

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



**SAPIENTIA  
ERDÉLYI MAGYAR  
TUDOMÁNYEGYETEM**

DineEase: Innovatív asztalfoglalásrendszer éttermek számára

**DIPLOMADOLGOZAT**

Témavezető:

Dr. Márton Gyöngyvér,  
Egyetemi adjunktus

Végzős hallgató:

Suciú Aksza

**2024**

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



**UNIVERSITATEA  
SAPIENTIA**

DineEase: Sistem inovator de rezervare pentru restaurante

**LUCRARE DE DIPLOMĂ**

Coordonator științific:  
Dr. Márton Gyöngyvér,  
Lector universitar

Absolvent:  
Suciu Aksza

**2024**

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



**SAPIENTIA**  
HUNGARIAN UNIVERSITY  
OF TRANSYLVANIA

DineEase: Innovative table reservation system for restaurants

**BACHELOR THESIS**

Scientific advisor:  
Dr. Márton Gyöngyvér,  
Lecturer

Student:  
Suciu Aksza

**2024**

**LUCRARE DE DIPLOMĂ**

Coordonator științific:  
**Lector dr. Márton Gyöngyvér**

Candidat: **Suciuc Aksza**  
Anul absolvirii: **2024**

**a) Tema lucrării de licență:**

DineEase: SISTEM INOVATOR DE REZERVARE PENTRU RESTAURANTE

**b) Problemele principale tratate:**

- Studiul bibliografic al aplicațiilor similare
- Tratarea problemelor de siguranță: autentificare și autorizare
- Prezentarea tehnologiilor utilizate
- Prezentarea specificațiilor sistemului creat: cerințe utilizator, cerințe sistem
- Arhitectura sistemului creat, baza de date, interfețele aplicației, urmărirea versiunilor, proiect management

**c) Desene obligatorii:**

- Schema bloc a aplicației
- Schema bazei de date
- Diagramele use-case

**d) Softuri obligatorii:**

.NET: pentru backend  
MSSQL: pentru baza de date  
Flutter: pentru frontend

**e) Bibliografia recomandată:**

Mark Stamp. *Information security: principles and practice*. John Wiley & Sons, 2011  
Microsoft Corporation. Asp.net core documentation.

<https://learn.microsoft.com/en-us/aspnet/core/?view=aspnetcore-7.0>

Microsoft Corporation. Microsoft sql server documentation.

<https://learn.microsoft.com/en-us/sql/?view=sql-server-ver16>

Google LLC. Flutter documentation. <https://docs.flutter.dev/>

**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

Primit tema la data de: 1 mai 2023

Termen de predare: 24 iunie 2024

Semnătura Director Departament

Semnătura responsabilului  
programului de studiu

Semnătura coordonatorului

Semnătura candidatului

### **Declarație**

Subsemnatul/a .....SUCIU AKSZA....., absolvent(ă) .....al/a  
specializării .....INFORMATICA....., promoția.....2024..  
cunoscând prevederile Legii învățământului superior nr. 199 din 2023 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, CORUNCA  
Data: 13.06.2024.

Absolvent  
Semnătura.....Suciu.....

# Kivonat

Dolgozatom témája egy egységes asztalfoglalórendszer megvalósítása az emberek és éttermek számára. Mint tudjuk, leggyakrabban az asztalfoglalásokat telefonhívással vagy üzenet küldésével intézik, ami problémát jelenthet, olyan esetekben, amikor a telefonszámot nehéz megtalálni vagy kutatni kell az interneten utána, sokszor sikertelenül, és az egész folyamat sok időt vesz igénybe.

A fejlesztett alkalmazás célja, hogy olyan környezetet biztosítson a felhasználók számára, amelyben minden éttermet megtalálhatnak, gyorsan rákereshetnek egy általuk preferált étteremre és egyszerűen foglalhatnak, az éttermek pedig ezeket a foglalásokat könnyedén kezelhetik. Hangsúly van fektetve a felhasználói véleményekre is, amelyek segítségével könnyebben lehet választani az éttermek közül. Ugyanakkor a nagyobb események szervezésre szánt éttermeket sem hagyhattuk ki. Lehetőséget szerettem volna nyújtani nekik, hogy számukra is könnyebben működjön a kommunikáció az érdekelt féllel, így nekik ez az alkalmazás teret biztosít egy személyes meeting programálására az eseményt szervezni vágyó emberekkel. Egy másik kérdés amivel szembenéznek az éttermek, az hogy hogyan és hol reklámozhatják a megrendezésre kerülő eseményeket? Ugyanúgy a felhasználóknak is problémát jelenthet találni egy általuk kedvelt eseményt, ugyanis nem tudják hol kereshetnek rá vagy nem értesülnek róla. Erre is megoldást szerettem volna nyújtani az alkalmazásban, hogy helyet biztosítson egy hirdető felületre is.

A dolgozatom során azt tárgyalom, miként készült el a DineEase alkalmazás és milyen módszerekkel lehet azt egy jól működő és használható alkalmazássá alakítani, amely a felhasználói preferenciáknak eleget tesz. A megvalósított alkalmazás modern megközelítéssel oldja meg a problémát, ezért úgy gondolom, forradalmat jelenthet úgy az éttermek mint a felhasználók életében is, mivel arra szolgál, hogy megkönnyítse mindenkit fél dolgát.

# Rezumat

Tema lucrării mele este realizarea unui sistem unificat de rezervare pentru persoane și restaurante. După cum știm, rezervările de mese sunt cel mai frecvent făcute prin apeluri telefonice sau prin trimiterea de mesaje, ceea ce poate fi problematic în cazurile în care numărul de telefon este greu de găsit sau necesită căutări extensive pe internet, deseori fără succes, ceea ce face ca întregul proces să fie consumator de timp.

Scopul aplicației dezvoltate este de a oferi un mediu în care utilizatorii pot găsi rapid un restaurant preferat și pot face rezervări cu ușurință, în timp ce restaurantele pot gestiona aceste rezervări fără efort. De asemenea, se pune accentul pe păreriile utilizatorilor, care ajută la alegerea mai ușoară între restaurante. În plus, nu am putut trece cu vederea restaurantele dedicate organizării de evenimente mai mari. Am dorit să le ofer o modalitate de a facilita comunicarea cu părțile interesate, oferindu-le astfel o platformă pentru a programa întâlniri personale cu organizatorii de evenimente. O altă problemă cu care se confruntă restaurantele este cum și unde să-și facă publicitate evenimentelor viitoare. Similar, utilizatorii pot găsi dificil să găsească evenimentele care le plac, deoarece s-ar putea să nu stie unde să caute sau să nu fie informați despre acestea. Am dorit să abordez această problemă prin oferirea unei platforme de publicitate în cadrul aplicației.

În lucrarea mea, discut despre cum a fost dezvoltată aplicația DineEase și metodele care o pot transforma într-o aplicație bine funcțională și ușor de utilizat, care să corespundă preferințelor utilizatorilor. Aplicația implementată folosește o abordare modernă pentru a rezolva problema, și cred că poate revoluționa viețile atât ale restaurantelor, cât și ale utilizatorilor, prin simplificarea procesului pentru ambele părți.

# Abstract

The topic of my thesis is the implementation of a unified table reservation system for people and restaurants. As we know, table reservations are most commonly made through phone calls or sending messages, which can be problematic in cases where the phone number is difficult to find or requires extensive internet searching, often unsuccessfully, making the whole process time-consuming.

The goal of the developed application is to provide an environment where users can quickly find a preferred restaurant, and make reservations easily, while restaurants can manage these reservations effortlessly. The emphasis is also placed on user reviews, which help in choosing between restaurants more easily. Additionally, we could not overlook restaurants dedicated to organizing larger events. I aimed to offer them a way to facilitate communication with interested parties, thus providing a platform for them to schedule personal meetings with event organizers. Another issue restaurants face is how and where to advertise their upcoming events. Similarly, users may find it challenging to find events they like, as they might not know where to look or might not be informed about them. I wanted to address this by providing an advertising platform within the application.

In my thesis, I discuss how the DineEase application was developed and the methods that can transform it into a well-functioning and user-friendly application that meets user preferences. The implemented application uses a modern approach to solve the problem, and I believe it can revolutionize the lives of both restaurants and users by making the process easier for both parties.

# Tartalomjegyzék

<b>1. Bevezető</b>	<b>11</b>
<b>2. Hasonló alkalmazások, felhasznált technológiák</b>	<b>13</b>
2.1. Hasonló alkalmazások . . . . .	13
2.2. Felhasznált technológiák . . . . .	14
2.2.1. Flutter . . . . .	14
2.2.2. .NET . . . . .	15
2.2.3. MSSQL . . . . .	16
2.3. Biztonság . . . . .	16
2.3.1. Authentication és authorization . . . . .	16
2.3.2. Jelszó védelme . . . . .	17
2.3.3. Adatvédelem . . . . .	18
<b>3. Rendszer specifikációja</b>	<b>19</b>
3.1. Felhasználói követelmények . . . . .	19
3.1.1. Felhasználó . . . . .	19
3.1.2. Étterem . . . . .	21
3.2. Rendszerkövetelmények . . . . .	23
3.2.1. Funckionális követelmények . . . . .	23
3.2.2. Nem funkcionális követelmények . . . . .	28
<b>4. Tervezés</b>	<b>30</b>
4.1. Szoftver terv . . . . .	30
4.1.1. Logikai nézet . . . . .	30
4.1.2. Folyamat nézet . . . . .	31
4.2. Adatbázis terv . . . . .	32
4.3. Felhasználói felület terv . . . . .	33
4.3.1. Felhasználó felület . . . . .	34
4.3.2. Étterem felület . . . . .	35
4.4. Verziókövetés . . . . .	36
4.5. Projektmenedzment . . . . .	37
<b>5. Megvalósítás</b>	<b>38</b>
5.1. Szerveroldali komponensek . . . . .	38
5.1.1. API . . . . .	38
5.1.2. Adatbázis . . . . .	40
5.2. Kliensoldali komponens . . . . .	41

<b>6. Felhasználói felület</b>	<b>44</b>
6.1. Belépés és regisztráció	44
6.2. Felhasználó	45
6.2.1. For you oldal	46
6.2.2. Keresés	47
6.2.3. Restaurant és Event oldal	47
6.2.4. Éttermekhez és eseményekhez kapcsolódó funkciók	48
6.2.5. Foglalás és meeting programálás	49
6.2.6. Profil oldal	50
6.3. Étterem	51
6.3.1. Foglalás és Meeting oldal	51
6.3.2. Waitinglist oldal	52
6.3.3. Your Events oldal	53
6.3.4. Your Menu oldal	53
6.3.5. Restaurant oldal	54
6.3.6. Statistics oldal	55
<b>Összefoglaló</b>	<b>56</b>
<b>Ábrák jegyzéke</b>	<b>57</b>
<b>Irodalomjegyzék</b>	<b>59</b>

# **1. fejezet**

## **Bevezető**

Sokszor felmerül a kérdés az emberekben, hogy milyen kikapcsolódási lehetőségek vannak a városunkban, hova lenne érdemes beülni egy kávéra, vagy milyen eseményekre lenne érdemes elmenni. Gyakran hallhatjuk azt, hogy nincsenek a városban lehetőségek, nincsenek általunk kedvelt események megrendezve, legfőképpen akik nem laknak a közelben, azok nehezen találnak megfelelő kikapcsolódási helyet maguknak, mivel nem ismerik a kínálatot. Viszont ez a probléma felmerülhet itt előkkel is, akik csak egy szokásos helyre járnak el mindenig és azt gondolják nincs bőséges választási lehetőség. Ezekre a tévhitekre szeretnék rácáfolni alkalmazásom által, és készíteni egy olyan felületet, ahol átláthatóan találva van a felhasználókkal az éttermek kínálata, ahol szabadon kutathatnak és megtalálhatják a számukra legmegfelelőbb kikapcsolódási helyet.

Kijelenthetjük azt, hogy ma már a technológia fénykorát éljük. Ez ugyanakkor nem vonatkozik éppen mindenre, az éttermek asztalfoglalási rendszere még elég viaszamádott, telefonhívással vagy üzenetküldéssel szokás általában asztalt foglalni. Ezek sokszor problémákat is jelenthetnek, időigényes folyamat lehet kutakodni telefonszámok után, néha nem veszik fel, meg kell keresni hogy hova kell küldeni az üzenetet és megírni őket, nem tudni hogy melyik étteremnél hogyan működik az asztalfoglalás. Sajnos elmondható, hogy nincs egy egységes hely vagy módszer ennek megvalósítására. Éppen ezért jött létre a DineEase alkalmazás, hogy összegyűjtsön minden éttermet és egy egységes környezetet biztosítson a vásárlóknak az asztalfoglalások megvalósítására néhány kattintás által.

Arra is lehetőséget szerettem volna nyújtani, hogy olyan éttermek is megjelenjenek az alkalmazásban, amelyekben inkább nagyobb eseményeket szerveznek, mint például esküvőt, keresztelőt, ballagásokat. Ezekre a helyekre természetesen nem az az eljárás, hogy egyből lefoglaljuk a helyet, hanem egy személyes találkozó során szokás megbeszélni a részleteket és megtekinteni a helyet. Így egy meeting programálást lehet ezekre a helyekre szervezni. Ezzel megkönnyítve a szervezők dolgát, a sok telefonhívástól és kutakodástól.

A következő dolog ami nagy előnyt jelenthet az az, hogy a megalkotott platform lehetőséget nyújt az éttermek számára az eseményeik hirdetésére. Manapság szociális média felületeken vagy plakátokkal szokták hirdetni a megrendezésre kerülő eseményeket. Ezek természetesen bevált módszerek a reklámozásra, viszont olyan módszerek is ezek, hogy ha szerencsénk van találkozunk a hirdetéssel, de ha nem akkor lemaradhatunk az eseményről. Az alkalmazás lehetőséget biztosít az éttermeknek eseményeik közzétételére, a felhasználók ezek között találhatnak személyes preferencia szerint olyan rendezvényeket, amelyekre szívesen elmennének és azonnal foglalhatnak arra a dátumra asztalt.

