azt úgy tudja megvalósítani, hogy megnyissa az oldalsó menüben a beállítások oldalt és ott kiválassza a kijelentkezés opciót. Ha a felhasználó kijelentkezett, akkor visszakerül a bejelentkezés oldalra.

## **2.13. Új esemény hozzáadása (admin felhasználó)**

Egy másik admin művelet amit el lehet végezni, az az új esemény hozzáadása. Ezt a műveletet is az admin oldalon keresztül lehet megtenni. Itt minden szükséges adatot fel kell vinni, ahhoz, hogy egy esemény teljes legyen, hogy minden információ meglegyen róla.

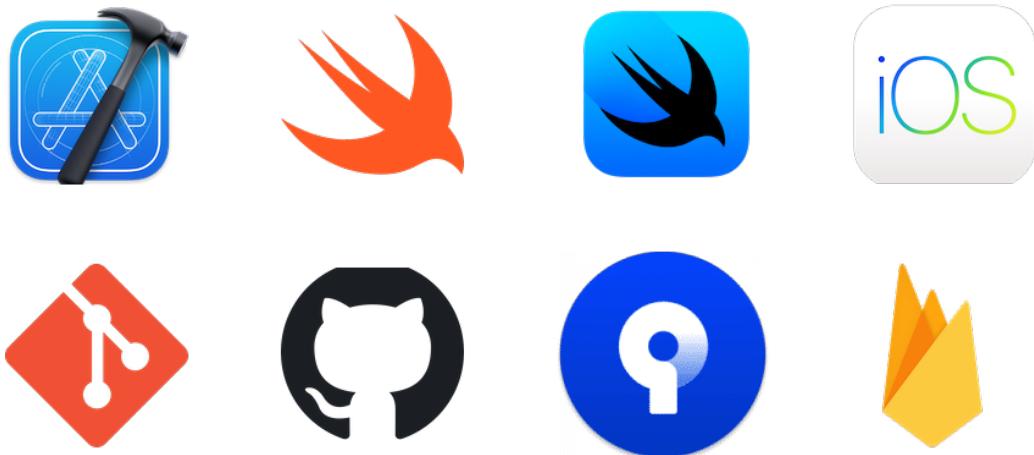
## **2.14. Esemény módosítása/törlése (admin felhasználó)**

A többi extra művelet mellett, az admin felhasználó módosítani, illetve törölni képes egy adott eseményt. Hogyha módosítani szeretné, akkor csak kikell cserélni az adatokat, hogyha pedig törölni szeretné, akkor pedig a törlés gombot kell megnyomni.

## 3. fejezet

# Technológiai áttekintés

Ebben a fejezetben a felhasznált technológiák kerülnek bemutatóra. Az alábbi [3.1](#) ábra tartalmazza a legfontosabbakat.



**3.1. ábra.** Használt technológiák

### 3.1. Xcode

Az app elkészítése Xcode fejlesztői környezetben történt meg. Elkészítését nagyon segítette a fejlesztői környezet, amely rugalmasságot, illetve számos kényelmi funkciót biztosított. Az Xcode az Apple integrált fejlesztői környezete macOS-hez, amelyet macOS, iOS, iPadOS, watchOS, tvOS és visionOS szoftverek fejlesztésére használnak. [\[17\]](#)

### 3.2. Swift

A Swift egy hatékony és intuitív programozási nyelv minden Apple platformra. A Swift használatának megkezdése egyszerű, tömör, mégis kifejező szintaxissal és modern

funkciókkal. A Swift kód biztonságos kialakítású és villámggyorsan futó szoftvert készít. [15]

### 3.3. SwiftUI

A SwiftUI segítségével nagyszerű megjelenésű alkalmazásokat lehet készíteni az összes Apple platformon a Swift erejével és meglepően kevés kóddal. Nézeteket, vezérlőket és elrendezési struktúrákat biztosít az alkalmazás felhasználói felületének deklarálásához. [16]

### 3.4. iOS

Az iOS egy mobil operációs rendszer, amelyet az Apple Inc. Kizárolag hardveréhez fejlesztett ki. Ez az operációs rendszer működteti a vállalat számos mobileszközét, beleértve az iPhone-t is. [9]

### 3.5. Git

A Git egy ingyenes és nyílt forráskódú elosztott verziókezelő rendszer, amelyet arra terveztek, hogy minden gyorsan és hatékonyan kezeljen a kicsitől a nagyon nagy projektig. [7]

### 3.6. GitHub

A GitHub egy kódtároló platform verziókezeléshez és együttműködéshez. Lehetővé teszi, hogy Ön és mások bárhonnan együtt dolgozzanak a projekteken. [8]

### 3.7. Sourcetree

A SourceTree egy ingyenes asztali alkalmazás, amelyet Git-tárolók kezelésére használnak. Olyan grafikus felületet biztosít, amely leegyszerűsíti a Git használatát, és megkönnyíti a felhasználók számára a gyakori Git-műveletek végrehajtását a parancssor használata nélkül. [14]

### 3.8. Firebase

A Firebase egy alkalmazásfejlesztési platform, amely segít a felhasználók által kedvelt alkalmazások és játékok fejlesztésében és bővítésében. A Google támogatásával és vállalkozások millióinak bizalmát élvezik szerte a világon. [6]

## 4. fejezet

# Követelmények

A legfontosabb követelmény egy olyan mobil applikáció létrehozása, amely rendelkezik egy grafikus felülettel és iOS operációs rendszerű mobil készüléken kell működjön. Az alkalmazás bejelentkezést és regisztrálást kell tudja kezelni, az események listájának megtekintését, egy kiválasztott esemény részletes megtekintését. Ezek mellett lehetőséget kell nyújtszon arra, hogy egy kiválasztott esemény le lehessen menteni a kedvenc események listába, azt a listát megtekinteni, egy adott kedvenc esemény részletes információit megtekinteni, illetve ha a felhasználónak igénye van rá, akkor az adott kedvenc esemény törlését a kedvenc események listából. Kell tudja a felhasználó adatainak a módosítását kezelni, az alkalmazás téma-jának megváltoztatását, illetve a kijelentkezést a fiókból. Ha admin fiókba van bejelentkezve, akkor tudnia kell kezelní a csak admin által elvégezhető műveleteket.

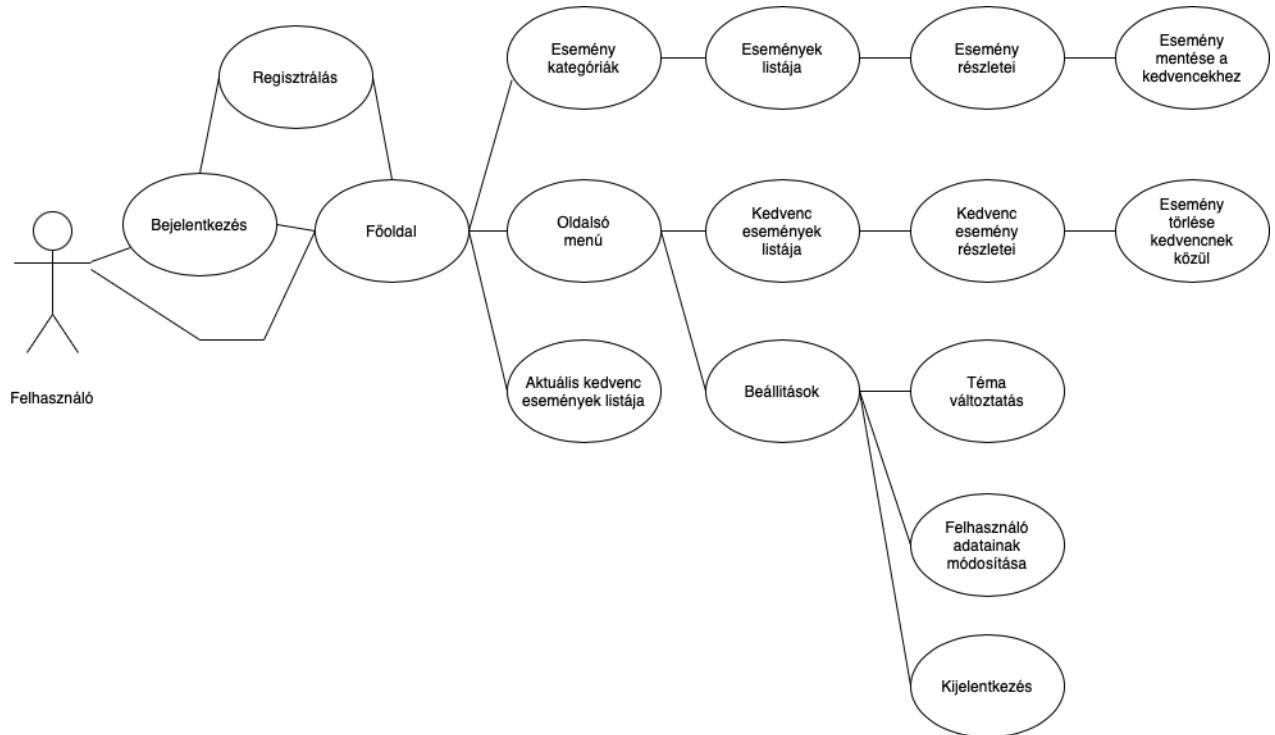
### 4.1. Felhasználói követelmények

Ebben az alfejezetben bemutatóra kerülnek a rendszer felhasználói, valamint a műveletek amiket ők végeznek el az alkalmazás használatához. A könnyebb, jobb megértés és ismertetés céljából Használat Eset Diagramok (Use Case Diagram) segítségével vannak szemléltetve a felhasználók által elvégzett műveletek.