Az éttermek számára is egy hasznos felületet szerettem volna biztosítani olyan módszerekkel, amikkel megkönnyíthető a dolguk. Egy olyan felülettel találkozhatnak, ahol számon tarthatják asztalfoglalásait vagy meetingeket, eseményhirdetéseket készíthetnek, visszaigazolhatják a foglalásokat, illetve statisztikákat tekinthetnek meg az étterem forgalmáról. Ezzel hozzájárulva ahhoz, hogy könnyedén számon tudják tartani a foglalások és események menetét és információkat merítsenek belőlük.

A következő fejezetekben fogjuk tárgyalni a tervezés és megvalósítás módját és menetét, összehasonlítsokat végzünk, és következtetéseket vonunk le.

## 2. fejezet

# Hasonló alkalmazások, felhasznált technológiák

### 2.1. Hasonló alkalmazások

Dolgozatom ezen részében először megvizsgálunk néhány olyan alkalmazást, amelyek hasonló feladatot végeznek, mint az általam megvalósított alkalmazásunk.

A piacra található néhány asztalfoglaló alkalmazás, amelyek hasonló funkcionálitásokkal vannak ellátva, mint a DineEase alkalmazás. Dolgozatom során megvizsgáltam több alkalmazást is, és elmondható róluk hogy felépítésük nagyon hasonló. Ezek inkább külföldön elterjedtek, Romániában található éttermekre nem sikerült találni ilyen felületet, ezért is egy nagy újításnak számít ennek a dolgozatnak a megvalósítása.

A vizsgált alkalmazások között szerepelnek a következők: [EatApp](#), [Tock](#), [TableList](#), [OpenTable](#).

Ezek nagyon egyszerű alkalmazások, amelyek asztalfoglalással foglalkoznak. Az összesnek van egy Home oldala, ahol különféle ajánlások vannak vagy egyéb érdekességeket mutatnak be. Az EatApp például népszerűség szerint ajánl, vagy a helymeghatározás szerint a legközelebbieket, a TheFork pedig pontozás alapján ajánl különféle éttermeket és listázza a legutóbb megtekintett helyeket. Itt még ajánlják az alkalmazások hogy egy adott tulajdonság szerint szűrjék az éttermeket. Ami közös az összes alkalmazásban, természetesen az, hogy lehet keresni. A vizsgált alkalmazások között a leggyakoribb az egyszerű szó alapú keresés, esetleg lehet szűrni az étterem címe alapján őket, viszont az egyedi alkalmazás, ahol több fajta szűrést is lehet alkalmazni, az az OpenTable.

A éttermek megtekintésénél kilistáznak néhány fontosabb információt róluk, van ahol a menüt is meg lehet tekinteni, de előrendelni nem lehet. Továbbá a legtöbb alkalmazásban lehet kedvencekhez hozzáadni az éttermet, pontozni, vagy megjegyzésekkel írni.

A fontosabb funkcionálitások ezeknél az alkalmazásoknál a fent leírtakban kimerülnek. Ugyanakkor vannak külön oldalak készítve az elmentett vagy kedvencnek bejelölt éttermek megtekintésére, vagy a foglalás előzmények megtekintésére, illetve a TheFork alkalmazás hozzájárul a felhasználók élményéhez promóciós kódokkal. Ezen felül pedig a saját profilt is meg lehet tekinteni és szerkeszteni az adatokat az összes alkalmazásban.

Amely alkalmazás az események hirdetésével is foglalkozik, az nem más mint a TableList. Ez az alkalmazás viszont az éjszakai életet helyezi inkább előtérbe, tehát amerikai

szórakozóhelyek vannak jelen benne, és ezekben megrendezendő események, amelyekről információkat kaphatnak és jelentkezhetnek rájuk.

A felsorolt alkalmazások rendelkeznek hozzájuk tartozó étterem felülettel is, amelyben a foglalásokat kezelhetik. Ezek mind olyan alkalmazások, amelyekre fizetni kell. A TheFork rendelkezik egy ingyenes opciónal is, ahol csak a foglalásait kezeli, de a többi funkcionálisra fizetni kell. Ezekbe a funkcionálisokba beletartozik az ültetés megszervezése, foglalások elfogadása vagy elutasítása, étterem információinak szerkesztése. A TableList még segíti az étterem tulajdonosokat abban, hogy statisztikákkal mérjenek néhány fontosabb adatot.

Összességében ha a megvalósított alkalmazásunkhoz hasonlítjuk a vizsgáltakat, akkor kijelenthetjük, hogy sokkal összetettebb a miénk, mivel ebben az alkalmazásban nem csak éttermek, hanem események is megtalálhatók, illetve olyan éttermek is, amelyekben saját rendezvényeket lehet tartani. Ugyanakkor, megállapítható az is, hogy nagy szükség van egy ilyen alkalmazásra a városunkban is, mivel az összes alkalmazás amely a piacon jelen van, csak és kizárálag külföldi éttermekkel foglalkozik.

## 2.2. Felhasznált technológiák

Az előző fejezetben felsorolt alkalmazásokat technológiai szempontból is megvizsgáltam. Arra a következtetésre jutottam, hogy legelterjedtebb, és jól bevált nyelkekkel íródta ezek az alkalmazások. A leggyakoribb a *JavaScript* különböző felhasználásai voltak általában *React Native* keretrendszer formájában illetve *Node.js* segítségével. Mindemellett néhány alkalmazásban megjelenik a *PHP*, *Java*, *.NET* és a *Go* is, de kis százalékban. Tehát kijelenthető, hogy az alkalmazás fejlesztők túlnyomó része a *JavaScript* nyelvet preferálja.

Feltevődik a kérdés, hogy miért is használják ennyien a *JavaScript* nyelvet, miért ennyire elterjedt? A válasz egyszerű, egy olyan programozási nyelv, amely rengeteg előnnyel jár. Egynézz a platformfüggetlenségének köszönhetően vált ismertté a köztudatban, ami annyit takar, hogy a legtöbb böngészőben natívan fut, ezért nincs szükség külön beépülő modulokra. A nagymértékű könyvtár kínálat és keretrendszer is hozzájárul ahhoz, hogy megkönnyítse a fejlesztők munkáját, mint például a *React*, *Angular* és *Vue.js*. Nem utolsó sorban pedig támogatja az aszinkron programozást, ami különösen hasznos a hálózati műveletek és a felhasználói élmény optimizálása szempontjából. [tea23]

A megvalósított alkalmazásom nem követi ezen alkalmazások példáját, egynézz egediség céljából is és több más megfontolás alapján döntöttem úgy, hogy nem követem más alkalmazások technológiájának sorát, amelyet a következő fejezetekben kifejtök.

### 2.2.1. Flutter

Frontend technológiában a *Flutter*re esett a választás. A *Flutter* egy nyílt forráskódú UI fejlesztési keretrendszer, amelyet a Google hozott létre és a Dart nyelv van a háttérben. Elsősorban mobilalkalmazások fejlesztésére szolgál, de az a különlegesség benne, hogy támogatja a keresztpalatformos fejlesztést is.

Rengeteg előnnyel jár ennek a technológiának a használata.

A hot reload funkciója lehetővé teszi a fejlesztők számára, hogy futtatás során azonnal láthassák a változtatásokat anélkül, hogy újra kellene indítani azt, ez jelentősen fel-

gyorsítja a fejlesztési folyamatot és javítja a produktivitást is. A Dart nyelv, amelyre a Flutter épül egyszerű és könnyen tanulható szintaxiszal rendelkezik, ami gyorsabbá és egyszerűbbé teszi a kódolást.

Ugyan a tervezett platform egy mobilalkalmazás, viszont az is figyelembe volt véve a technológia választása során, hogy minél költséghatékonyabban épüljön fel, és testreszabható legyen más felületekre is. A Flutter lehetővé teszi, hogy egyetlen kód bázissal fejlesszünk iOS, Android, webes és asztali alkalmazásokat is. Ez időt és erőforrásokat takarít meg, mivel nincs szükség külön-külön fejleszteni az egyes platformokra. Közel natív teljesítményt is elérhetünk ezzel a keretrendszerrel, mivel a kód közvetlenül a natív gépi kódra fordul, ami gyors és hatékony alkalmazásokhoz vezet. Így mivel egyetlen kód bázissal több platformot is támogathatunk, a fejlesztés és karbantartás költségei jelentősen csökkennek.

Amivel még hozzájárul a Flutter a fejlesztéshez, az a beépített widgetek segítségével történő testreszabhatóság. Ezekkel könnyen és gyorsan létrehozhatóak felhasználóbarát és jól funkcionáló felhasználói felületek, ezek követik a *Material Design* és *Cupertino* irányelveket, így platformspecifikusabb megjelenést biztosítanak. Ugyanakkor támogatja azt is, hogy a fejlesztők testreszabják a meglévő widgeteket vagy saját widgeteket is létrehozhatnak, amely rugalmasságot biztosít a felhasználói élmény megalapozásában. [LLC24]

## 2.2.2. .NET

A backend technológiája *ASP .NET Core*-ban készült, egy API formájában. A .NET egy C#-ra épülő keretrendszer, ez lehetővé teszi a háttérben futó szerveroldali logika kialakítását és az alkalmazásokkal megvalósuló kommunikációt.

Az *ASP .NET Core* számos előnyvel rendelkezik, ezek között megjelenik a keresztplatformosság is, amely lehetővé teszi az alkalmazások futtatását különböző platformokon. A .NET egy nagyon gyors és optimalizált nyelv, amely hatékony futást tesz lehetővé az alkalmazások számára. Egy másik jellemző, ami miatt a választás erre a technológiára esett az az, hogy kiváló skálázási lehetőségeket kínál, illetve a moduláris felépítése lehetővé teszi az alkalmazások bővítését és karbantartását.

Ez a technológia kommunikációt teremt az adatbázis és a frontend között. Nagyon jól működik a Flutterrel, mivel a HTTP kérések kezelése által lehetővé teszi az adatok cseréjét a Flutter alkalmazással. Támogatja a JSON adatformátumot, ami a legideálisabb a Flutter alkalmazásokban az adatcserére, mivel könnyen és hatékonyan dolgozza fel és kezeli ezeket az adatokat a Flutter. Nem utolsó sorban pedig az aszinkron működés támogatását lehetne említeni mint előny, amely hatékonyan kezeli a hálózati kéréseket, így a frontenddel való együttműködés során az alkalmazások gyorsaságát és hatékonyságát segíti elő. [Cor24a]

### 2.2.3. MSSQL

A *Microsoft SQL Server (MSSQL)* egy relációs adatbáziskezelő rendszer, amelyet a Microsoft fejlesztett ki. Mivel az alkalmazáshoz egy elég nagy adatbázis tartozik, amely minden össze 24 táblát foglal magába, és ezek között szoros kapcsolatok vannak, az adatok nagymennyiségek, több felhasználót is ki kell szolgálnia, ezért a legoptimálisabb választásnak az MSSQL tűnt.

Az MSSQL nagyszerű teljesítményt és skálázhatóságot kínál, és lehetővé teszi, hogy nagy mennyiségeket adatokat hatékonyan kezeljen és feldolgozzon, amely az alkalmazás szempontjából egy kulcsfontosságú tulajdonság. Biztonsági funkciókat is kínál, illetve fejlett adatkezelési képességeket, mindig naprakész rendelkezésre állást (Always On Availability Groups), és beépített adatmentési és helyreállítási lehetőségeket biztosít.

Kiválóan alkalmazható a .NET-el, ugyanis natíván integrálódik, hiszen minden kettő Microsoft technológia. Az *Entity Framework* népszerű ORM eszköz a .NET-ben, ennek felhasználása jelentősen megkönnyítette az adatbázisok kezelését. Ennek segítségével objektumokként lehet kezelni a táblákat, code-first megoldással pedig minimalizálni lehet az SQL kódolás szükségességét. A .NET-el való kapcsolatban az egységes fejlesztői környezet csökkenti a komplexitást és a fejlesztési sebességhez járul hozzá, minden mellett optimizált teljesítményt is nyújt. [Cor24b]

## 2.3. Biztonság

A DineEase alkalmazásban nagy hangsúly van fektetve a biztonságra. Több különféle módszert alkalmazok arra, hogy az adatok biztonságban legyenek.

### 2.3.1. Authentication és authorization

Elsősorban, mivel a bejelentkezés egy kulcsfontosságú lépés az alkalmazás eléréséhez, így a *felhasználói hitelesítés* is fontos szerepet játszik benne. A felhasználó hitelesítési folyamata JSON WEB Tokeneket használ, amelyek tartalmazzák a fontosabb felhasználói adatokat, illetve jogosultságokat. A *JWT* egy állapotmentes hitelesítési módszer, amely segítségével a szerver ellenőrzi a felhasználó jogosultságait is. Ezeknek a tokeneknek a hitelességét biztosítja a szerver digitális aláírása, amely segítségével megbízhatóak maradnak, és a manipulálásukat is meggyártják. A JWT tokenek létrehozását, validálását és aláírását a *Microsoft.IdentityModel.Tokens* és *System.Security.Cryptography* könyvtárral sikerült megoldani. Ezek a tokenek használatosak a szemközti kezelésre is, be van állítva a lejárati dátumuk ezeknek a tokeneknek, amely egy év, így egy évig bejelentkezve tud maradni a felhasználó. [AM19]

Authorization-ra is használatba vettettem a *JWT* tokeneket, amelyek tartalmazzák egy Claim-ben a felhasználó szerepét, amelyet a szerver oldalon ellenőrzünk. Ezekkel meghatározhattuk, hogy a felhasználó melyik erőforrásokat és műveleteket veheti igénybe. Ez a *végpontok védelmére* is szolgál, biztosítva azt, hogy csak az arra jogosult felhasználók férjenek hozzá.

Az alábbi 2.1 ábrán látható egy példa egy User szerepkörű felhasználó tokenjének elkészítésére, ahol tisztán látható, hogy milyen információkat tárolunk el: email, id és sze-

repkör. Ezután az Appsettings konfigurációs fájljában tárolt kulccsal aláíratjuk a tokent, majd kigeneráljuk és visszatérítjük.

```
● ● ●

using Microsoft.IdentityModel.Tokens;
using System.IdentityModel.Tokens.Jwt;
using System.Security.Claims;
using System.Security.Cryptography;

...
private string CreateTokenUser(User user)
{
    List<Claim> claims = new List<Claim>
    {
        new Claim(ClaimTypes.Email, user.Email),
        new Claim(ClaimTypes.NameIdentifier, user.Id.ToString()),
        new Claim(ClaimTypes.Role, "User")
    };

    var key = new SymmetricSecurityKey(System.Text.Encoding.UTF8.GetBytes(
        _configuration.GetSection("AppSettings:Token").Value
    ));
    var creds = new SigningCredentials(key, SecurityAlgorithms.HmacSha512Signature);

    var token = new JwtSecurityToken(
        claims: claims,
        expires: DateTime.Now.AddYears(1),
        signingCredentials: creds);

    var jwt = new JwtSecurityTokenHandler().WriteToken(token);

    return jwt;
}
```

**2.1. ábra.** Tokenizálás

### 2.3.2. Jelszó védelme

A következő fontos szempont az adatok tárolására az volt legfőképpen, hogy a jelszavakat olyan módon tároljuk el, hogy a támadóknak nehezükre essen eze-  
ket feltörni. Erre való tekintettel felhasználtam a .NET beépített könyvtárát, a *System.Security.Cryptography* nevű könyvtárat, amely eszköz használatával garantálható a  
jelszavak és egyéb adatok biztonságos kezelése. Ezen könyvtáron belül a felhasznált algo-  
rithmus a *HMACSHA512* volt, amelyet jelszóhash-elésre használtam. A jelszót egy kulccsal  
kombináljuk, amelyet salt-nak nevezünk, ez egy random érték, amelyből minden felhasználónál új generálódik, majd a HMACSHA512 algoritmus segítségével hash-eljük. Ezzel  
biztosítjuk a jelszavak biztonságos tárolását és védelmét a brute force támadásoktól, mi-  
vel a saltolás növelte biztonságot nyújt, azáltal hogy minden jelszó egyedi hash-t kap, még  
akkor is, ha két felhasználó ugyanazt a jelszót használja. [Sta11]

Az alábbiakban megtekinthető, a 2.2 ábrán, az általam felhasznált függvény, ahogyan hash-eltem a jelszót.

```

● ● ●

using System.Security.Cryptography;

...
private void CreatePasswordHash(string password, out byte[] passwordHash, out byte[] passwordSalt)
{
    using(var hmac = new HMACSHA512())
    {
        passwordSalt = hmac.Key;
        passwordHash = hmac.ComputeHash(System.Text.Encoding.UTF8.GetBytes(password));
    }
}

```

**2.2. ábra.** Jelszó hash-elés

### 2.3.3. Adatvédelem

Az előbb felsorolt biztonsági tényezők csak a backend oldalról szóltak, de beszéljünk kicsit a frontendról is, ugyanis itt is ugyanolyan fontos tényező a biztonság. Flutterben két módszert is használtam a biztonság magasabb szintre emelésére: *SharedPreferences* és *Provider* használatával kezeltem az érzékeny adatokat, ezek Flutter által készített csomagok, kiemelkedően a biztonság céljára vannak kifejlesztve. A Provider segítségével történt a token kezelése, míg a SharedPreferences segítségével tároltam néhány fontosabb adatot.[RSB23]

A biztonság növelésére a naplózást használtam, a *Logger* csomagot felhasználva. Ez azért fontos, mivel a rendszer kritikus eseményeit elnaplózhatjuk, például a bejelentkezási kísérleteket, rendszerhibákat. Itt viszont figyelni kell arra, hogy titkos információkat semmiképpen sem szabad naplózni, tehát jelszavaknál használatba venni ezt tilos. [ZCSC19]

Az alkalmazás biztonságához még kiemelném az *input validation*ót is, úgy szerver oldalon mint a kliens oldalon. Ez biztosítja azt, hogy a bevitt adatok megfeleljenek az elvárásoknak és a káros tartalmakat megszűrjük. Megelőzhetőek vele az input alapú támadások is, mint például a cross scripting vagy a parancs injekció.[HO06]

## 3. fejezet

# Rendszer specifikációja

### 3.1. Felhasználói követelmények

A felhasználókat két kategóriára osztjuk: étterem és általános felhasználó, akik különböző funkcionálitásokkal találkozhatnak az alkalmazás használata során.

#### 3.1.1. Felhasználó

- **Platform**

Elérhető egy mobil alkalmazás, amelyen keresztül böngészhetnek a felhasználók az éttermek és azok eseményei között, és foglalásokat tehetnek le, illetve időpontokat kérhetnek személyes találkozóra nagyobb események esetében.

- **Belépés**

A felhasználónak van egy bejelentkezés és egy regisztrációs oldal készítve. Ha a felhasználó még nincs beregisztrálva, az megteheti a regisztrációs oldalon a szükséges adatok kitöltésével, majd a fiók létrehozása után beléphet a fiókjába.

- **For you oldal**

A sikeres bejelentkezés után megjelenik egy "For you"-nak nevezett oldal, amely a nevéből is adódóan olyan oldal, amelyen ajánlások szerepelnek előző foglalások, pontozások, vagy kedvencek listája alapján, itt események és éttermek listája is megjelenik.

- **Restaurants oldal**

Itt megjelennek az éttermek listái, ezek között lehet böngészni, vagy kedvelni őket.

- **Events oldal**

Ha a felhasználó eseményeket szeretne keresni, ezen az oldalon megtalálhatja az összes étteremben megrendezésre kerülő eseményt meghirdetve.

- **Restaurants for events oldal**

Ezen az oldalon megtalálható olyan éttermek listái, ahol nagyobb szabású eseményeket lehet szervezni.

- **Search, filter**

Lehetőségük nyílik keresni is az éttermek és események között szavak alapján, és szűrni is lehet különböző tulajdonságok alapján, mint például kategória, árkategória, pontozás, és sok más.

- **Profil**

A profil oldalon a felhasználó szerkesztheti személyes adatait és megtekintheti aktivitását. Itt találhatja meg azon éttermek listáját, amelyeket kedvelt, pontozott, vagy véleményt írt róluk, ezeket szerkesztheti vagy törölheti. Továbbá itt tudja megtekinteni az eddig foglalásait vagy az elkövetkezőket, illetve itt jelenik meg, éppen melyik étterem válaszára vár. A kijelentkezés is ezen az oldalon található.

- **Restaurant details oldal**

Itt található meg minden fontos információ az étteremről, itt lehet kedvelni az éttermet, véleményt írni róla, megtekinteni a menüt.

- **Event details oldal**

Ezen az oldalon az esemény információit lehet megtekinteni.

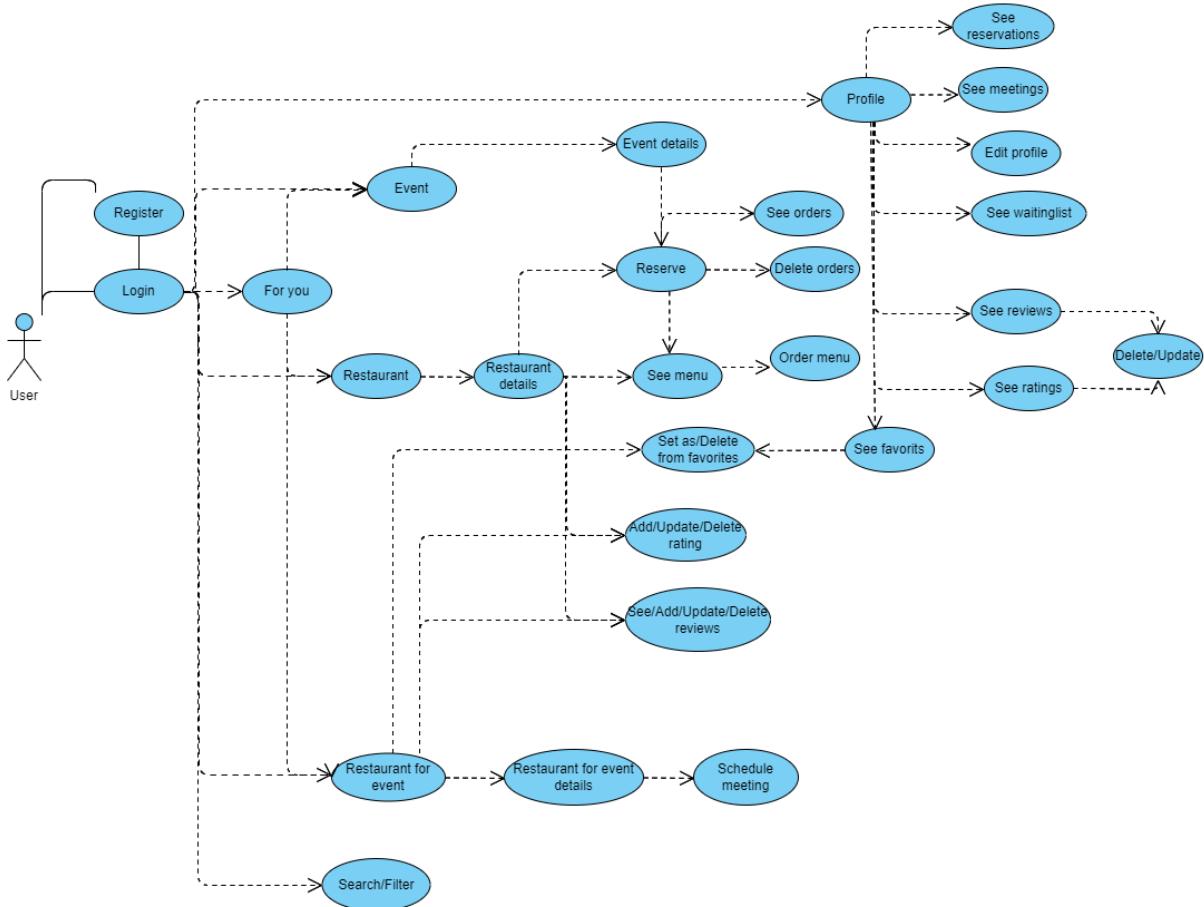
- **Foglalás**

Az éttermek és események details oldaláról is el lehet érni ezt a funkciót. Amikor a felhasználó úgy dönt, hogy foglalni szeretne egy asztalt, rákattint egy gombra, és az elvezeti arra az oldalra, ahol kitölthet néhány információt a foglalásról, esetlegesen előrendelhet a menüből valamit, és egy gombnyomással el is küldi a foglalás kérését, amelyet mielőbb visszaigazol az étterem.

- **Meeting programálás**

A Restaurant for events oldalon elérhető éttermekhez foglalás helyett a meeting programálás funkció jött létre. Hasonlóan működik ez is, mint az asztal foglalás, itt is kitölt néhány szükséges adatot, és egy gombnyomás után már csak a visszaigazoló értesítésre kell várni.

A [3.1](#) ábrán látható a felhasználói felület use-case diagramja, amely tartalmazza az alkalmazás fontosabb funkcionálisait és azokat az oldalakat, ahonnan elérhetőek ezek a funkcionálisok .



**3.1. ábra.** Felhasználói use-case diagram

### 3.1.2. Étterem

Az éttermek két kategóriára sorolhatók: átlagos éttermek és rendezvényekre speciálizálódott éttermek. A rendezvényekre specializálódott éttermekben meetingeket lehet szervezni, ahol megbeszélhetik a tervezett esemény menetét és részleteit.

- **Platform**

Elérhető egy mobil applikáción, vagy akár egy webes felületen keresztül is az éttermeknek egy alkalmazás, amelyen keresztül kezelhetik a foglalásokat és egyéb hasznos funkciókkal találkozhatnak.

- **Belépés**

Az étteremnek van egy regisztrációs és egy bejelentkezés oldal készítve. Itt ugyanazzal a bejelentkező oldallal lehet találkozni, mint a felhasználóknál, mivel képes az alkalmazás megkülönböztetni, hogy ki lépik be éppen: az étterem vagy a felhasználó. A regisztrációtól eltekintve a fontosabb információit az étteremnek, majd a részleteket hozzáadhatják a profil oldalukon.

- **Reservations vagy Meeting oldal**

Ahogy bejelentkezünk, már megjelenik a foglalásokat vagy meetingeket megjelenítő oldal (ez attól függ, milyen kategóriába tartozik az étterem), ahol megtekintheti, hogy milyen időpontok foglaltak és a hozzájuk tartozó részleteket egy naptár segítségével, amelyet háromféle nézetben lehet megtekinteni: nap, hét, hónap.

- **Waitinglist oldal**

Ezen az oldalon jelennek meg azok a foglalás- vagy meetingkérések, amelyeket át kell néznie az étteremnek és vissza kell jeleznie, ha tudják fogadni a foglalási dátumra, esetleg megjegyzéseket is írhat, vagy válaszolhat a felhasználók kérdéseire. Itt ha elfogadja az asztalfoglalást, akkor át is kerül a Reservations oldalra az.

- **Events oldal**

Itt jelennek meg az eddig megszervezett események és a jövőben megszervezendők is. Ezen az oldalon lehet újabb eseményeket is hozzáadni.

- **Menu oldal**

Ezen az oldalon találhatóak az étteremhez tartozó menük. Itt hozzáadahat újat és szerkeszthet régi menüket.