#### 4.1.1. Hagyományos felhasználói követelmények

Ezeket a műveleteket az 4.1 ábra szemlélteti. A felhasználó mikor megnyissa az alkalmazást, akkor vagy a bejelentkezés oldal jelenik meg abban az esetben, hogyha nincsen bejelentkezve vagy bérregisztrálva, vagy maga a főoldal jelenik meg abban az esetben, hogyha be van jelentkezve és nem jelentkezett ki mikor bezárta az alkalmazást. Regisztrálás után egyből be is jelentkezeti a használót, így utána a főoldal fog megjelenni. A főoldalon meglehet nézni az aktuális időtartamon belül lévő kedvenc eseményeket, ki lehet nyitni az oldalsó menüt, illetve az esemény kategóriákat lehet megtekinteni, hogy azok alapján listázza az eseményeket vagy az összes eseményt kategória szűrő nélkül. Az események listájában kilehet választani egyet, majd annak meglehet tekinteni az információit. Az esemény részlet oldalon lehetőség adódik arra is, hogy az adott eseményt le lehessen menteni a kedvencek közé. Az oldalsó menüt kinyitva át lehet navigálni a kedvenc események lista oldalra, illetve a beállítások oldalra. A kedvenc események lista

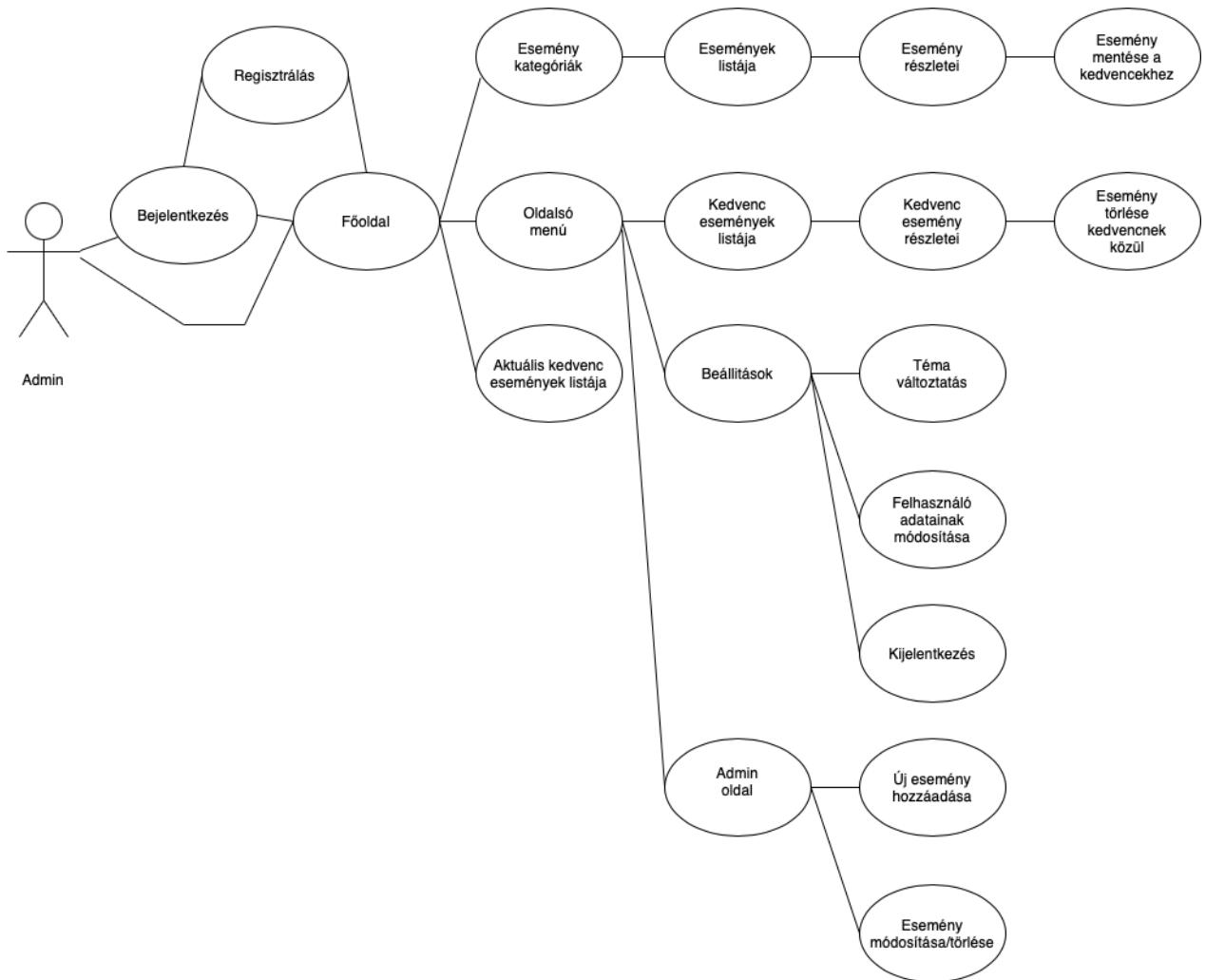
oldalon kiválasztva egyet meglehet tekinteni az információit. A kedvenc esemény részlet oldalon a felhasználónak lehetősége adódik arra is, hogy az adott eseményt kitörölje a kedvencek közül. A beállítások oldalon a felhasználó megváltoztathassa az applikáció témaját, a saját adatait meg módosíthatja, illetve ki is tud jelentkezni.



**4.1. ábra.** A hagyományos felhasználó használat eset diagramja

#### 4.1.2. Admin felhasználói követelmények

Amint az 4.2 ábra is mutatja, az admin felhasználó rendelkezik ugyanazokkal a funkcionálisokkal, mint a hagyományos felhasználó és még pár extrával. Az admin felhasználó a hagyományos felhasználóhoz képest új eseményeket tud hozzáadni az alkalmazáshoz, illetve módosítani és törölni tud egy adott eseményt.



**4.2. ábra.** Az admin felhasználó használat eset diagramja

## 4.2. Nem funkcionális követelmények

Ebben az alfejezetben bemutatóra kerülnek a nem funkcionális követelmények. A felhasználó, hogy tudja használni ezt az applikációt amely Apple telefonokra lett tervezve, kell rendelkezzen egy olyan okostelefonnal, amely megfelel a következő követelményeknek:

- iOS operációs rendszerrel rendelkező készülék
- Internet hozzáférési lehetőséggel
- Email cím

### **4.3. Funkcionális követelmények**

A rendszernek a következő funkciókkal kell rendelkeznie, amiket az alkalmazás elkészítése során le kell implementálni, amiknek a leírása részletesebben megtalálható a 2 fejezetben:

- Bejelentkezés
- Regisztrálás
- Események listájának megtekintése
- Esemény részletes megtekintése
- Esemény lementése a kedvencek közé
- Kedvenc események listájának megtekintése
- Kedvenc esemény részletes megtekintése
- Esemény törlése a kedvencek közül
- Kedvenc események megtekintése az aktuális nap alapján
- Felhasználó fiókjának adatainak a megváltoztatása
- Alkalmazás témájának megváltoztatása
- Kijelentkezés
- Új esemény hozzáadása (admin felhasználó)
- Esemény módosítása/törlése (admin felhasználó)

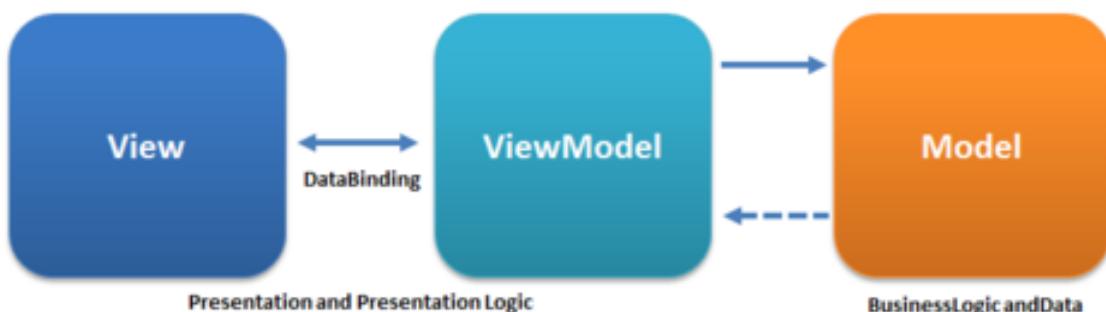
## 5. fejezet

# A projekt architektúrája

A projekt megvalósításához MVVM (Model–View–ViewModel, magyarul: Modell–Nézet–NézetModell) architekturális minta lett használva. Az MVVM minta segít tisztán elkülöníteni az alkalmazás üzleti és megjelenítési logikáját a felhasználói felülettől. Ez az architekturális minta az MVC (Model-View-Controller, magyarul: Modell-Nézet-Vezérlő) architekturális mintából származtatható. [11]

### 5.1. Az MVVM architektúra

Amikor egy alkalmazás létrehozásánál az MVVM mintát használjuk, akkor az három jól elkülöníthető részre osztja fel az alkalmazást, melyek így jobban átláthatóvá válnak, ahogy a 5.1 ábra is jól mutatja. [11]



5.1. ábra. MVVM architektúra szerkezete

Ez a minta leválasztja a grafikus felhasználói felületet az üzleti logikától (adatmodelltől). A nézetmodell egy értéktranszformátor, amely az adatok átalakításáért felelős a könnyű feldolgozás és megjelenítés érdekében. A nézetmodell inkább modell, mint nézet, de megjelenítési logikát is létrehoz. Megvalósíthat köztes programozási mintákat is, amelyek szabályozzák a hozzáférést a használati esetek megtérítéséhez. [11]

### **5.1.1. Az MVVM és MVC közötti különbség**

Az MVC architektúrában [12]:

- A vezérlő az alkalmazás belépési pontja
- Egy-a-többhöz kapcsolat a vezérlő és a nézet között.
- A nézetnek nincsen hivatkozása a vezérlőre.
- Az MVC régi modell.
- Nehéz olvasni, módosítani, egységesztelni és újra felhasználni ezt a modellt.
- Az MVC modell komponens a felhasználótól elkülönítve tesztelhető.

Az MVVM architektúrában [12]:

- A nézet az alkalmazás belépési pontja
- Egy-a-többhöz kapcsolat a nézet és a nézetmodell között.
- A nézetnek van hivatkozása a nézetmodellre.
- Az MVVM egy viszonylag új modell.
- A hibakeresési folyamat bonyolult lesz, ha összetett adatkötéseink vannak.
- Könnyen elvégezhető különálló egységeszteléshez és a kód eseményvezérelt.