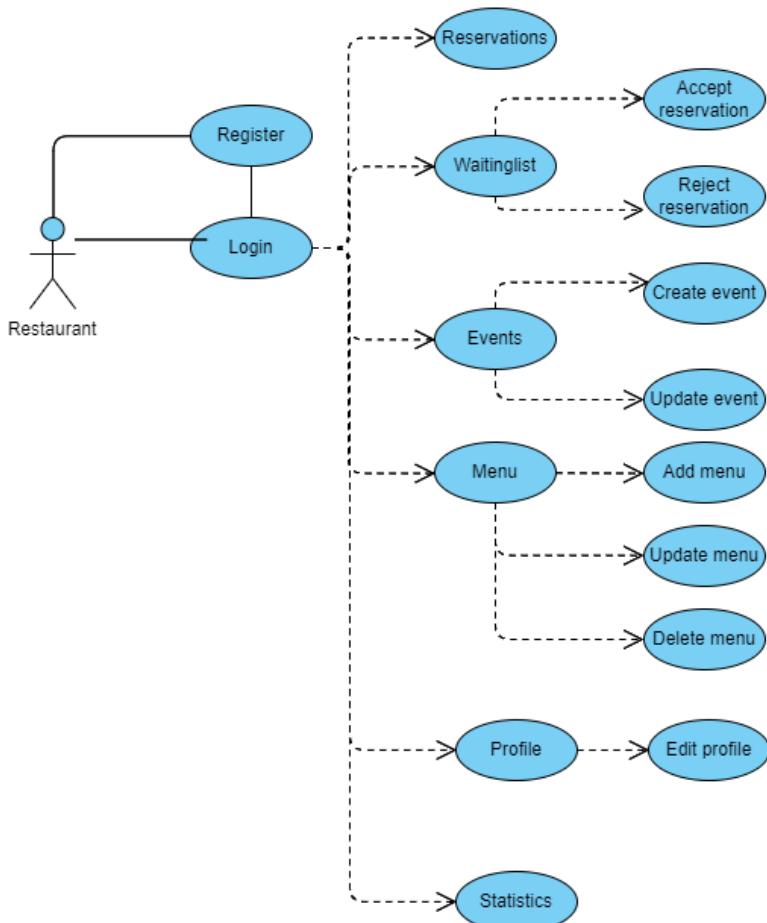
- **Statisztikák**

Megtekinthetők a foglalásokkal és eseményekkel kapcsolatos információk, valamint az alkalmazásban történt étteremmel kapcsolatos interakciók mérése, például pontozások, illetve foglalással és eseményekkel kapcsolatos statisztikák.

- **Étterem profil**

Ezen az oldalon lehet frissíteni az étteremhez kapcsolódó információkat.

Az étteremhez tartozó use-case diagram a [3.2](#) ábrán megtekinthető, ahol láthatóak az elérhető oldalak és azok fontosabb funkcionálitásai.



**3.2. ábra.** Étterem use-case diagram

## 3.2. Rendszerkövetelmények

### 3.2.1. Funkcionális követelmények

Az alábbiakban bemutatok minden fontos funkciót az alkalmazásból.

- **Regisztráció**

Úgy a *felhasználónak* mint az étteremnek elérhető a regisztráció funkció, ahol megadják a szükséges adatokat. Ezeknek megadása után lépik fel a hibakezelés, a rendszer megvizsgálja, hogy minden szükséges mező ki van-e töltve, ha nem akkor nem engedi tovább a regisztrálandó felet és egy üzenettel jelzi, hogy ki kell tölteni mezőket. Ugyanígy, hogyha rossz formátumban adta meg adatait, például a telefonszám-ba beírt betűket is és nem megfelelő a hossza vagy ha a kétszer megadott jelszó nem egyezik meg, akkor egy üzenettel jelzi ezt. Miután a felhasználónak sikeresen megadni adatait, a rendszer elmenti az adatbázisba az adatait biztonságban, tehát a megadott jelszó egy bizonyos salt értékkel és hash függvénytel titkosítva lesz eltárolva az adatbázisban. A sikeres műveletek után pedig a felhasználó bejelentkezhet fiókjába és a további funkciókat is elérheti.

- **Bejelentkezés**

Ez a funkció is természetesen elérhető a felhasználók *mindkét* fajtájának. A felhasználó meg kell adja e-mail címét és jelszavát. Itt leellenőrződik az hogy létezik-e felhasználó a megadott e-mail címmel és megyezik-e a felhasználó jelszava a megadott jelszóval, ezekre a tévedésekre is visszajelzést küld az alkalmazás. Ha sikeres a bejelentkezés, akkor a felhasználó kap egy JWT tokenet, ez tárolja a meghatározó adatokat a felhasználóról mint az e-mail cím, role (ez lehet user és restaurant), stb. Ez egy hélig érvényes, tehát amíg le nem jár ez a token, addig elérheti a bejelentkezést követő egy héten az applikációt a felhasználó, úgy hogy nem kell bejelentkeznie. Ezek után pedig szabadon használhatja szerepkörének megfelelő funkciókat, amelyeket a következőkben részletezünk.

## Felhasználó

- **Keresés, filter**

Az applikáción belül a felhasználóknak van lehetőségük keresni egy adott szó alapján, ez a keresés az események és éttermek névében illetve leírásaikban keresnek, ha megjelenik bennük bárhol az a betűkombináció, amelyre rákeresett a felhasználó, eredményül visszaadja ezeket, elválasztva az események és éttermek listáját. A filter funkció pedig lehetőséget biztosít arra, hogy különböző kritériumok alapján listázzuk az éttermeket, mint például kategória, árkategória, étel preferencia.

- **Kedvenc éttermek**

A felhasználóknak lehetőségük van az éttermek kedvelésére, egy szív alakú ikonra kattintva. Ezek a kedvencnek kijelölt éttermek megjelennek majd a felhasználói profilban, könnyebb elérhetőséget biztosítva ezzel. Ezeket el is lehet távolítani a kedvencek közül, ha újra rákattint a szív alakú ikonra.

- **Értékelés**

Az éttermeket lehet értékelni egy és öt közötti csillaggal. Az értékeléseket lehet szerkeszteni is, illetve törlni. A felhasználók által adott értékelések átlagával határozzuk meg, milyen értékelésű egy étterem. A saját értékeléseit megtekintheti és szerkesztheti a felhasználó úgy a profil oldaláról, mint az étterem oldaláról is.

- **Visszajelzés**

Visszajelzéseket lehet írni a étteremről, ha a felhasználónak bármi megjegyzése, véleménye lenne. Azokat a visszajelzéseket, amelyeket a felhasználó írt, saját profiljáról bejelentkezve törölheti vagy szerkesztheti. Ezek megjelennek a profil oldalán is kilistázva, és ugyanezeket a műveleteket végrehajthatja ott is.

- **Asztalfoglalás**

Lehetőségük van egy adott étterembe asztalt foglalni a felhasználóknak. Ha nem tölti ki megfelelően a mezőket, akkor nem engedi az alkalmazás, hogy foglalást küldjön, tehát megjelenik egy üzenet, hogy helytelen adatot adott meg, például a telefonszám betűt is tartalmaz vagy nincs kitöltve minden kötelező mező. Itt le van kezelve, hogy a múltra ne lehessen foglalást küldeni és olyan időpontra se,

amikor az étterem nincs nyitva. Miután helyesen töltötte ki, felkerül a foglalása egy várólistára, ezt a várólistát a felhasználó a profiljánál tekintheti meg és vár arra, hogy az étterem elfogadja a kérést. Ha elfogadta az étterem vagy elutasította, akkor átkerül a foglalások felületére ez ahol megtekinthető lesz az összes foglalás.

- **Menü rendelés**

Az asztalfoglaláson belül a felhasználó előrendelhet az étterem menüiből. Ez a rendelés a foglaláshoz lesz csatolva. Természetesen vissza is lehet vonni a rendelést, amíg nem küldte el a foglalási kérést.

- **Meeting beütemezés**

A "restaurant for event"-nek nevezett éttermeknél asztalfoglalás helyett meetinget lehet programálni, ahol kötelezően jövőbeli dátumot kell kiválasztani a meeting és a szervezett esemény dátumának, illetve helyesen kell megadni minden adatot, különben egy hibaüzenettel fog szembesülni a felhasználó, amelyben közölve lesz, hogy helytelen adatokat adott meg. Sikeres meeting beütemezése után ugyanúgy a várólistára kerül ez, és az étteremnek kell majd visszajelennie. Ugyanúgy ezt is a profil oldalnak a waitinglist felületén megtekintheti majd a felhasználó, elfogadás vagy elutasítás után pedig a meeting felületén lehet kilistázni a meetingeket.

- **Adatok szerkesztése**

A felhasználó szerkesztheti személyes adatait a profil oldalán, ahol helyes formában kell megadnia minden, különben nem hajtható végre a frissítés.

## Étterem

Az éttermek két típusra oszthatók: átlagos éttermek (restaurant) és eseményekre szánt éttermek (restaurant for event). Ezek között nincs nagy különbség, csak annyi, hogy az éttermeknél asztalt lehet foglalni, az eseményekre szánt éttermeknél pedig meetinget lehet beütemezni az étterem megtekintésére és egy megbeszélésre az esemény részleteiről.

- **Foglalások kezelése**

Az étteremnek van egy várólistája, amelyben el lehet fogadni vagy el lehet utasítani a foglalási kéréseket. Itt, az elfogadásnál opcionálisan meg lehet adni, hányas asztalhoz ültetik őket, kizárálag számmal, másnépp nem fogadhatja el a foglalást és hibaüzenetet jelez ki az alkalmazás, illetve opcionálisan írhat egy commentet is a felhasználónak. Elfogadás után bekerül az asztalfoglalás naptárba, ahol megtekintheti az összes foglalás részletét. Ugyanakkor miután elfogadta vagy elutasította, annak a felhasználónak, aki foglalta az asztalt, megjelenik a foglalásoknál, hogy elfogadott vagy elutasított státuszba került.

- **Meetingek kezelése**

Ennél ugyanaz érvényes, mint a foglalásoknál, el lehet fogadni és elfogadásnál opcionálisan visszajelzést írni vagy el lehet utasítani. Az a különbség, hogy itt a meeting naptárba kerül be az elfogadott meeting és a felhasználó is a meeting oldalon tekintheti meg, hogy el lett fogadva vagy sem a kérése.

- **Események kezelése**

Az éttermek létrehozhatnak eseményeket, a kikötés az, hogy csak és kizárálag jövőbeli dátum szerepeljen, és minden kötelező mezőt töltön ki helyes adatokkal, ellenben hibaüzenettel fog találkozni, és a javításig nem lehet létrehozni az eseményt. Azokat az eseményeket, amelyeket már meghirdetett és még nem jártak le, szerkeszteni lehet, természetesen itt is kulcsfontosságú, hogy megfelelő adatokkal tölték ki.

- **Menü kezelése**

Az éttermek megadhatnak új menüket, vagy szerkeszthetik a már meglévő menüket is. Itt is fontos, hogy megfelelő adatokat adjanak meg a menüről.

- **Adatok kezelése**

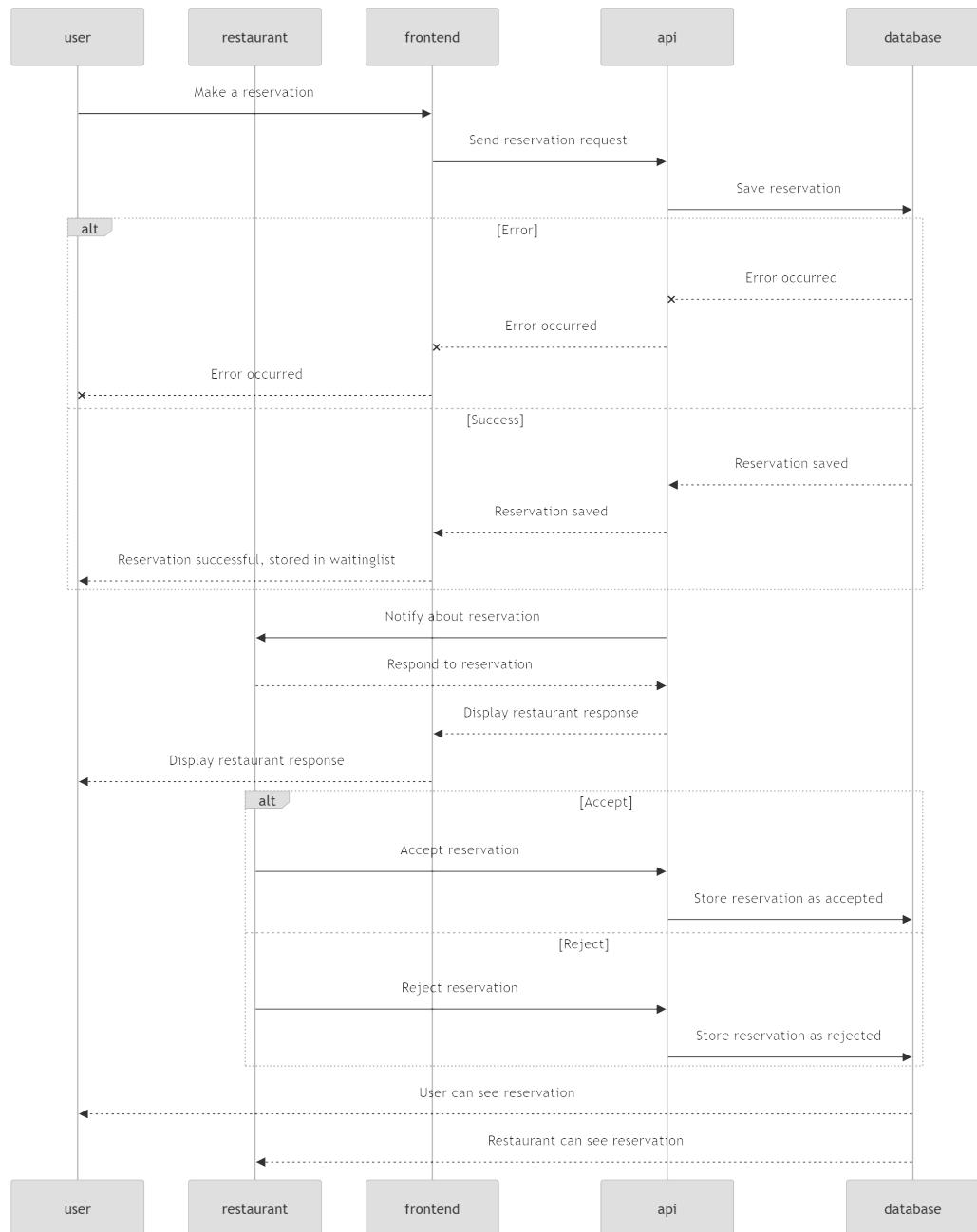
Megadhatnak részleteket az éttermek magukról. Étterem kategóriákat, étkezési kategóriákat, ültetési opciókat és még sok más adatot lehet hozzáadni és szerkeszteni, fényképet is fel lehet tölteni az étteremről vagy törölni is lehet ezeket. Illetve a már megadott adatokat lehet szerkeszteni.

- **Visszajelzések kezelése**

A Statisztika oldalon megtekinthetők többek között a felhasználók által írt visszajelzések az étteremről, ezeket az étteremnek jogában áll törölni. Ezt olyan szempontból láttam szükségesnek beintegrálni az alkalmazásba, hogy az éttermek tudják kezelni a bántó, fenyedegető, vagy trágár megnyilvánulásokat.

A következő 3.3 ábrán szemléltetem az asztalfoglalásnak a szekvencia diagramját, amelyben megfigyelhető, hogyan történik a komponensek között a kommunikáció.

A felhasználó próbál foglalni egy asztalt, hiba esetében kiírja azt, a foglalás továbbítódik az API-n keresztül az adatbázisnak, amely eltárolódik a waitinglist tartalmaként. Az étterem értesül a foglalásról, majd az választ ad rá, és frissül az adatbázis a foglalásra adott válasszal, mindenek után pedig ugyanúgy a felhasználó és az étterem megtekintheti foglalásait és azokra kapott vagy adott válaszait.



**3.3. ábra.** Asztalfoglalás szekvencia diagram

### 3.2.2. Nem funkcionális követelmények

Az applikáció elsősorban mobil alkalmazásnak, azon belül Android alkalmazásnak készült, viszont kihasználva a Flutter nyújtotta előnyöket, webes felületen is megfelelően működik, mivel a Flutter programozási nyelvvel könnyedén lehet cross-platform alkalmazásokat készíteni. Legfőképpen az étterem felületét készítettem úgy el, hogy jól használható legyen webes felületen is.

- **Operációs rendszer**

Az alkalmazás használatához szükséges egy **Android telefon**, amely **Android 4.1 és ez fölötti verziót** használ, ha **web-en** szeretné futtatni a felhasználó, akkor a következő böngésző típusok valamelyikével kell rendelkeznie: **Chrome 84, Firefox 72.0, Edge 1.2.0 és ezek fölötti verziók**. Illetve **Windows-os** futtatásra is van lehetőség **Windows 7 64-bit-es és fölölte lévő verziókkal**. [Tea24]

- **Tárhely**

Tárhely szempontjából legalább **200 MB**-ra van szükség.

- **Internet**

Az alkalmazás működéséhez szükség van internetkapcsolatra.

- **Teljesítmény**

A rendszer sebességének megfelelőnek kell lennie, a válaszidőnek gyorsnak kell lennie, így igyekeztem optimálisan megoldani minden lekérdezést.

- **Megbízhatóság**

Egy olyan platformot sikerült létrehozni, amely törekedik a hibamentességre, illetve ahol lehet ott a hibakezelés meg van oldva. Ezzel megelőzve, hogy ne crash-eljen az alkalmazás, ha hiba történik.

- **Biztonság**

Az alkalmazás biztonságos környezetet nyújt a felhasználók számára. Elsősorban az adatok tárolása biztonságosan van megoldva, a jelszó titkosítva van eltárolva, így meggyárolva a támadókat. Authentication-ra JWT tokenek vannak használva, amelyekben titkosítva tárolódnak a bejelentkezett felhasználó fontosabb adatai. Authorization szempontjából pedig, úgy van megoldva, hogy minden lekérést csak egy adott szerepkörrel rendelkező felhasználó érheti el, más nem. Ezekkel támogatjuk a felhasználók adatainak biztonságát, jogosultságkezelésekét és támadások elleni védelmet.

- **Használhatóság**

Az alkalmazás elkészítésekor odafigyeltem a felhasználói élmény javítására is. Nem szerettem volna egy szétszórt platformot készíteni, mindenkorral egy letisztult, egyszerű felhasználói felület volt a cél, minimalista designnal. Az alkalmazásra jellemző a lekerekített forma, a színek használatával se szerettem volna túlzásba esni,

így a jellemző szín a narancssárga lett. Egyszerű kezelhetőséget biztosít az alkalmazás, ikonok segítségével hívjuk fel a fontosabb területekre a figyelmet, letisztult voltának köszönhetően pedig átláthatóságot nyújt.

- **Hordozhatóság**

A rendszer bármilyen környezetben futtatható, a Flutter nyújtotta cross-platform lehetőséget kihasználva, így kijelenthető hogy az alkalmazásnak a hordozhatósága nagyon jó.

- **Karbantarthatóság**

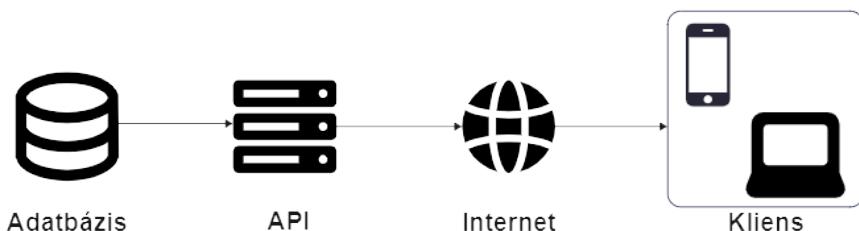
Az alkalmazás felépítése nagyon jól átlátható, minden fontosabb komponens külön mappába van szervezve. Például a backendnél felhasználtam a rétegezett architektúra elvét. A *rétegezett architektúra* a következőkből áll: *prezentációs réteg* - amely tartalmazza a felhasználói interfésszel kapcsolatos komponenseket, az én esetben a controllereket, alkalmazási réteg - itt az üzleti logikát megvalósító komponensek vannak jelen, *üzleti logikai réteg* - ahol az entitások és a hozzájuk tartozó logika található, *adat hozzáférési réteg* - ahol kezeljük az adatházis hozzáférést (repository, interface), *DTO*, *mapperek* - amelyek használatosak a különböző rétegek közötti átalakításra. Itt a kód felhasználhatóságra is figyeltem, például generic típusú repository-t hoztam létre a fontosabb függvényekre, általánosítás céljából és az időigényt megspórolva.

# 4. fejezet

## Tervezés

### 4.1. Szoftver terv

A rendszer architektúrája három fontos komponensből épül fel: felhasználói interfész (mobil vagy web), API és adatbázis. Ezek működéséhez pedig van még egy kulcsfontoságú komponens, ami nélkül nem működhetne az alkalmazás: az internet. A rendszer architektúráját a 4.1 ábrán meg lehet tekinteni, ahol látható, hogyan kapcsolódnak egymáshoz ezek.



4.1. ábra. Rendszer architektúrája

#### 4.1.1. Logikai nézet

Egy *mobil vagy webes felület* szolgáltatja a felhasználói interfészt, ahol a felhasználók interakcióba léphetnek az alkalmazás funkcióival. Ennek megvalósítását Flutter keretrendszerrel sikerült megoldani.

Az *API* nagyon fontos szerepet tölt be a szoftverben, mivel ez a komponens szolgáltatja az adatokat és a logikát, a kommunikációt az adatbázis és az interfész között. Ez .NET keretrendszer segítségével készült el. Itt valósul meg az autentikáció, az adatok kezelése és a kommunikálás az adatbázissal.

Az *adatbázis* szolgál az összes adat tárolására. Itt Code First megközelítést alkalmaztam az adatbázis modellek létrehozásához és kezeléséhez. Itt meg vannak határozva a táblákban a típusok, kulcsok, és kapcsolatok, amelyek segítségével könnyen kezelhető az alkalmazás.

Így elmondható, hogy a kommunikáció a komponensek között úgy zajlik, hogy a felhasználói felületen történik egy kérés a szerver felé, ha az internet kapcsolat rossz,

akkor nem sikerül a kérés megvalósítása, viszont ha van internet kapcsolat, akkor a kérés továbbítódik az API felé, az API feldolgozza a kérést és hogyha helyes adatok vannak megadva, az adatbázist segítségül hívva eléri az adatokat vagy manipulálja azokat.

#### **4.1.2. Folyamat nézet**

A következőkben bemutatom néhány fontos folyamatát, dinamikus viselkedését a rendszernek, különös tekintettel a futásidőbeli aspektusokra, hogy jobban átláthatassuk a rendszer viselkedését.

##### **1. Bejelentkezés folyamata**

- Felhasználó megadja adatait: beírja email címét és jelszavát a felhasználói felületen
- Az adatok átadódnak az API-hoz, itt ellenőrződik hogy hitelesek-e a megadott adatok
- Ha sikerült az adatokat hitelesíteni, akkor JWT token generálódik az API-ban, és ezt visszaküldi a felhasználónak
- A token tárolása a kliens oldalon Shared Preferences csomag felhasználásával történik, és ez a következő kérések során felhasználható lesz az azonosításhoz

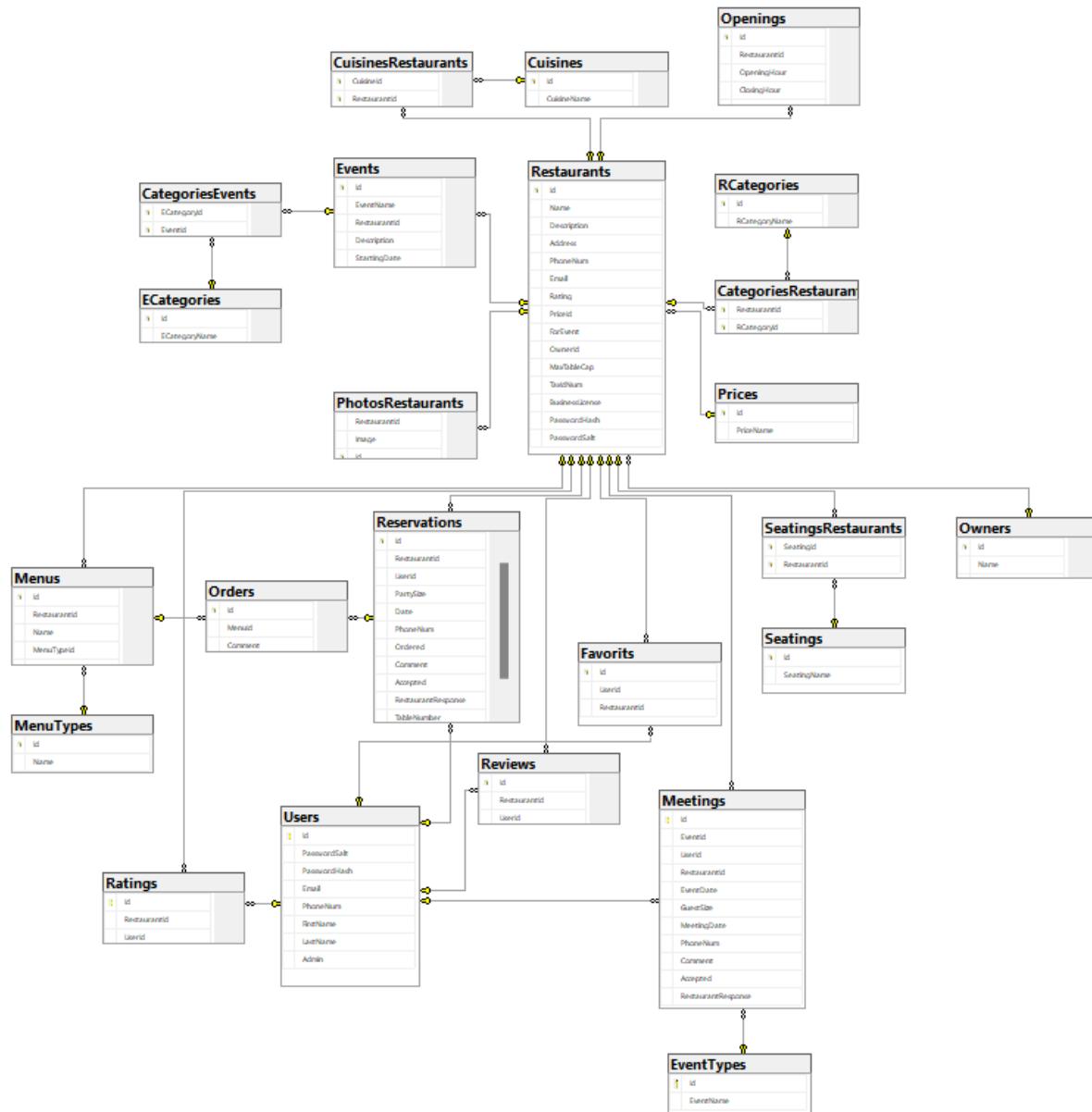
##### **2. Asztalfoglalás folyamata**

- Felhasználó rákattint a foglalás gombra és megadja szükséges adatait helyesen az asztalfoglaláshoz, ez kérést küld az API-nak
- Az API fogadja az adatokat és ellenőrzi azok helyességét
- Ha helyes adatokat adott meg és nem történik semmi hiba, az API segítségével frissül az adatbázis és visszaigazolást küld a felhasználónak általában 200-as exit code-al.

##### **3. Esemény hirdetése**

- Az étterem új eseményt hirdet a felhasználói felületen, kitöltve a szükséges adatokat helyesen
- A megadott adatokat az API validálja és ha helyesek elmenti őket az adatbázisba
- Visszaigazolást küld az étteremnek az API hogy sikeres volt-e a létrehozása.