### **5.1.2. Modell komponens feladata**

A modell az alkalmazás adatait és üzleti logikáját képviseli. Magában foglalja az alkalmazás működéséhez kapcsolódó adatstruktúrákat, állapotot és műveleteket. A modell réteg kölcsönhatásba léphet adatbázisokkal vagy más adatforrásokkal az adatok beolvasása és kezelése érdekében. [11]

### **5.1.3. Nézet komponens feladata**

A nézet a felhasználói felület felhasználó számára látható összetevőit jelöli. Felelős az adatok megjelenítéséért és a felhasználói bevitel rögzítéséért. A nézet passzív, és nem tartalmaz üzleti logikát. Ehelyett a nézetmodellhez kötődik, hogy frissítse a felhasználói felületet az alapul szolgáló adatok változásai alapján.[11]

### **5.1.4. Nézetmodell komponens feladata**

A nézetmodell közvetítőként működik a nézet és a modell között. Tartalmazza a megjelenítési logikát, az adatátalakításokat és a felhasználói interakciókat kezelő parancsokat. A nézetmodell olyan tulajdonságokat és parancsokat tesz elérhetővé, amelyekhez a nézet kötést köthet, lehetővé téve a nézet számára az adatok megjelenítését és frissítését. Emellett kommunikál a modell réteggel az adatok lekéréséhez és frissítéséhez. [11]

### 5.1.5. MVVM előnyei

Az MVVM-et úgy terveztek, hogy a WPF adatkapcsolati funkciókat használja a nézetfejlesztés jobb elkülönítésére a séma többi részétől. Hatékonyan eltávolítja a GUI kódot a nézetrétegről. Közvetlen kódolás helyett leíró nyelvet használhatnak, és adatkapcsolatokat hozhatnak létre az alkalmazásfejlesztők által létrehozott és karbantartott nézetmodellekkel. A szerepek szétválasztása miatt a nézetalkotóknak nem kell üzleti logikával foglalkozniuk. A rétegeket külön emberek vagy csapatok alakíthatják ki, így jobban betarthatók a határidők. Még a fejlesztők is használhatják ezt a mintát a dolgok felgyorsítására, mivel a legtöbb változtatási igény a nézetet érinti, nem a logikát, ezért kevésbé kell hozzájárni. [11]

Ez a minta megpróbálja ötvözni az MVC és az adat-összerendelés legjavát, miközben a keretrendszer a lehető legtisztábban tartja az adat-összerendeléssel. Érvényesítse a bejövő adatokat csatlakozási szoftverek, nézetmodellek és üzleti rétegbeli adatellenőrzés segítségével. Így a modell és a keretrendszer végzi el a munka nagy részét, kiküszöbölvé vagy minimalizálva a nézetekkel közvetlenül foglalkozó alkalmazáslogikát. [11]

## 5.2. Adatbázis

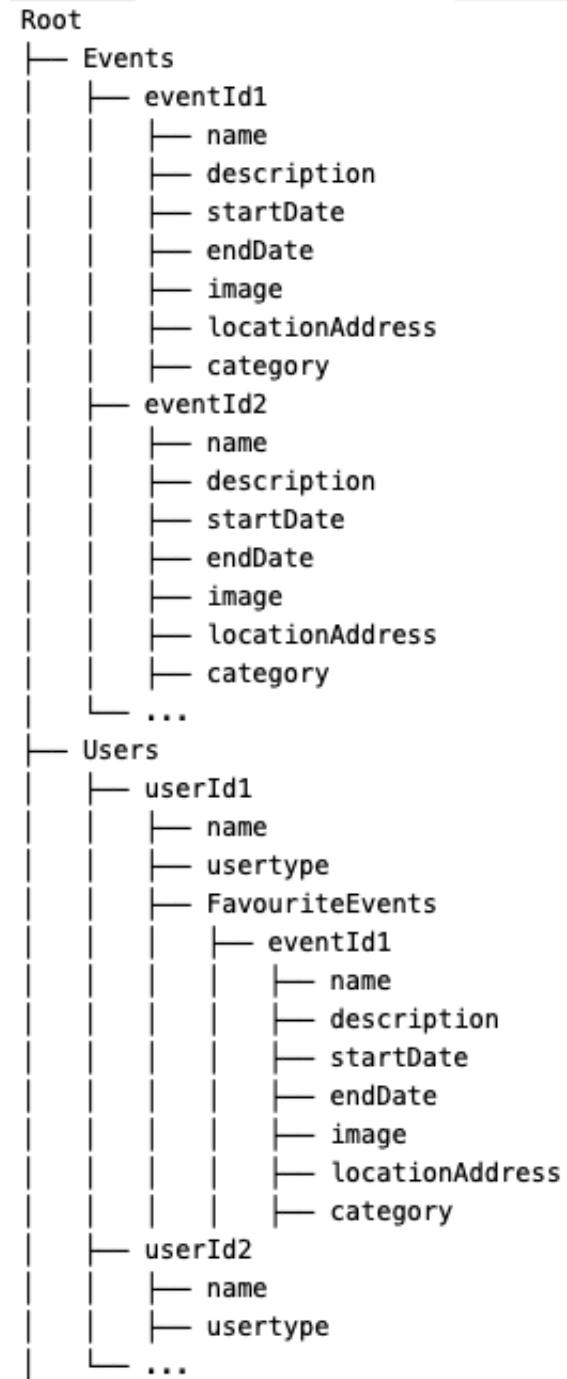
### 5.2.1. Firebase Realtime Database

A Firebase Realtime Database egy felhőben tárolt NoSQL-adatbázis, amelyet a Google Firebase platformja biztosít. Úgy terveztek, hogy valós időben tárolja és szinkronizálja az adatokat az ügyfelek és a kiszolgálók között. A valós idejű adatbázis lehetővé teszi a fejlesztők számára, hogy valós idejű alkalmazásokat készítsenek, amelyek azonnal frissíthetik az adatokat több eszközön és felhasználón.

A Firebase Realtime Database olyan alkalmazásokhoz alkalmas, amelyek valós idejű frissítéseket és együttműködési funkciókat igényelnek, például csevegőalkalmazásokhoz, együttműködő szerkesztőeszközökhez, valós idejű irányítópultokhoz és többszereplős játékokhoz. Leegyszerűsíti a valós idejű funkciók kiépítésének folyamatát, és könnyen használható felületet biztosít a fejlesztők számára az adatszinkronizálás kezeléséhez. [5]

### 5.2.2. Adatbázis szerkezete

A Firebase Realtime Database szerkezete JSON-fán alapul. Ez egy NoSQL-adatbázis, ami azt jelenti, hogy nem követ merev táblázatos struktúrát, mint a hagyományos SQL-adatbázisok. Ehelyett rugalmas, hierarchikus formátumban tárolja az adatokat. [5]



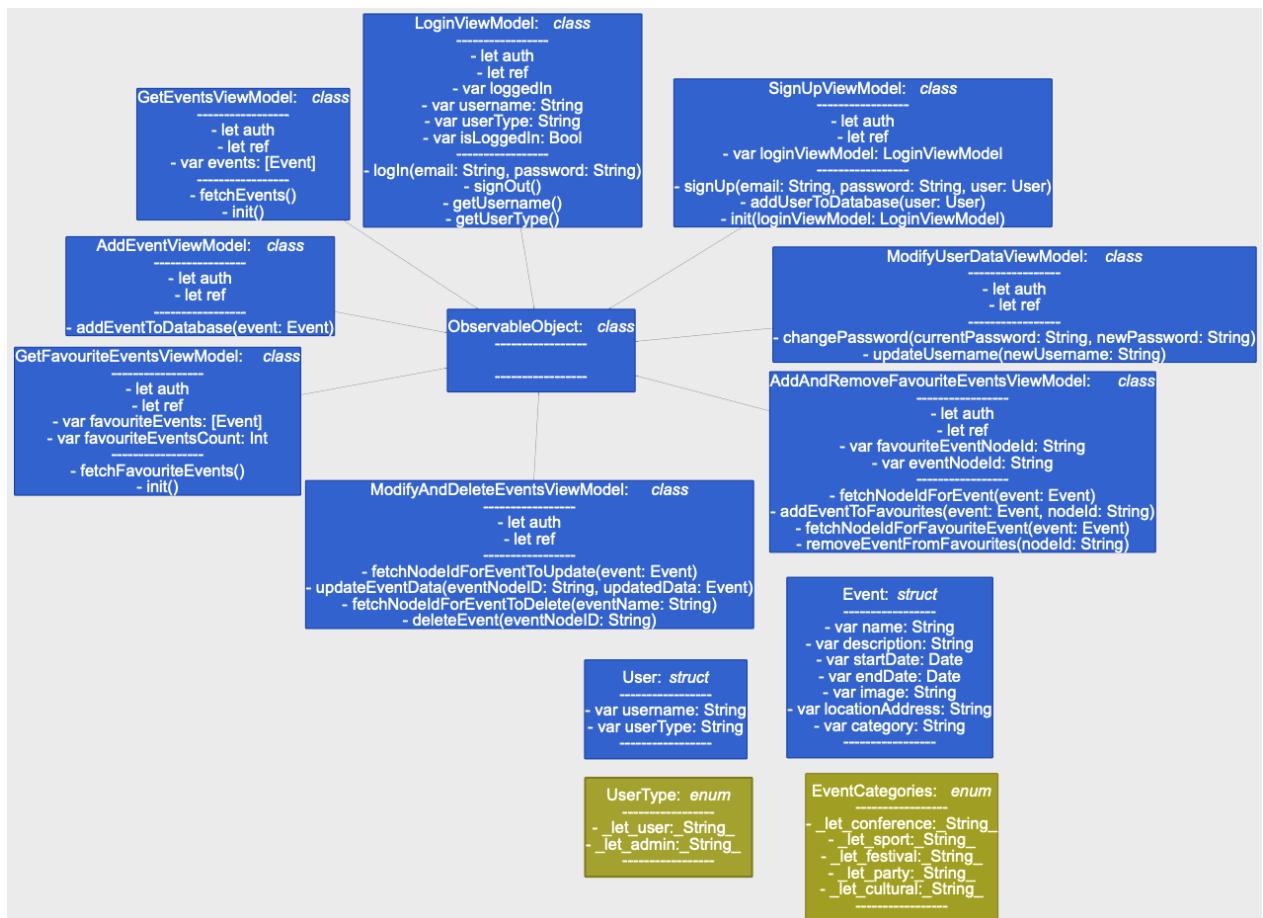
**5.2. ábra.** Adatbázis fa diagramja

Ezt a szerkezetet a [5.2 ábra](#) is jól mutassa. Ezen a fa diagramon az adatbázis struktúrája látható, amely két fő entitásból áll: "Events" és "Users". Az "Events" entitás alatt több eseményobjektum található, amelyek mindenek között az "eventId" kulcsot követően jelennek meg. minden esemény objektumnak több tulajdonsága van, mint a "name", "description", "startDate", "endDate", "image", "locationAddress" és "category". Ezek a tulajdonságok részleteket tartalmaznak az egyes eseményekről. A "Users" entitás alatt több felhasználói objektum található, amelyek mindenek között az "userId" kulcsot követően jelennek meg. minden felhasználónak több tulajdonsága van, mint a "name" és a "usertype". A diagramban elhagyott részletekkel (elszakított pontokkal) szemléltetik, hogy a struktúra további részeit is tartalmaz.

használói objektum rendelkezik olyan tulajdonságokkal, mint a "name" és a "usertype". Ezenkívül minden felhasználói objektum tartalmaz egy beágyazott "FavouriteEvents" entitást. Ez a beágyazott entitás az egyes felhasználók kedvenc eseményeit jelöli. A "FavouriteEvents" entitáson belül több eseményobjektum található (amelyekre az "eventId" hivatkozik), és mindegyik eseményobjektum ugyanazokat a tulajdonságokat tartalmazza, mint az "Events" entitásban. Ez a struktúra lehetővé teszi a felhasználók számára, hogy kedvenc eseményeik gyűjteményét társítsák a fiókjukhoz.