## 4.2. Adatbázis terv



4.2. ábra. Adatbázis terv

Az adatbázis tervezésében az első lépés az volt, hogy meghatározzam, hogy relációs adatbázist használjak vagy sem. Miután átgondoltam és megterveztem, az alkalmazás témaját, már nem volt kérdés, hogy milyen adatbázist használjon az alkalmazás. Sok részletet rögzít az alkalmazás, rengeteg különféle kapcsolatot alakít ki, minden össze 24 táblával.

Több nagyobb táblát kellett létrehoznom, mint például az étterem, felhasználó, meeting, foglalás és így tovább. Ugyanakkor a táblák számát legfőképpen a kisebb táblák bővítték, amelyek általában valamivel a tulajdonságait tárolja, ezeknek many-to-many kapcsolatuk volt, így egyre több táblával kellett számolni.

A sok reláció miatt egy olyan adatbázis-kezelő rendszert kellett keresni, amely könnyen karbantartható, elkerülve az adat redundanciát és biztosítja az adatok integritását.

Olyan szempontból esett a választás az MSSQL-re , mivel kívály teljesítményt és skálázhatóságot nyújt nagy mennyiségű komplex lekérdezések esetén. Nagyon jól integrálható a fejlesztői eszközökkel, az én esetben a Visual Studio-val, és rendelkezik fejlett adatbázis-kezelő eszközzel, mint az SQL Server Management Studio, amely nagy segítséget nyújtott a fejlesztés során.

Az adatbázisterv a [4.2](#) ábrán látható.

### 4.3. Felhasználói felület terv

A felhasználói felületeket papíron terveztem meg, kézzel rajzolva. A végeredmény nagyon hasonló lett, bár vannak némi újítások, van ahova még bekerült új screen, viszont az összkép megalkotására és ötletelésre nagyon hasznos volt ennek megtervezése. Amikor a design-t készítettem az alkalmazásnak, ezeket a dokumentumokat vettettem elő, és próbáltam minél hasonlóbban elkészíteni. A tervben szerepel 21 screen, amelyet a fejlesztés során optimálisabbnak tartottam bővíteni, illetve új ötletek is kerültek be, így a végeredményben 29 darab screen valósult meg, ha a splash screen-t is beleszámítjuk, akkor összesen 30.

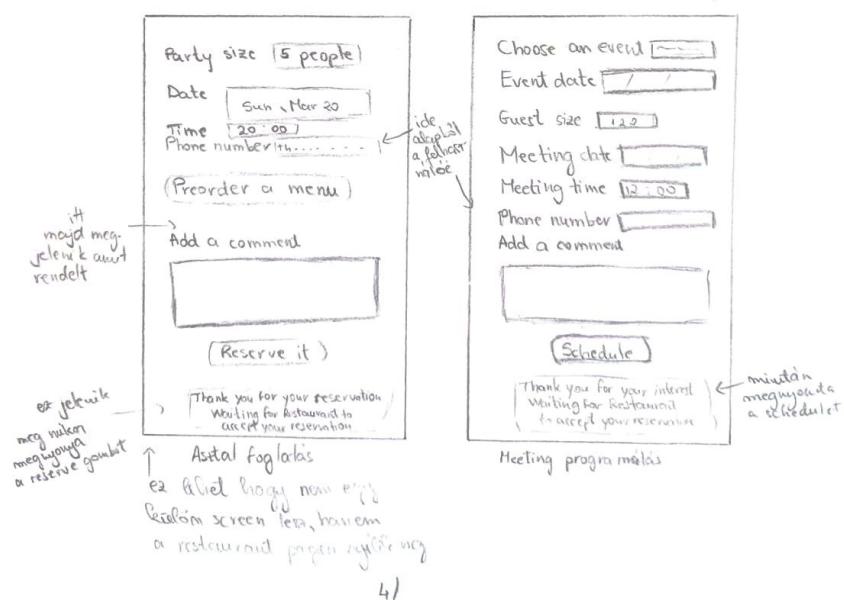
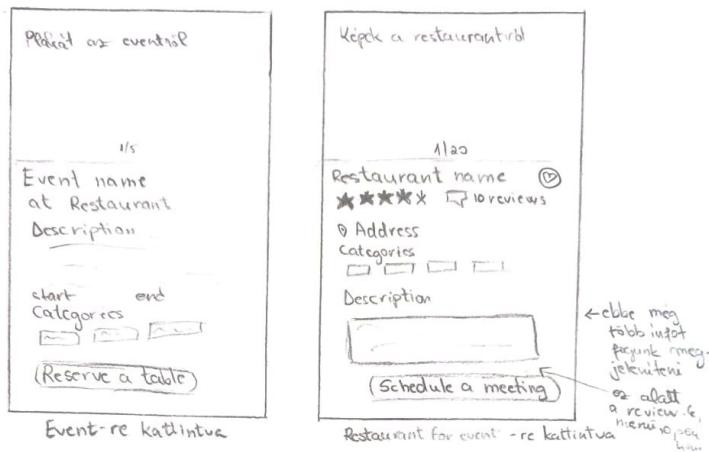
Mivel két szerepkör található meg az alkalmazáson belül, így értelemszerűen különböző funkciókkal és felületekkel találkozhatnak, a szerepüktől függően. A következő két alfejezetben bemutatom ezek tervét.

### 4.3.1. Felhasználó felület

Az átlagos felhasználó szerepkörhöz tartozó kliensnek elsősorban mobilra lett tervezve az alkalmazás. A keresztplatform nyújtotta előnyök viszont a fejemberben voltak, ezért feltétlenül olyan megközelítést használtam, amellyel úgy gondoltam, hogy ha webes felületen futtatnánk, nem kellene nagy változtatás és jól nézne ki ott is.

A tervben 16 darab screen jelenik meg, ugyanakkor a kivitelezésnél még merültek fel olyan esetek, amelyekre újabb képernyőket kellett létrehozni.

Az alábbi, 4.3 ábrán látható képen megtekinthető a szkennelt változata a tervem egyik oldalának, ehhez hasonlóan készült el a többi képernyő terve is.



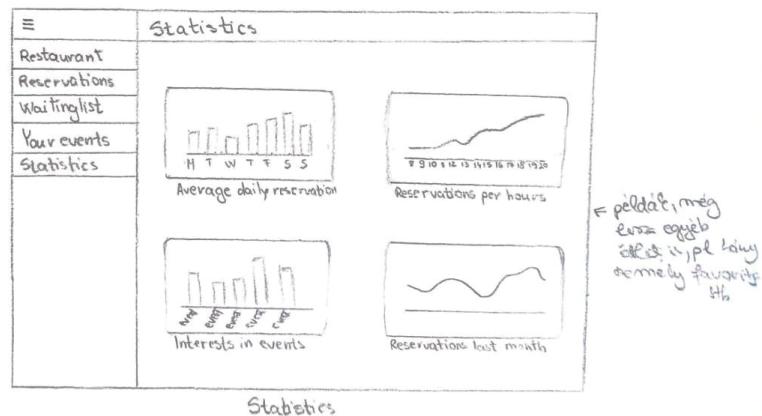
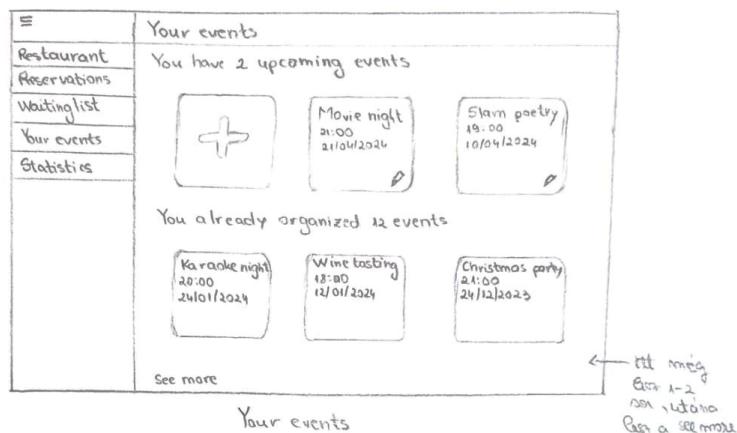
4.3. ábra. Felhasználó felület terv

### 4.3.2. Étterem felület

Az étterem felületét ugyancsak papíron terveztem. Itt webes megközelítést használtam a tervezésnél, viszont ugyanúgy figyelembe volt véve, hogy ez egy mobiltelefonon fusson elsősorban megfelelően és jól mutasson.

Végül a megalkotásnál elsősorban itt is a mobilos felületre készült el, annak ellenére hogy webes felületre volt megtervezve. Természetesen itt is kerültek be új képernyők, viszont a struktúra és design teljes egészében követve volt.

Az alábbiakban, a 4.4 ábrán, megtekinthető egy példa, hogyan nézett ki a tervezése az étterem felületnek.



2|

**4.4. ábra.** Étterem felület terv

## 4.4. Verziókövetés

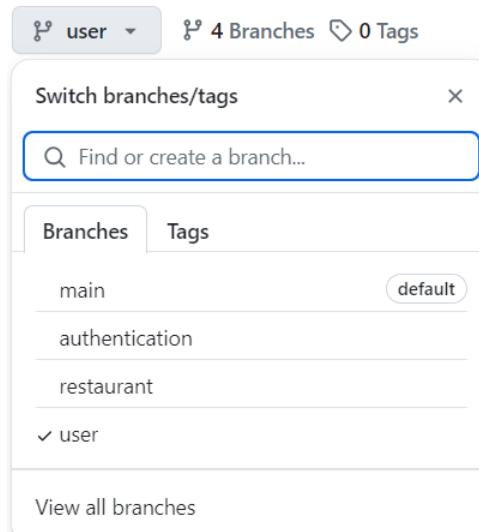
A verziókövetés elengedhetetlen volt a projekt elkészítése során, ugyanis egy nagy projektről van szó, amelyben felléphetnek hibák és problémák, valamint több funkcionálitást kell implementálni. Ezért biztonságosabb esetben ha használatba veszünk egy verziókövető eszközt, amely az én esetben a GitHub volt, amelyet a 4.5 ábrán meg lehet tekinteni.

A projekt egy Repository-ból állt, ahova külön mappában feltöltésre került a front-end és a backend kódja, és ezekhez tartozó .gitignore fájlok, amelyekben szerepelnek a nem publikus és mások számára szükségtelen fájlok, ennek segítségével kiszűrjük ezeket. minden változtatásnál commit-oltam, hogy mindenkor legfrissebb, működő verzió legyen megőrizve.

aksza fixed event listing		688a357 · 16 hours ago	111 Commits
DineEaseApp	fixed event listing	16 hours ago	
dine_ease	fixed event listing	16 hours ago	
.gitignore	created api	2 months ago	
README.md	Initial commit	3 months ago	

4.5. ábra. Verziókövető rendszer

Ahhoz, hogy minden hatékonyan tudjak programozni, készült néhány branch (4.6 ábra), amelyekkel elkülöníteni próbáltam a rendszer építését. Négy darab branch volt használatba véve: main - amelyre kerülnek a végeleges, működő verziók, amikor egy adott funkciót sikerült befejezni; authentication - amelyen az authentication feature-t készítettem el; user - ezen a felhasználói funkcióinakat és felületet írtam meg; restaurant - itt pedig az étteremnek a kódja íródott.



4.6. ábra. Branch-ek

## 4.5. Projektmenedzment

Egy nagyméretű projekt esetében elengedhetetlen, hogy projektmenedzsment eszközt használunk, hogy átláthassuk a projektünk felépítését, haladását, beoszthassuk az időket a megoldandó feladatokra, és nem utolsó sorban megjegyezzük az elvégzendő feladatokat.

Erre a Trello-t választottam, mivel már régóta ismerem, könnyű felhasználása van, akár mobilról is könnyedén elértem, ha bevillan egy ötlet a fejembe akkor bármikor használatba vehetem.

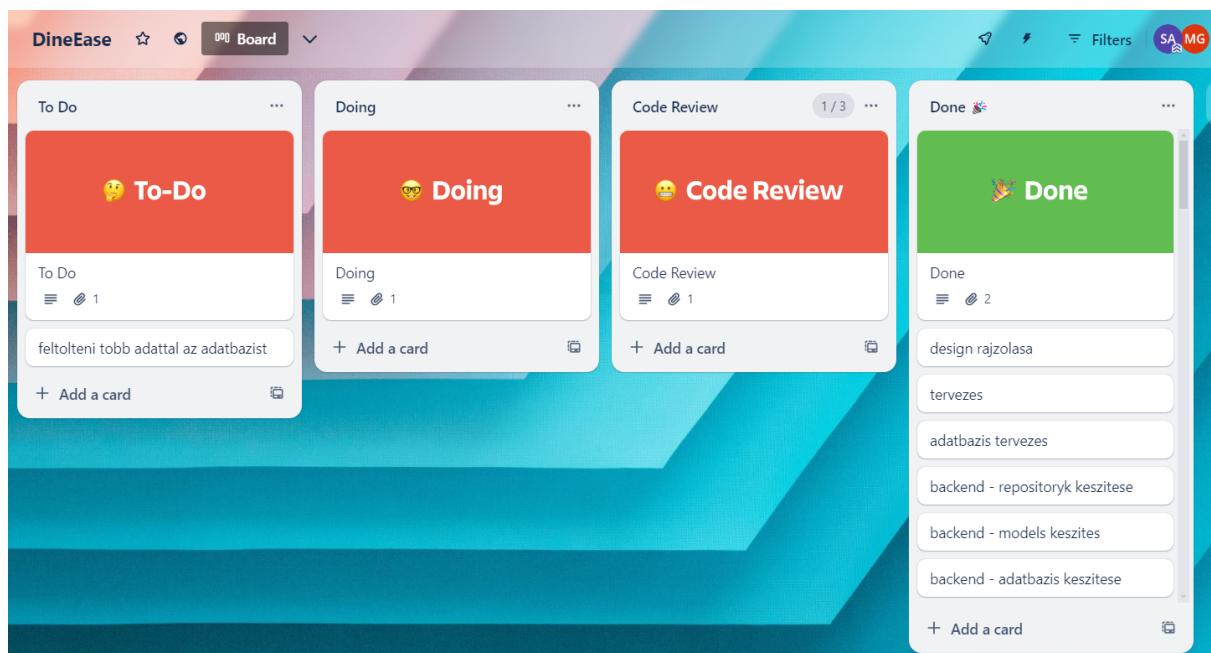
Az alább látható ábrán megtekinthető az általam használt projektmenedzsment eszköz felépítése. Négy részből épül fel: TO DO, DOING, CODE REVIEW és DONE.