### 5.3. Osztályok

Az osztálydiagramok az objektumorientált modellezés fő építőkövei. Alkalmazásszerkezet általános fogalmi modellezésére, valamint részletes modellezésére, majd ezeknek a modelleknek programozási kóddá való konvertálására használják. Az osztálydiagramok adatmodellezéshez is használhatók. Az osztálydiagram osztályai az alkalmazás fő elemeit, interakcióit és a programhoz szükséges osztályokat képviselik. [13]



### 5.3. ábra. Osztály diagram

A projekt osztály struktúrája a 5.3 ábrán látható. Az MVVM architektúra alapján minden view model elem egy osztály és minden ObservableObject osztályból származik. Mindegyik osztály különböző műveleteket hajt végre:

- **LoginViewModel** felelős a bejelentkezésért és a kijelentkezésért
- **SignUpViewModel** felelős a regisztrálásért
- **ModifyUserDataViewModel** felelős a fealhasználó adatainak a módosításáért
- **GetEventsViewModel** felelős az események lekéréséért
- **AddEventViewModel** felelős egy új esemény hozzáadásáért (csak admin felhasználó)
- **ModifyAndDeleteEventsViewModel** felelős egy esemány információinak a módosításáért, illetve az adott esemény törléséért. (csak admin felhasználó)
- **GetFavouriteEventsViewModel** felelős a kedvenc események lekéréséért
- **AddAndRemoveFavouriteEventsViewModel** felelős egy adott esemény hozzáadásáért a kedvenc eseményekhez, illetve a törléséért egy adott eseménynek a kedvencek közül

## 6. fejezet

# Felhasználói kézikönyv

### 6.1. Splash képernyő

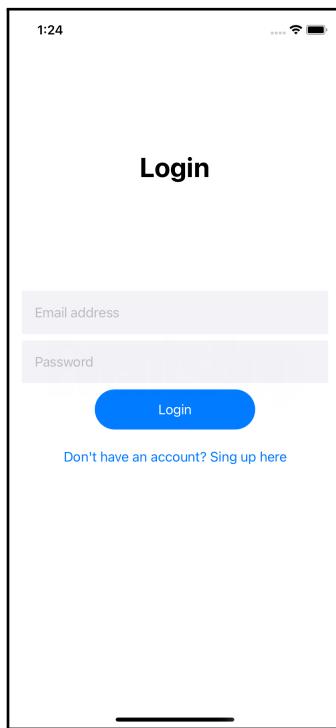
A felhasználó amikor elindítja az alkalmazást, a splash képernyő jelenik meg számára először. Ez a képernyő tartalmazza az applikáció logóját, ami az app nevéből áll. A splash képernyő az alkalmazás elindítása után annyi ideig marad látható, ameddig szükséges, hogy betöltsön az alkalmazás.



6.1. ábra. Splash képernyő

## 6.2. Bejelentkezés

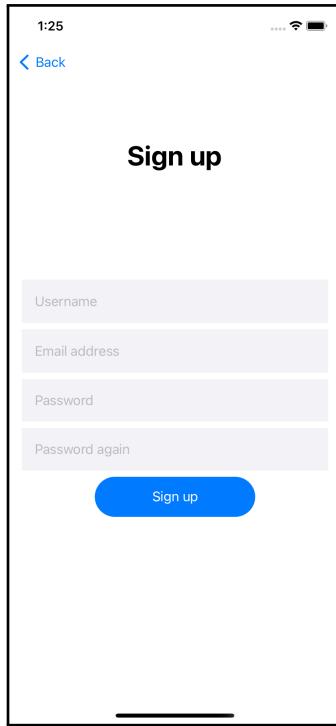
Az splash képernyő lejárta után a bejelentkezés oldal jelenik meg, abban az esetben, hogyha telepítés után indult vagy senki sincsen bejelentkezve. A bejelentkezést úgy lehet megtenni, hogy a felhasználó megadja a fiókjának az email címét és jelszavát, majd megnyomja a "**Login**" gombot. Ha minden adat helyes, akkor átkerül a főoldalra. Ha a felhasználó nem rendelkezik fiókkal, akkor át tud navigálni a regisztrálás oldalra, megnyomva a "Login" alatti "**Don't have an account? Sign up here**" alatti gombot.



6.2. ábra. Bejelentkezés képernyő

## 6.3. Regisztrálás

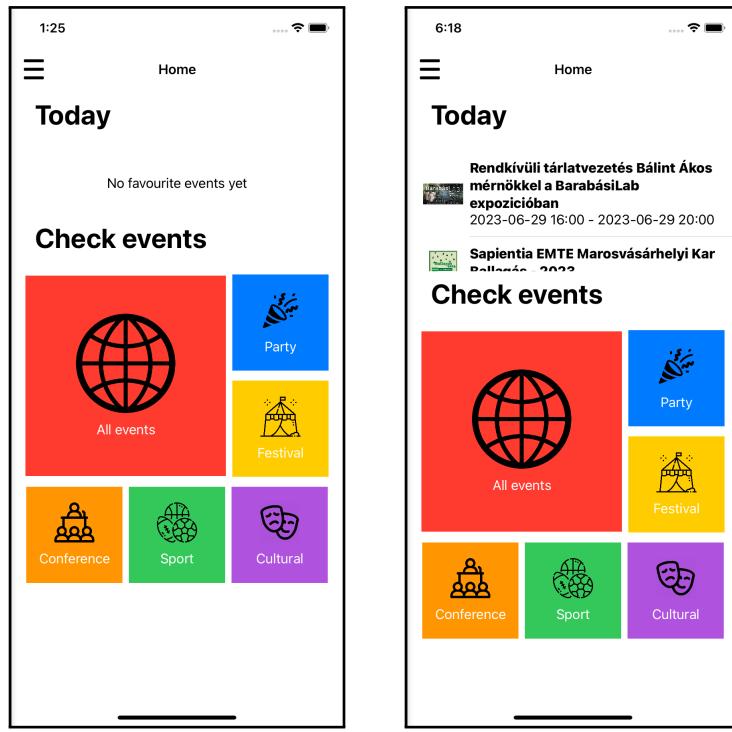
A bejelentkezés oldalról átnavigálva erre, a felhasználó megteheti, hogy létrehozzon egy fiókot, ha még nem rendelkezik vele. Itt szükséges, hogy megadjon egy felhasználó nevet, email címet, jelszót. Ha minden mezőt kitöltött helyesen, akkor a "**Sign up**" gombot megnyomva megtörténik a regisztrálás, illetve az automatikus bejelentkezés.



**6.3. ábra.** Regisztrálás képernyő

## 6.4. Főoldal

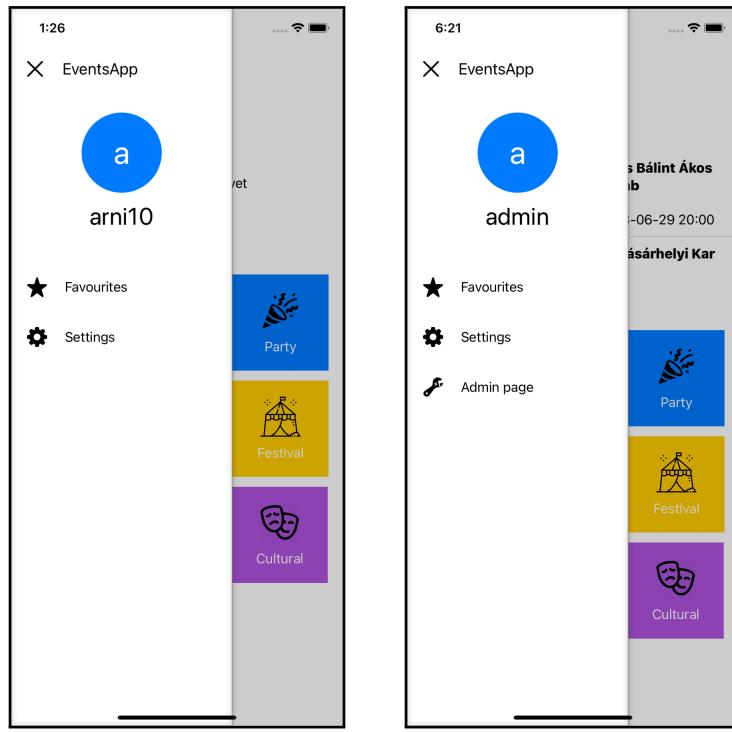
Miután megtörtént a bejelentkezés vagy a felhasználó úgy indította el az alkalmazást, hogy bejelentkezve maradt, akkor a főoldal fog megjelenni. Ezen az oldalon jelennek meg azok a kedvenc események amelyek időtartama tartalmazza az aktuális napot, ha ez üres, akkor az a szöveg jelenik meg, hogy "**No favourite events yet**". Itt még megjelenik egy olyan nézet, ami megengedi, hogy tovább navigálunk az összes eseményre vagy olyan esemény lista oldalra, ahol az események szűrve vannak a kategóriájuk alapján. A bal felső sarokban megjelenik egy gomb, amelyet ha a felhasználó megnyom akkor megnyílik az oldalsó menü.



**6.4. ábra.** Főoldal képernyő kedvenc aznapi eseménnyel és anélkül

## 6.5. Oldalsó menü

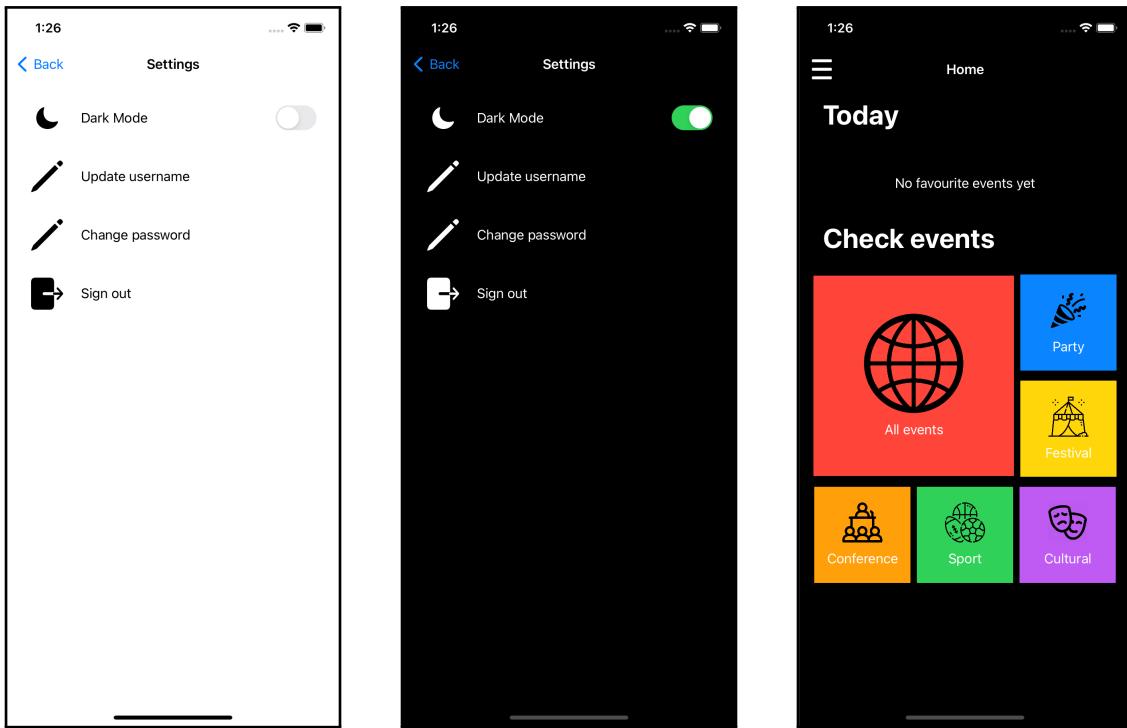
Ha a felhasználó megnyomta a bal felső sarokban lévő gombot, akkor jelenik meg az oldalsó menü. Itt megjelenik a megnyitás gomb helyett egy bezárás gomb, a felhasználó neve, illetve lehetőség adódik a felhasználó számára, hogy megtekintse az általa lementett kedvenc eseményeket megnyomva a "**Favourites**" gombot és a beállítások oldalt megnyomva a "**Settings**" gombot. Amennyiben egy admin felhasználó van bejelentkezve, az illetőnek megjelenik egy plusz oldal amire navigálni tud az "**Admin page**" gomb megnyomásával. Ezen az oldalon olyan műveleteket lehet végrehajtani amikre csak egy admin felhasználó képes.



6.5. ábra. Oldalsó menü

## 6.6. Beállítások

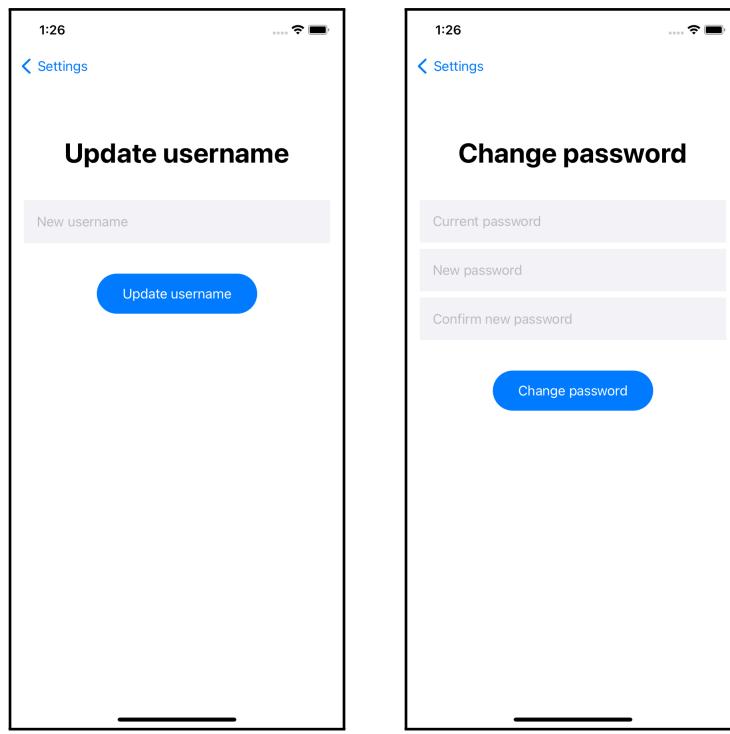
A beállítások oldalon a felhasználónak lehetősége adódik arra, hogy megváltoztassa az alkalmazás témáját a "**Dark mode**" kapcsolót használva, a felhasználó nevét a "**Update username**" gombot megnyomva, a jelszavát a "**Change password**" gombot megnyomva, illetve arra, hogy kijelentkezzen a "**Sign out**" gombot megnyomva.



**6.6. ábra.** Beállítások oldal világos és sötét témaival, illetve a főoldal sötét témaival

### 6.6.1. Felhasználó adatainak a módosítása

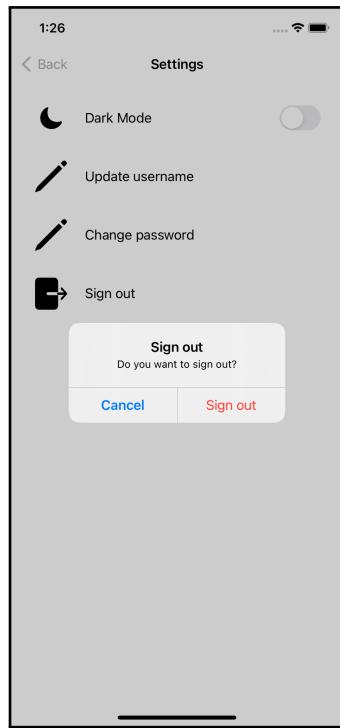
Ha a felhasználó módosítani szeretné a felhasználó nevét vagy a jelszavát akkor azokat megteheti az erre szánt oldalakon. Itt meg kell adja az új nevet és jelszót, majd meg kell nyomja a szövegmező alatti gombot, hogy a művelet végrehajtódjon.



6.7. ábra. A felhasználó név és jelszó módosítása oldalak

### 6.6.2. Kijelentkezés

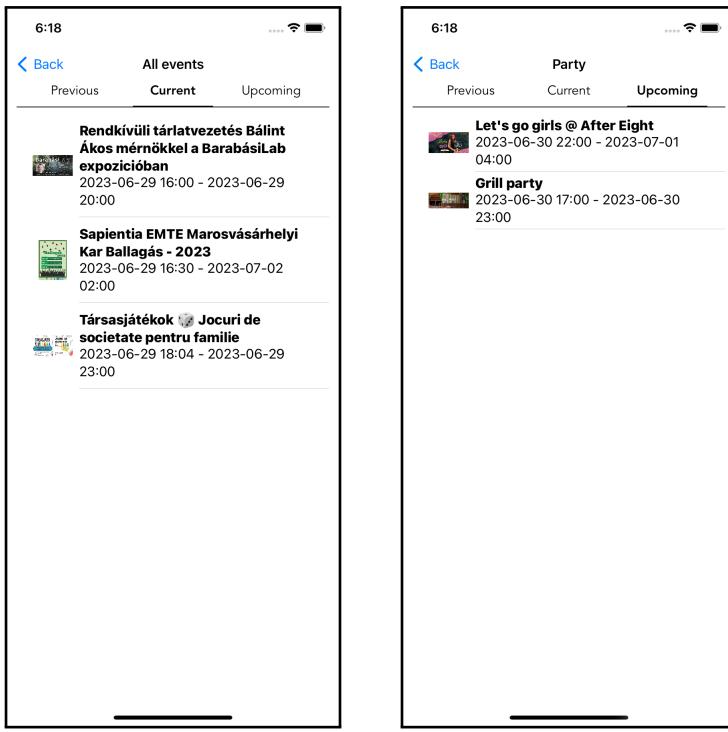
Ha a felhasználó ki szeretne jelentkezni a fiókjából, akkor azt úgy teheti meg, hogy ha megnyomja a "**Sign out**" gombot, majd megjelenik előtte egy kis ablak ami alapján megerősítheti a döntést "**Sign out**" gombot nyomva vagy visszavonhassa a "**Cancel**" gombbal. Ennek a műveletnek a hatására, a bejelentkezés oldalra való navigálás fog meg-történni.



**6.8. ábra.** Kijelentkezés ablak

## 6.7. Események listája

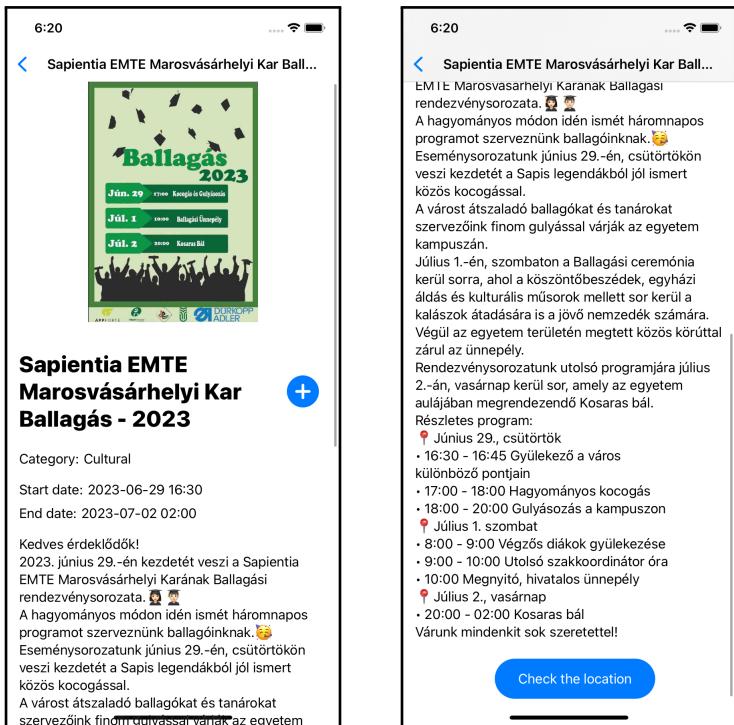
A felhasználó a főoldalon választhat, hogy megszeretné tekinteni az összes eseményt vagy csak egy típusú eseményeket. Az 6.9 ábra első képernyőjén az összes jelenleg zajló esemény látható, míg a másodikon pedig a jövőben zajló buli események láthatóak.