A TO DO résznél jelennek meg az ötletek, az elvégzendő feladatok, itt ha sürgősnek érzem, adok határidőt is a feladatnak, hogy jobban motiváljon az elvégzésben.

A DOING részhez azokat a feladatokat illesztem be, amelyeknek nekilátok.

A CODE REVIEW résznél azok a feladatok jelennek meg, amelyek elvileg elkészültek, de még át kell menniük néhány ellenőrzésen, hogy azt mondhassam róluk, hogy igenis megvannak.

A DONE részhez pedig a sikeresen elvégzett feladatok kerülnek.



4.7. ábra. Projektmenedzsment

# 5. fejezet

## Megvalósítás

Az alkalmazás rétegelt architektúrával [LT95] volt készítve. A rétegelt architektúra egy közismert tervezési minta, amely több különálló rétegre osztja a rendszert. A DineEase alkalmazás esetében 3 réteget különböztetünk meg: *prezentációs réteg, üzleti logikai réteg, adatkezelő réteg*.

A prezentációs réteg a felhasználói felületet biztosítja, a Flutter alkalmazással, ahol a képernyők, widgetek és egyéb design elemek találhatók.

Az üzleti logikai rétegen a szolgáltatások és szabályok kerülnek előtérbe. Itt az alkalmazás logikája valósul meg.

Az adatkezelő rétegen az adatbázisra van a hangsúly fektetve, itt kezelendőek a műveletek. Ez tartalmazza a modelleket és repositorykat.

### 5.1. Szerveroldali komponensek

#### 5.1.1. API

Az API .NET keretrendszerrel lett megvalósítva. Itt igyekeztem minél átláthatóbban szervezni a projektet, így struktúra szempontjából az 5.1 ábrán látható módon szerveztem az API mappáit.

A projekt kilenc mappán belül osztódik el, amelyeknek minden megvan a saját különálló szerepük.

- **Controllers**

Ez a mappa tartalmazza azokat az osztályokat, amelyek az API végpontjainak kezelésére szolgálnak. Ezek az osztályok HTTP kéréseket fogadnak, mint például a GET, POST, UPDATE, DELETE request-ek. Ezeket az osztályokat származtatjuk az *ApiController* osztályból, amely által megvalósíthatók a kérések és válaszok küldése.

- **Data**

Ez a mappa tartalmazza a *DbContext* nevű fájlt, amely az adatbázissal kapcsolatos konfigurációkat és inicializációkat tartalmazza. Ez a fájl az *Entity Framework Core* [Mic24] ORM (Object-Relational Mapper) segítségével az adatbázis kapcsolatokat és műveleteket kezeli. Például itt adtam meg a many-to-many kapcsolatokat az adatbázishoz.

- **DTO**

A DTO mappa az adatátviteli objektumokat határozza meg, ezeket a kontrollerek használják fel az adatcserére. Ezek biztonságot nyújtanak arra, hogy az adatok struktúráltan kerüljenek átadásra, anélkül hogy a belső adatmodellek közvetlenül ki lennének téve a klienseknek. Emellett ezek segítségével egyszerűsítjük vagy bővíthetjük az adatokat, amelyeket fogadunk vagy átadunk a kliensnek.

- **Helper**

Ez a mappa azokat a segédosztályokat tartalmazza, amelyek különböző általános feladatokat látnak el. Az én esetben a *Mapper* fájl jelenik itt meg, amely segítséget nyújt arra, hogy különböző modelleket átalakítson más DTO-vá vagy modellé és fordítva.

- **Interfaces**

Ez a mappa a repository-khoz tartozó interfészeket tartalmazza. Kulcsfontosságú szerepük van, mivel így elérhetővé válik a *Dependency Injection* [RJ07], ugyanis lehetővé teszi a kód szabadabb felhasználását, implementációk lecseréléseit újraírás nélkül, olyan kapcsolatokat alakítva ki, amelyek nagyon fontosak a karbantarthatóság szempontjából.

- **Migrations**

A migrációs fájlok találhatóak itt meg, amelyek az adatbázis sémájának változtatásait kezelik, biztosítja az adatbázis naprakészségét és verziókezelését.

- **Models**

Ebben találhatóak az adatbázis tábláira szolgáló adatmodellek. Ezek tükrözik az adatbázis struktúráját és kezelik a műveleteket *Entity Framework* segítségével.

- **Repository**

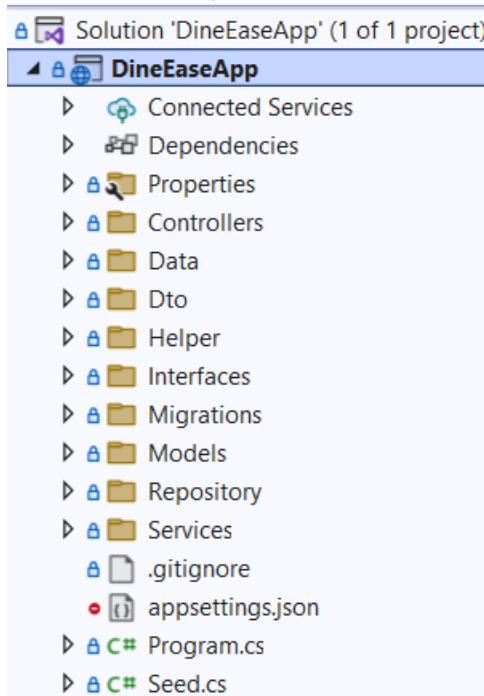
Itt vannak megvalósítva az adatbázisműveletek, itt valósul meg az adatok kezelése és elérése, segítséget nyújt a kód szétválasztásában és szervezésében. minden táblához külön osztály tartozik. Viszont van köztük egy *generic repository*, amely a kód újrafelhasználhatóságát nyújtja, ez az osztály rendelkezik azokkal a műveletekkel, amelyeket gyakran használunk, így egy származtatással könnyedén használhatjuk ezt az osztályt is.

- **Services**

Ez a mappa tartalmazza azokat a fájlokat, ahol megvalósul az üzleti logika. A *Di-  
neEase* alkalmazás esetében a fényképek tárolásának és kezelésének logikája valósul itt meg.

A *Program.cs* fájlban valósultak meg a szolgáltatások konfigurálásai.

Az *AddControllers*) segítségével az *MVC* szolgáltatásokat alkalmazhattuk, és lehetővé tettük az API végpontok kezelését vele. A *Dependency injection*-höz az *AddTransient* és *AddScoped* metódusokat hívtuk segítségül, amelyek a szolgáltatásokat regisztrálják a függőséginjektáló rendszerben.



**5.1. ábra.** API mappa struktúra

A *JWT tokeneket* az *AddAuthentication* metódus használatával szolgáltatjuk, míg a token validációra a *TokenValidationParameters* metódust használtam, amely az érvényességet és hitelességet állítja be elsősorban.

Itt konfiguráljuk az SQL szervert is, illetve a *CORS* (Cross Origin Resource Sharing) konfiguráció is kulesfontosságú mozzanat volt, ugyanis az alkalmazás webes és mobil interfész közötti kommunikációt ezzel lehetett biztosítani.

### 5.1.2. Adatbázis

Az adatbázisrendszer egy nagyon fontos komponens, amely a felhasználói, éttermi adatok és részletek tárolását, lekérdezését, kezelését biztosítja. Ennek struktúrája *Entity Framework Core, Code first* megközelítéssel lett megalkotva.

Az *ORM*-nek köszönhetően nem kell direkt SQL lekérdezéseket írni az adatbázishoz, hanem az általa létrejött absztrakció segítségével az adatbázis magasabb szintűvé válik és objektumorientált lesz. Ez hozzájárul a kód karbantartásához és átláthatóságához, ugyanis minden táblát egy objektumként kezelünk. A *Code first* megközelítés során létrehozhatjuk az osztályokat, amelyek a táblák lesznek, így definiáljuk a kapcsolatokat a táblák között is (lásd 5.2 ábra). Itt kulcsfontosságú az *adatbázis migráció* is, amelyet az Entity Framework migrációs parancsaival generáljuk ki, ezáltal lehet módosításokat és újításokat belevinni a táblákba.

Az ASP .NET Core biztosítja azt is, hogy az adatbázist inicializáljuk valamilyen módon, ez az esetünkben a Seed fájlon belül valósul meg, ahol SQL kód nélkül könnyedén feltölhetjük a tábláinkat kezdeti adatokkal.

```

using System.ComponentModel.DataAnnotations.Schema;

namespace DineEaseApp.Models
{
    public class Event
    {
        public int Id { get; set; }
        public string EventName { get; set; }
        [ForeignKey("Restaurant")]
        public int RestaurantId { get; set; }
        public Restaurant Restaurant { get; set; }
        public string? Description { get; set; }
        public DateTime StartingDate { get; set; }
        public DateTime EndingDate { get; set; }
        public ICollection<CategoriesEvent> CategoriesEvents { get; set; }
    }
}

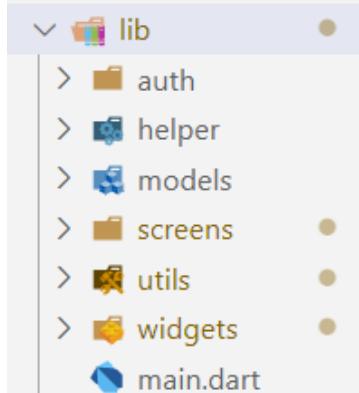
```

**5.2. ábra.** Példa tábla létrehozására

## 5.2. Kliensoldali komponens

Az alkalmazás Flutter keretrendszerben készült el, amely cross-platform támogatást és gyors, natív teljesítményt nyújt a felhasználók számára. A projekt felépítésében különböző funkcionálitások szerint voltak készítve és elrendezve a mappák, amely segíti a kód átláthatóságát és karbantarthatóságát.

Az **5.3** ábrán látható a projekt felépítése, ahogyan a különböző komponenseket elhártoltam egymástól, itt hat mappára oszlott fel a rendszer.



**5.3. ábra.** Flutter mappa struktúra

- **Auth**

Ez a mappa tartalmazza az autentikációs logikát, a bejelentkezés és a regisztrációhoz. Itt valósul meg a tokenek biztonságos eltárolása a *Shared Preferences* csomag használatával, ez az osztály tartalmazza a felhasználói hitelesítést.

- **Helper**

Itt tarolódnak el a segédosztályok, például a *ChangeNotifier*ek, amelyek segítségével értesíteni lehet a felhasználói felületet és a listenereket, arról hogy frissült az adott komponens.

- **Models**

Az adatmodellekkel lehet találkozni ebben a mappában, ez szolgáltatja az alkalmazás üzleti logikáját. Itt vannak megírva a konvertálások is, amelyeket az API-val való kommunikáció során használunk, *toMap* és *toJson* függvényekkel. A *toMap* használatos a GET requestek alkalmazásánál, ez alakítja át az adott JSON formátumot modellé, változóit pedig a meghatározott típusokká alakítva. A *toJson* pedig olyankor használatos, amikor az APInak kell küldeni adatokat, például egy POST requestet, ez segít átalakítani a modell típusait JSON típusokká.

- **Screens**

Ez a mappa tartalmazza a különböző képernyőket és a UI tervezeteket, függvényeit. minden képernyő egy külön fájl, külön osztályként van kezelve, amelyek kezelik a felhasználói interakciókat és adatokat. Itt összesen 30 darab képernyő jött létre. Ezek a képernyők rendelkeznek route nevekkel, amelyek biztosítják a képernyők közötti egyszerű navigációt.

- **Utils**

Ez a mappa, a nevéről adódóan is megállapítható, hogy a hasznos eszközöket tárolja, amelyeket gyakran használhatunk fel. A projekt esetében itt az API requestek találhatóak. Itt az volt a metódus, hogy egy *.env* fájlból eltároltam az endpoint elérhetőségét, ez természetesen a *.gitignore* fájlból megjelent, tehát az endpointokat privát módon kezeltük. Ezután pedig függvények segítségével íródtak meg a fetch-elések.

- **Widgets**

Itt újrahasznosítható, általam designolt UI elemeket tárolunk, amelyeket több képernyőn fel lehet használni. Ezek támogatják az egységes kinézetet és a kód újrahasznosíthatóságát.

Az alkalmazás során felhasználtam néhány fontosabb könyvtárcsomagot. Ezeknek a függőségeknek a beimportálására a *pubspec.yaml* fájlt használja a Flutter, a dependencies kulcsszó alatt. A *Provider* állapotkezelést kínál, amely lehetővé teszi a különböző komponensek közötti adatmegosztást és a UI automatikus frissítését is adatok változásakor, a *shared preferences* is fel volt használva, ez kulcs-érték párokat tárol helyileg a felhasználó eszközén, a mi esetünkben tokenek tárolására lett alkalmazva és a felhasználók adatainak tárolására. A *logger* segítségével történt meg a naplózás a projektben, amellyel hibákat,

információkat naplóztunk ki, ez hasznos volt a fejlesztés és hibakezelés során. A *jwt decoder* csomag biztosítja a JSON web tokenek dekódolását, illetve ez segít a tokenekben tárolt adatok eléréséhez.

Ezen felül a widgetekhez is használtunk néhány könyvtárcsomagot, ezek között a fontosabbak: *image picker*, *carousel slider*, *flutter datetime picker*, *syncfusion flutter calendar*.