**6.9. ábra.** Összes jelenleg zajló esemény és jövőben zajló party események

## 6.8. Esemény részletei és hozzáadás a kedvencekhez

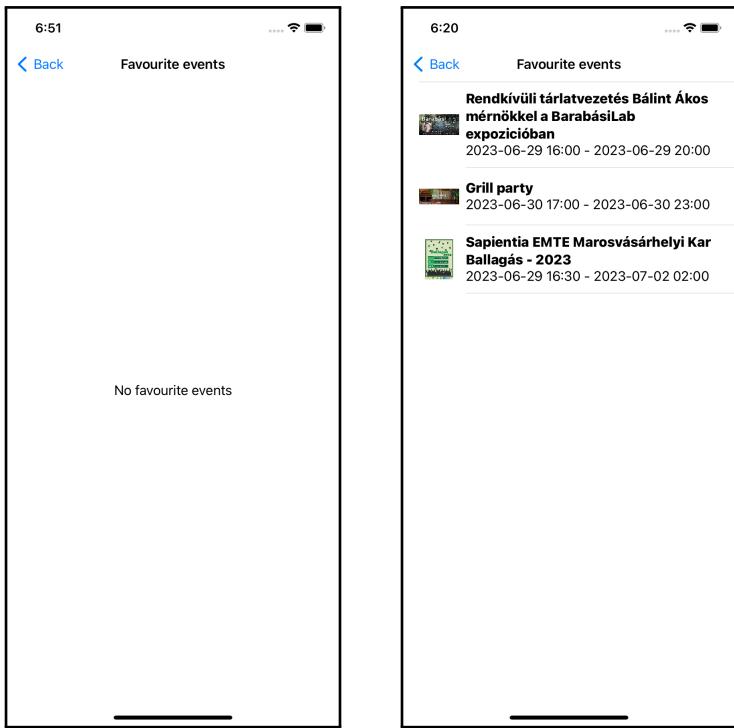
Miután a felhasználó kiválasztott egy eseményt, megtekintheti annak a részleteit, ha megnyomja maga az eseményt. Ezen a képernyőn minden információ megtekinthető az eseménnyel kapcsolatban: képe, neve, kategóriája, kezdeti és végzeti ideje, leírása, illetve a helyszínt, amelyet a "**Check the location**" gomb megnyomásával lehet, ez majd megnyissa a helyszínt az Apple térkép segítségével. Ha a felhasználó el szeretné menteni az adott eseményt a kedvencek közé, akkor az esemény neve mellett " +" gomb segítségével teheti ezt meg.



6.10. ábra. Esemény részletei

## 6.9. Kedvenc események megtekintése

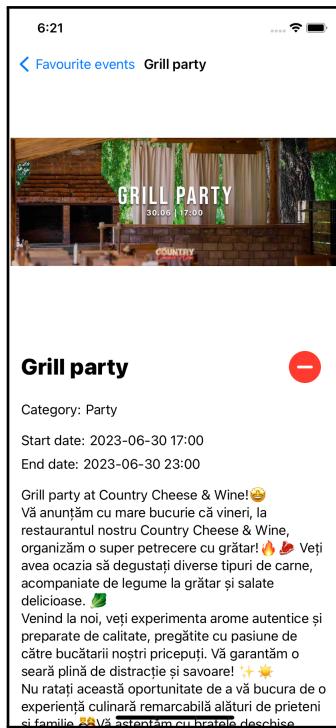
A felhasználó ha lementett egy eseményt vagy többet a kedvencek közé, akkor ezt/ezeket megtekintheti a kedvenc események lista oldalon. Hogy arra az oldalra navigáljon, vissza kell térjen a főoldalra, nyissa meg az oldalsó menüt, nyomja meg a "Favourites" gombot és már is megjelenik az oldal előtte.



6.11. ábra. Üres és nem üres kedvenc események lista

## 6.10. Kedvenc esemény részletei és törlés a kedvencek közül

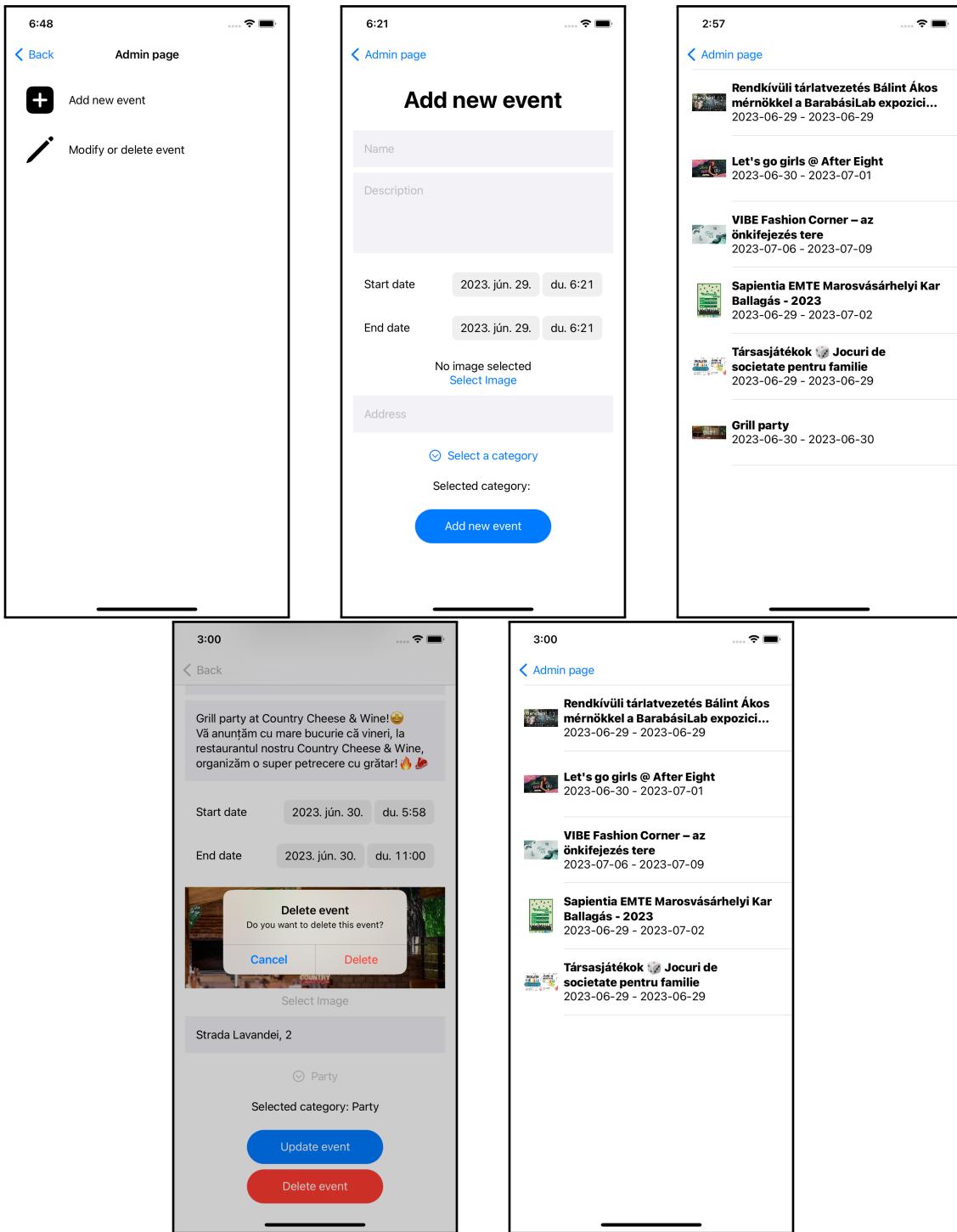
Egy adott kedvenc esemény részletei megtekinthetők, ha a felhasználó megnyomja a képernyőn a megnézni kivánt eseményt a listában. Ezen a képernyőn ugyan úgy látható az összes információ az adott eseményről. Ha a felhasználó törölni szeretné az adott eseményt a kedvencek közül, akkor azt megteheti az esemény neve melletti "-" gombbal, majd ezek után visszakerül a kedvenc események lista oldalhoz.



6.12. ábra. Kedvenc esemény részletei

## 6.11. Admin oldal

Amennyiben egy admin felhasználó van bejelentkezve, az illetőnek megjelenik egy plusz oldal amire navigálni tud az "**Admin page**" gomb megnyomásával az oldalsó menüben. Itt lehetőség adódik számára, hogy új eseményt hozzon létre, illetve, hogy már meglévő esemény információin módosítson és törölje azt.



**6.13. ábra.** Admin oldal és a hozzá tartozó extra műveletek

## 7. fejezet

# Megvalósítás

Ebben a fejezetben bemutatóra kerül a projekt megvalósításának a folyamata. Az alkalmazás Xcode keretrendszerben lett megvalósítva, ami sok segítséget, könnyebbsegést és rugalmasságot biztosított a fejlesztés során. Az alkalmazás elkészítése során a verziókezelésre Git, GitHub, illetve Sourcetree volt használva, hogy az elkészítés folyamata visszakövethető legyen, könnyen lehessen visszaállítani az alkalmazást egy korábbi működő verziójára, ha valami a fejlesztés során elromlott, de legeslegfontosabb indok a használatra az a biztonságos tárolás, hogy bármilyen probléma, hiba esetén a kód megmaradjon és ne vesszen el és a vele együtt befektetett idő és munka. A továbbiakban bemutatóra kerül az egyes funkcionálitások implementálása, a megvalósítások menete, illetve a verziózás.

### 7.1. Bejelentkezés

Az alkalmazás felhasználó kezelése során, a Firebase Authentication beépített módsusa van használva, pontosabb az email és jelszóval való bejelentkezés. A Firebase Authentication átfogó funkciókészletet biztosít a felhasználói hitelesítéshez, beleértve a bejelentkezési és regisztrációs funkciókat is. [4] Az 7.1 ábrán látható, hogy miután ez megkapta az email címet és jelszót, azokkal bejelentkezteti a felhasználót a fiókjába és átállítja a bejelentkezve státuszt **true**-ra, ha nem történt semmi probléma közben.

```
func logIn(email: String, password: String) {
    auth.signIn(withEmail: email, password: password) { [weak self] result, error in
        guard result != nil, error == nil else {
            return
        }

        DispatchQueue.main.async {
            self?.loggedIn = true
        }
    }
}
```

7.1. ábra. Bejelentkezés megvalósítása

## 7.2. Regisztrálás

A felhasználó bérregisztrálása is a Firebase Authentication beépített metódusával történik. Az 7.2 ábrán látható, hogy miután az email cím és jelszó alapján bérregisztráltatta a felhasználót, egyből be is jelentkezteti és feltölti a felhasználóról való információkat az adatbázisba.

```
func signUp(email: String, password: String, user: User) {
    auth.createUser(withEmail: email, password: password) { [weak self] result, error in
        guard result != nil, error == nil else {
            return
        }

        DispatchQueue.main.async {
            self?.loginViewModel.loggedIn = true
            self?.addUserToDatabase(user: user)
        }
    }
}

func addUserToDatabase(user: User) {
    do {
        let jsonEncoder = JSONEncoder()
        let jsonData = try jsonEncoder.encode(user)
        let jsonObject = try JSONSerialization.jsonObject(with: jsonData, options: []) as? [String: Any]
        ref.child("Users").child(auth.currentUser!.uid).setValue(jsonObject)
    } catch {
        print("Error occurred")
    }
}
```

7.2. ábra. Regisztrálás megvalósítása

## 7.3. Kijelentkezés

A kijelentkezés folyamata is a Firebase Authentication beépített metódusával történik. Az 7.3 ábrán látható, hogy miután megtörtént a kijelentkezés, a bejelentkezési státusz átkerül **false**-ra, ami jelzi, hogy jelenleg nincsen senki sem bejelentkezve.

```
func signOut() {
    try? auth.signOut()

    self.loggedIn = false
}
```

7.3. ábra. Kijelentkezés megvalósítása

## 7.4. Események lekérése

Az események lekérésnek a folyamata látható az 7.4 ábrán. Itt látható ahogyan a lekérés folyamatában hozzáadja az eseményeket az **events** nevű tömbhöz, majd ebből lesznek betöltve a képernyőn lista formájában.

```
func fetchEvents() {
    ref.observe(.value) { snapshot in
        var events: [Event] = []

        for child in snapshot.children {
            if let snapshot = child as? DataSnapshot,
                let event = snapshot.value as? [String: Any] {
                if let eventData = try? JSONSerialization.data(withJSONObject: event),
                    let decodedEvent = try? JSONDecoder().decode(Event.self, from: eventData) {
                    events.append(decodedEvent)
                }
            }
        }

        DispatchQueue.main.async {
            self.events = events
        }
    }
}
```

7.4. ábra. Események lekérésének a megvalósítása

## 7.5. Esemény hozzáadása a kedvencekhez

Az 7.5 ábra szemlélteti egy adott esemény hozzáadását a kedvencekhez. Miután az esemény neve alapján megkeresi az esemény csomópontjának az azonosítóját, azzal az azonosítóval hozzáadja az adott eseményt a felhasználónak a kedvenc eseményeihez.

```
func fetchNodeIdForEvent(event: Event) {
    let query = ref.child("Events").queryOrdered(byChild: "name").queryEqual(toValue: event.name)

    query.observeSingleEvent(of: .value) { snapshot in
        guard let node = snapshot.children.allObjects.first as? DataSnapshot else {
            print("Data not found")
            return
        }

        self.addEventToFavourites(event: event, nodeId: node.key)
    }
}

func addEventToFavourites(event: Event, nodeId: String) {
    do {
        let jsonEncoder = JSONEncoder()
        let jsonData = try jsonEncoder.encode(event)
        let jsonObject = try JSONSerialization.jsonObject(with: jsonData, options: []) as? [String: Any]
        ref.child("Users").child(auth.currentUser?.uid ?? "Undefined").child("FavouriteEvents").child(nodeId).setValue(jsonObject)
    } catch {
        print("Error occurred")
    }
}
```

7.5. ábra. Események kedvencekhez hozzáadásának a megvalósítása

## 7.6. Kedvenc események lekérése

A kedvenc események lekérésének a folyamata nagyon hasonló az események lekérésének a folyamatához. Amint az 7.6 ábra is mutassa, megkeresi a bejelentkezett felhasználó kedvenc eseményeit, majd azokat eltárolja a **favouriteEvents** tömbben és abból fogja megjeleníteni őket a képernyőn. Eltárolásra kerül a kedvenc események száma is, hogy ha ez üres, akkor ne egy üres lista jelenjen meg, hanem egy szöveg ami jelzi ezt.

```
func fetchFavouriteEvents() {
    ref.child("Users").child(auth.currentUser?.uid ?? "Undefined").child("FavouriteEvents").observe(.value) { snapshot in
        var favouriteEvents: [Event] = []

        for child in snapshot.children {
            if let snapshot = child as? DataSnapshot,
                let event = snapshot.value as? [String: Any] {
                if let eventData = try? JSONSerialization.data(withJSONObject: event),
                    let decodedEvent = try? JSONDecoder().decode(Event.self, from: eventData) {
                    favouriteEvents.append(decodedEvent)
                }
            }
        }

        DispatchQueue.main.async {
            self.favouriteEvents = favouriteEvents
            self.favouriteEventsCount = favouriteEvents.count
        }
    }
}
```

7.6. ábra. Kedvenc események lekérésének a megvalósítása

## 7.7. Esemény törlése a kedvencek közül

Egy adott eseménynek a törlése a kedvencek közül 2 részre van felosztva. Az 7.6 ábrán látható ez jól, hogy első sorban megkeresi az esemény csomópontjának az azonosítóját, majd annak a segítségével kitörli az azt az eseményt az aktuális felhasználó kedvencei közül.

```
func fetchNodeIdForFavouriteEvent(event: Event) {
    let query = ref.child("Users").child(auth.currentUser?.uid ?? "Undefined").child("FavouriteEvents").queryOrdered(byChild:
        "name").queryEqual(toValue: event.name)

    query.observeSingleEvent(of: .value) { snapshot in
        guard let node = snapshot.children.allObjects.first as? DataSnapshot else {
            print("Data not found")
            return
        }

        self.removeEventFromFavourites(nodeId: node.key)
    }
}

func removeEventFromFavourites(nodeId: String) {
    let nodeRef = ref.child("Users").child(auth.currentUser?.uid ?? "Undefined").child("FavouriteEvents").child(nodeId)

    nodeRef.removeValue { error, _ in
        if let error = error {
            print("Failed to remove data: \(error)")
        } else {
            print("Data removed successfully")
        }
    }
}
```

7.7. ábra. Események kedvencek közül való törlésének a megvalósítása

## 7.8. Felhasználó fiók adatainak a megváltoztatása

A felhasználónak lehetősége adódik, hogy változtasson a fiókjának pár adatán, pontosabban a felhasználó nevén és jelszaván.

### 7.8.1. Felhasználó név megváltoztatása

Ezt a folyamatot az 7.8 ábra illusztrálja. Itt látható, hogy megkeresi a bejelentkezett felhasználó nevét, majd kicseréli az újra.

```
func updateUsername(newUsername: String) {
    ref.child("Users").child(auth.currentUser?.uid ?? "Undefined").updateChildValues(["username": newUsername]) { error, _ in
        if let error = error {
            print("Failed to update username: \(error)")
        } else {
            print("Username updated successfully")
        }
    }
}
```

7.8. ábra. Felhasználó név megváltoztatásának a megvalósítása

### 7.8.2. Felhasználó jelszavának megváltoztatása

Felhasználó jelszavának megváltoztatása 2 részből tevődik össze. A 7.9 ábra prezentálja ennek a megvalósítását, hogy újra hitelesíti a felhasználót az email címe és aktuális jelszava alapján, majd frissíti a jelszót az újra.

```
func changePassword(currentPassword: String, newPassword: String) {
    guard let currentUser = Auth.auth().currentUser else {
        return
    }

    let credential = EmailAuthProvider.credential(withEmail: currentUser.email!, password: currentPassword)

    currentUser.reauthenticate(with: credential) { _, error in
        if error != nil {
            print("Failed to reauthenticate!")
        } else {
            currentUser.updatePassword(to: newPassword) { error in
                if error != nil {
                    print("Failed to update password!")
                }
            }
        }
    }
}
```

7.9. ábra. Felhasználó jelszavának megváltoztatásának a megvalósítása

## 7.9. Új esemény hozzáadása (admin felhasználó)

Egy új esemény hozzáadásának megvalósítását az 7.10 ábra írja le. Látható, hogy az újonnan létrehozott eseményt hozzáadja az adatbázishoz.

```

func addEventToDatabase(event: Event) {
    do {
        let jsonEncoder = JSONEncoder()
        let jsonData = try jsonEncoder.encode(event)
        let jsonObject = try JSONSerialization.jsonObject(with: jsonData, options: []) as? [String: Any]
        ref.child("Events").childByAutoId().setValue(jsonObject)
    } catch {
        print("Error occurred")
    }
}

```

**7.10. ábra.** Új esemény hozzáadásának a megvalósítása

## 7.10. Esemény módosítása/törlése (admin felhasználó)

Továbbá lehetőség adódik egy admin felhasználónak, hogy módosítson egy esemény adatain, illetve, hogy kitöröljön eseményt az adatbázisból.

### 7.10.1. Esemény módosítása

Egy esemény információinak a megmódosítása 2 részből áll, ahogyan ezt az 7.11 ábra is ismerteti. Első sorban megkeresi a módosítani kívánt esemény csomópontjának az azonosítóját, majd az alapján megkeresi az eseményt amelyiken módosítani szeretnénk és frissíti azt az új adatokkal.

```

func fetchNodeIdForEventToUpdate(event: Event) {
    let query = ref.child("Events").queryOrdered(byChild: "name").queryEqual(toValue: event.name)

    query.observeSingleEvent(of: .value) { snapshot in
        guard let node = snapshot.children.allObjects.first as? DataSnapshot else {
            print("Data not found")
            return
        }

        self.updateEventData(eventNameID: node.key, updatedData: event)
    }
}

func updateEventData(eventNameID: String, updatedData: Event) {
    let encoder = JSONEncoder()
    guard let newDataDict = try? JSONSerialization.jsonObject(with: encoder.encode(updatedData), options: []) as? [String: Any] else {
        return
    }

    let eventsRef = ref.child("Events").child(eventNameID)
    eventsRef.updateChildValues(newDataDict)

    let usersRef = ref.child("Users")
    usersRef.observeSingleEvent(of: .value) { snapshot in
        guard let userData = snapshot.value as? [String: Any] else { return }

        for (userID, userData) in userData {
            guard var userDict = userData as? [String: Any],
                  var favoriteEvents = userDict["FavouriteEvents"] as? [String: [String: Any]],
                  let favoriteEvent = favoriteEvents[eventNameID]
            else { continue }

            favoriteEvents[eventNameID] = favoriteEvent.merging(newDataDict) { (_, new) in new }
            userDict["FavouriteEvents"] = favoriteEvents

            let userRef = usersRef.child(userID)
            userRef.updateChildValues(userDict)
        }
    }
}

```

**7.11. ábra.** Esemény módosításának a megvalósítása

## 7.10.2. Esemény törlése

Folyamat ami során egy eseményt törölni lehet, 2 részre oszlik le, ahogyan ezt az [7.12](#) ábra is vázolja. Első lépésnek megkeresi a törölni kivánt esemény csomópontjának az azonosítóját, majd azt felhasználva kitörli azt az eseményt az adatbázisból.

```
func fetchNodeIdForEventToDelete(eventName: String) {
    let query = ref.child("Events").queryOrdered(byChild: "name").queryEqual(toValue: eventName)

    query.observeSingleEvent(of: .value) { snapshot in
        guard let node = snapshot.children.allObjects.first as? DataSnapshot else {
            print("Data not found")
            return
        }

        self.deleteEvent(eventNodeID: node.key)
    }
}

func deleteEvent(eventNodeID: String) {
    let eventsRef = ref.child("Events").child(eventNodeID)
    eventsRef.removeValue()

    let usersRef = ref.child("Users")
    usersRef.observeSingleEvent(of: .value) { snapshot in
        guard let userData = snapshot.value as? [String: Any] else { return }

        for (userID, userData) in userData {
            guard var userDict = userData as? [String: Any],
                  var favoriteEvents = userDict["FavouriteEvents"] as? [String: Any]
                  else { continue }

            if favoriteEvents[eventNodeID] != nil {
                favoriteEvents.removeValue(forKey: eventNodeID)
                userDict["FavouriteEvents"] = favoriteEvents

                let userRef = usersRef.child(userID)
                userRef.updateChildValues(userDict)
            }
        }
    }
}
```

**7.12. ábra.** Esemény törlésének a megvalósítása

## 7.11. Verziókezelés

A projekt elkészítése során a verziókezelésre GitHub, illetve Sourcetree lett használva. A feltöltött változatok megtekinthetők az [7.13](#) ábrán.

Graph	Commit	Description	Author	Date
o	dbc2813	Delete user adding from admin page, some refactoring	Pall Arnold-Barna...	Yesterday at 20:16
	fa3705e	Modify and delete event added, some refactoring	Pall Arnold-Barna <...	Yesterday at 15:54
	0e4cdb6	Move dark mode switch from side menu to settings page	Pall Arnold-Barna <...	28 Jun 2023 at 12:53
	16a0968	Update username and change password functionalities added	Pall Arnold-Barna <...	27 Jun 2023 at 21:13
	e055780	Add today favourite events to the home screen, some refactoring	Pall Arnold-Barna <...	27 Jun 2023 at 20:12
	5c3b89f	Some refactoring regarding the event addition and removal to/from favourites and some other modifications	Pall Arnold-Barna <...	27 Jun 2023 at 14:10
	510954d	Adding and removing events to/from favourite events, some refactoring	Pall Arnold-Barna <...	26 Jun 2023 at 22:29
	3362497	Show the selected events details on the detail page, open the location of the event via Maps, some refactoring	Pall Arnold-Barna <...	26 Jun 2023 at 17:54
	f1b2e5b	Event filtering bases on the date and the type added, some refactoring	Pall Arnold-Barna <...	26 Jun 2023 at 15:20
	4eb2913	Get to show the events in a list, some refactoring	Pall Arnold-Barna <...	24 Jun 2023 at 21:15
	01842cf	Add new event and user using the admin account, some refactoring	Pall Arnold-Barna <...	23 Jun 2023 at 20:30
	5defaa6	Change the field check from alert to popup view, some refactoring	Pall Arnold-Barna <...	22 Jun 2023 at 22:50
	666fc8e	Add app icon, add launch screen, some refactoring	Pall Arnold-Barna <...	21 Jun 2023 at 19:15
	f264789	Add EventList, some refactoring	Pall Arnold-Barna <...	10 May 2023 at 21:21
	7f2553f	Some refactoring, hide Admin page based on the usertype	Pall Arnold-Barna <...	8 Apr 2023 at 16:52
	1015415	Adding the user to the Database, retrieve data from the Database and some other modifications	Pall Arnold-Barna <...	7 Apr 2023 at 19:32
	19cedda	Update .gitignore file	Pall Arnold-Barna <...	5 Apr 2023 at 21:34
	98f6703	Add some views, add navigation links to the side menu, improve the side menu, save the status of dark or light mode	Pall Arnold-Barna <...	21 Mar 2023 at 18:56
	2ad83a3	.gitignore is now working	Pall Arnold-Barna <...	14 Mar 2023 at 19:42
	5b0139a	Initial commit	Pall Arnold-Barna <...	14 Mar 2023 at 19:26
	1ece2ef	Initial commit	Pall Arnold-Barna <...	14 Mar 2023 at 19:00
	e720d04	Initial Commit	Pall Arnold-Barna <...	10 Mar 2023 at 17:52

7.13. ábra. Commit-ok Sourcetree-ből

# Összefoglaló

Összefoglalonak elmondhatom, hogy sikerült egy olyan alkalmazást létrehozni, amely tartalmazza az alap funkcionálisokat, amit egy esemény követő alkalmazásnak tudnia kell, meglehet tekinteni az események listáját, részleteit, amik segíthetnek a részvétel be-tervezésén, hozzá lehet adni a kedvencekhez. Ezt egy korszerű alkalmazás végzi el, ami könnyen használható bármilyen korosztályú felhasználó számára, mivel nagyon átlátható és a használata is egyszerű. Továbbá alkalmazás kinézet testreszabásával is rendelkezik, illetve a felhasználó megtudja módosítani némi adatát. Ezek mellett van admin felhasználó típus is, aki mindegyik hagyományos felhasználó által elvégzett műveleteket végre tudja hajtani és még egy párat extrában, mint például egy új esemény létrehozása.

Az MVVM architekturális mintának köszönhetően az applikáció tovább fejleszthető új funkciókkal anélkül, hogy a már meglévőkön változtatni kellene. Az egyik funkcionálitás amivel bővíteni lehetne az az események megosztása a felhasználók között, hogy ha egy felhasználó talált egy olyan eseményt amiben ő és egy másik felhasználó is érdekelt lenne, akkor azt tudja megosztani az illetővel. Egy másik funkcionálitás, hogy lehessen értesítéseket kapni egy kedvenc esemény kezdetéről, meghatározva, hogy a kezdeti idő előtt mennyivel értesítsen. Továbbá olyan extra funkcionálitás lehetne, hogy egy kedvenc eseményt lementsünk az okostelefon naptárjába, hogy ott lehessen megtekinteni az időtartamát.

A projekt megtalálható az alábbi linken: <https://github.com/PallArnoldBarna/EventsApp>

# Köszönetnyilvánítás

Végül köszönetet szeretnék mondani Olteán-Péter Borókának, aki a dolgozat során végig irányított, segített a megvalósításában. Értékes tanácsai és útmutatásai nagyban hozzájárultak a dolgozat sikeréhez.

# Ábrák jegyzéke

3.1. Használt technológiák . . . . .	17
4.1. A hagyományos felhasználó használat eset diagramja . . . . .	20
4.2. Az admin felhasználó használat eset diagramja . . . . .	21
5.1. MVVM architektúra szerkezete . . . . .	23
5.2. Adatbázis fa diagramja . . . . .	26
5.3. Osztály diagram . . . . .	27
6.1. Splash képernyő . . . . .	29
6.2. Bejelentkezés képernyő . . . . .	30
6.3. Regisztrálás képernyő . . . . .	31
6.4. Főoldal képernyő kedvenc aznapi eseménnyel és anélkül . . . . .	32
6.5. Oldalsó menü . . . . .	33
6.6. Beállítások oldal világos és sötét témával, illetve a főoldal sötét témával . . . . .	34
6.7. A felhasználó név és jelszó módosítása oldalak . . . . .	35
6.8. Kijelentkezés ablak . . . . .	36
6.9. Összes jelenleg zajló esemény és jövőben zajló party események . . . . .	37
6.10. Esemény részletei . . . . .	38
6.11. Üres és nem üres kedvenc események lista . . . . .	39
6.12. Kedvenc esemény részletei . . . . .	40
6.13. Admin oldal és a hozzáartozó extra műveletek . . . . .	41
7.1. Bejelentkezés megvalósítása . . . . .	42
7.2. Regisztrálás megvalósítása . . . . .	43
7.3. Kijelentkezés megvalósítása . . . . .	43
7.4. Események lekérésének a megvalósítása . . . . .	44
7.5. Események kedvencekhez hozzáadásának a megvalósítása . . . . .	44
7.6. Kedvenc események lekérésének a megvalósítása . . . . .	45
7.7. Események kedvencek közül való törlésének a megvalósítása . . . . .	45
7.8. Felhasználó név megváltoztatásának a megvalósítása . . . . .	46
7.9. Felhasználó jelszavának megváltoztatásának a megvalósítása . . . . .	46
7.10. Új esemény hozzáadásának a megvalósítása . . . . .	47
7.11. Esemény módosításának a megvalósítása . . . . .	47
7.12. Esemény törlésének a megvalósítása . . . . .	48
7.13. Commit-ok Sourcetree-ből . . . . .	49

# Irodalomjegyzék

- [1] Bandsintown, <https://www.bandsintown.com>.
- [2] Eventbrite, <https://apps.apple.com/ie/app/eventbrite/id487922291>.
- [3] Eventful, <https://eventful-app.com>.
- [4] Firebase Authentication, <https://firebase.google.com/docs/auth>.
- [5] Firebase Realtime Database, <https://firebase.google.com/docs/database>.
- [6] Firebase, <https://firebase.google.com>.
- [7] Git, <https://git-scm.com>.
- [8] Github, <https://docs.github.com/en/get-started/quickstart/hello-world>.
- [9] iOS, <https://en.wikipedia.org/wiki/IOS>.
- [10] Meetup, <https://www.meetup.com>.
- [11] Modell-Nézet-NézetModell architektúra, <https://hu.wikipedia.org/wiki/Modell-nÃ©zet-nÃ©zetmodell>.
- [12] Modell-Nézet-NézetModell VS Modell-Nézet-Vezérlő, <https://www.guru99.com/mvc-vs-mvvm.html>.
- [13] Osztálydiagram, <https://hu.wikipedia.org/wiki/OsztÃ¡lydiagram>.
- [14] Sourcetree, <https://www.sourcetreeapp.com>.
- [15] Swift, <https://developer.apple.com/swift/>.
- [16] SwiftUI, <https://developer.apple.com/xcode/swiftui/>.
- [17] Xcode, <https://developer.apple.com/xcode/>.