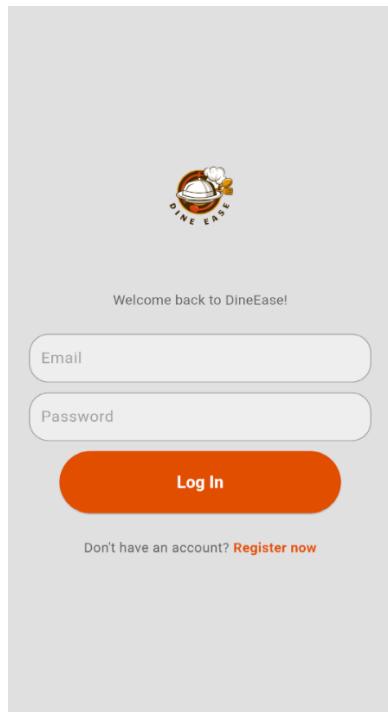
## 6. fejezet

### Felhasználói felület

Felhasználói felületet tekintve, igyekeztem egy letisztult, nem túl zsúfolt, egyszerű és átlátható felületet biztosítani. Az alkalmazás meghatározó színe a narancssárga. A DineEase alkalmazásban két szerepkörbe tartozó felhasználó létezik: étterem és átlag felhasználó. Ezeknek két különböző felület jött létre, viszont egy dolog megegyezik, a bejelentkezés és regisztráció képernyők.

#### 6.1. Belépés és regisztráció

Ahogy megnyitja az alkalmazást a felhasználó, egy splash screen megjelenése után, ha nem volt előzőleg bejelentkezve még, vagy lejárt a bejelentkezési ideje (tokenje), következik az a képernyő, ahol kiválaszthatja, hogy bejelentkezni szeretne vagy új fiókot szeretne létrehozni.



6.1. ábra. Bejelentkezés oldal

A Login gombra kattintva, elvezeti a bejelentkezéshez a felhasználót, ahol megadhatja adatait. Ha még nincs fiókja, átnavigálhat a Register now szövegre kattintva a regisztrációs felületre. Itt bármelyik szerepkörbe tartozó felhasználó be tud jelentkezni, ugyanis felismeri az alkalmazás, hogy milyen típusú felhasználó szeretne bejelentkezni. Ez látható a 6.1 ábrán.

A Sign up gombot választva a regisztrációs oldallal találkozik a felhasználó. Itt ki-választhatja, milyen szerepkörbe tartozó fiókot szeretne létrehozni, és aszerint tölti ki a szükséges adatokkal a mezőket. A mezők ellenőzik, hogy megfelelő formátumú adatot adott-e meg a regisztráló. Természetesen itt is van olyan opció, hogy átlépjenek a bejelentkezéshez, egy szövegre kattintva. A regisztrációs oldalak a 6.2 ábrán láthatóak.

The image contains two side-by-side screenshots of a mobile application's registration interface. Both screens feature a header with the Dine Ease logo at the top center. Below the logo, there are two tabs: 'User' (highlighted in orange) and 'Restaurant' (highlighted in grey). The left screenshot, labeled 'User', contains six input fields: 'First Name', 'Last Name', 'Email', 'Phone', 'Password', and 'Confirm Password'. Below these fields is a large orange 'Sign Up' button. At the bottom of the screen, there is a link in red text that says 'Do you have an account? Log in'. The right screenshot, labeled 'Restaurant', contains eight input fields: 'Name', 'Email', 'Phone', 'Description', 'Address', 'Max Table Capacity', 'Tax ID Number', and 'Password'. Below these fields is a large orange 'Sign Up' button. There is no 'Confirm Password' field for the Restaurant registration.

**6.2. ábra.** Regisztrációs oldalak

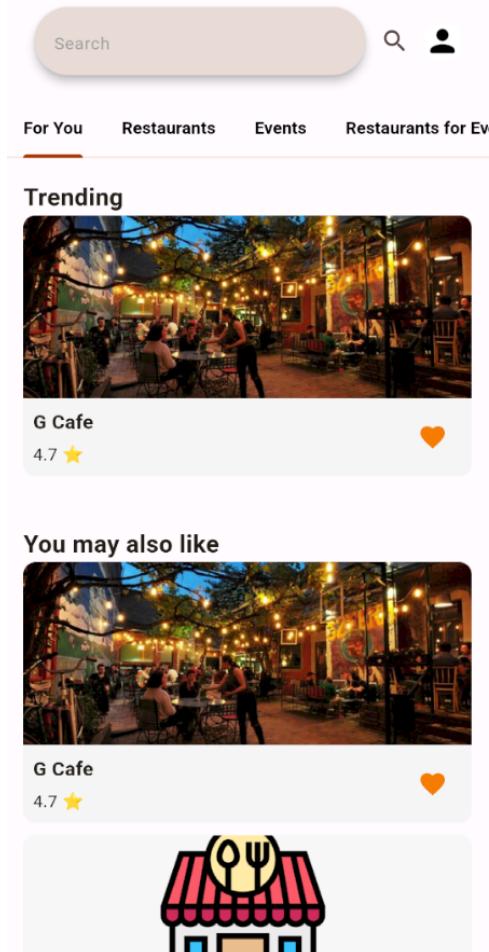
## 6.2. Felhasználó

A 6.3 ábrán észrevehető, hogy egy navigációs sáv segítségével lehet váltogatni a négy oldal között: *For You, Restaurants, Events, Restaurants for Events*. Itt használatba vehető a keresés funkció is, illetve az ember ikonra kattintva a felhasználó a *profil* oldalára is navigálhat.

### 6.2.1. For you oldal

A bejelentkezett felhasználó először a *For You* nevű oldallal találkozhat, amelyet a 6.3 ábrán lehet látni. Itt ajánlásokkal találkozhat a felhasználó, négy szekcióra felosztva. A *Trending* címszó alatt jelenik meg az a három étterem, amely az utóbbi hét napban a legtöbb foglalással rendelkezett. A *You may also like* címszó alatt jelenik meg a három legnagyobb értékeléssel rendelkező étterem. A *Recommended events* alatt olyan eseményeket ajánl az alkalmazás a felhasználónak, amelyeket olyan éttermek szerveznek, amiket kedvencnek jelölt be a felhasználó, ha nincs kedvence bejelölve, akkor egyszerűen néhány random eseményt ajánl. A *Did you like* címszó nem mindenkinél jelenik meg, itt olyan éttermek láthatóak, amelyeket már meglátogatott a felhasználó, azaz a foglalásai szerint vannak listázva.

Az éttermek megjelenítésénél látható egy *kép*, a *neve*, *értékelése* és egy szív alakú ikon, amely megnyomásával lehet az adott éttermet *kedvencnek* megjelölni, vagy épp kitörölni a kedvencek közül.



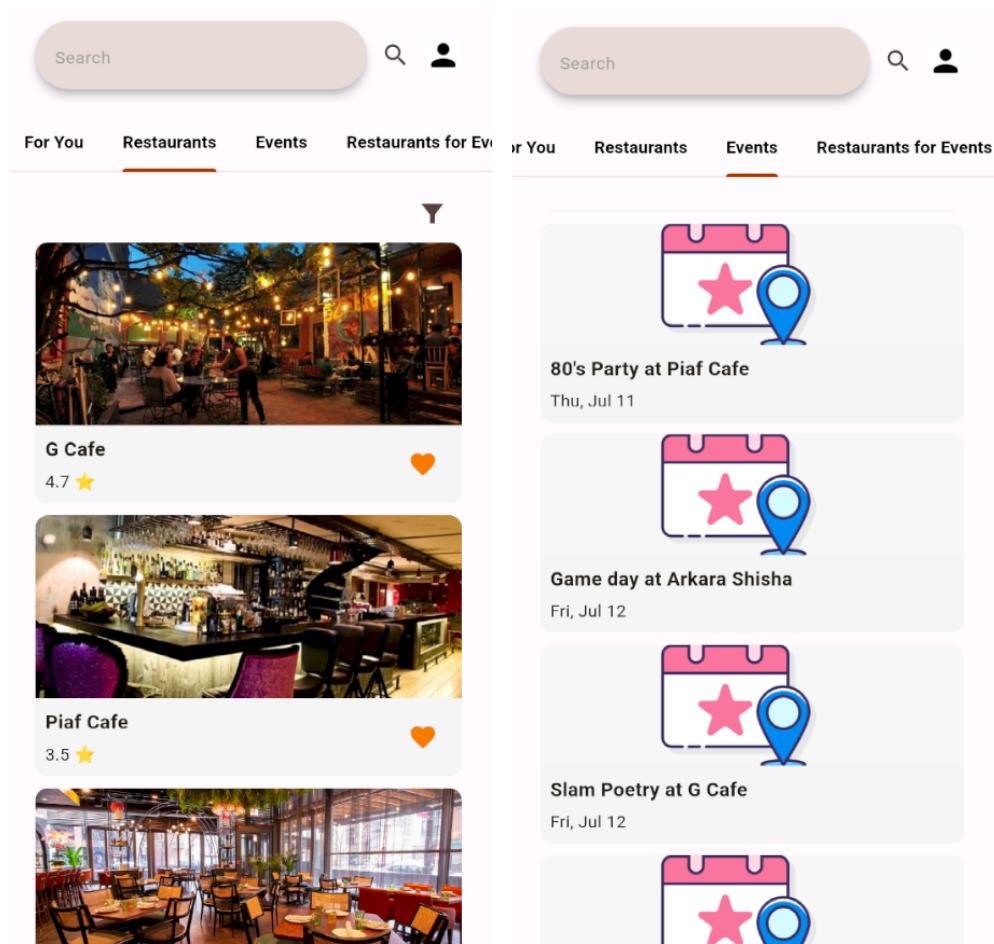
6.3. ábra. For you oldal

## 6.2.2. Keresés

A 6.3 ábrán látható felül a kereső, ahol lehetőség van rákeresni éttermekre és eseményekre a beírt karakterek alapján. A kereső az éttermek és események nevében és leírásukban keresi a karaktereket és küldi vissza azok listáját.

## 6.2.3. Restaurant és Event oldal

A 6.4 ábrán látható a Restaurant és Event oldal, ahol ki van listázva az összes étterem és az összes előkövetkező esemény, amelyeket szerveznek az éttermekben. Ugyanígy néz ki a Restaurants for Events oldal is, ahol azok az éttermek vannak kilistázva, ahol nagyobb eseményeket lehet szervezni. A Restaurants oldalnak a jobb felső sarkában található egy ikon, amely segítségével szűrhetjük különböző tulajdonság szerint az éttermeket.



6.4. ábra. Restaurant és Event oldalak

#### 6.2.4. Éttermekhez és eseményekhez kapcsolódó funkciók

Ha kiválaszt egy éttermet, megtekintheti annak részletes információit, ez látható a 6.5 ábrán. Itt több funkció is elérhető. A szív alakú ikonra kattintva *kedvencnek jelölheti* be az éttermet. A Rating címszó alatt értékelheti az éttermet, vagy törölheti értékelését a felhasználó. *Visszajelzést* is írhat az étteremről a felhasználó, vagy épp törölheti is a saját megjegyzését, és megtekintheti itt másokét is. A Show menu gomb lenyomásával megtekinthető az étterem által kínált menüsor.

Az események is hasonlóképpen jelennek meg, ott csak a részleteit lehet megtekinteni és ilyen funkcionálisokkal nem rendelkeznek, viszont ott is megjelenik a Reserve a table gomb, amellyel asztalt lehet foglalni az esemény dátumára.

The screenshot displays a restaurant profile for "G Cafe". The top left shows a thumbnail of the restaurant's exterior at night. The top right shows the opening hours from Tuesday to Saturday. Below that is a "Rating" section with a 4.7 star average and two reviews. At the bottom are buttons for "Show menu" and "Reserve a table".

**G Cafe**

Strada Cuza Vodă 33, Târgu Mureş 540027

Categories: fast food pub

Cuisines: Italian transylvanian

Seating: teras bar

average pricing

Description: A good placez

Opening hours:

Day	Opening Hours
Tuesday	8:00 - 21:00
Wednesday	8:00 - 21:00
Thursday	8:00 - 21:00
Friday	9:00 - 23:00
Saturday	9:00 - 22:00
Sunday	9:00 - 21:00

Rating: ★★★★☆

2 reviews

Write your review here...

string string  
it was a good experience

test test  
my favorit restaurant.....

Show menu

Reserve a table

6.5. ábra. Étterem részleteinek oldala

### 6.2.5. Foglalás és meeting programálás

Mivel két fajta étterem kategóriát különböztetünk meg az alkalmazásban, így két különböző funkcionalitást is lehet elérni.

A Restaurant for Event kategóriából kiválasztott éttermekbe meetinget lehet szerzézní, ez a 6.6 ábrának a jobb oldalán látható, ahogyan különböző adatok megadásával egy személyes meetinget szervez a részletek megbeszélésére az étterem tulajdonosával.

A Restaurants kategóriából választott étterembe foglalást lehet letenni, ami a 6.6 ábra bal oldalán tekinthető meg. Itt lehet a Menu gombra kattintva előrendelni az ételt a foglaláshoz, ezek a megrendelt ételek megjelennek a foglalás oldalán is, ahol az összeget is írja, amennyit fizetnie kell majd, viszont ha úgy dönt, hogy mégsem szeretné azt a menüt választani, kitörölheti egy gomb lenyomásával.

The screenshot displays two side-by-side booking forms. The left form is for 'Reserve at G Cafe' and the right form is for 'Meeting at new'. Both forms have orange header bars with back arrows and titles.

Reserve at G Cafe	Meeting at new
Party Size 6	Event Date 2024-10-10
Date 2024-06-28	Event Time 11:00
Time 18:30	Guest Size 150
Phone Number 0711111111	Meeting Date 2024-07-12
Comment	Meeting Time 10:00
<b>Your orders: 20.0 RON</b>	
<b>Salad box</b> 20 RON	<b>X</b>
<b>Menu</b>	
<b>Reserve</b>	<b>Schedule Meeting</b>

6.6. ábra. Foglalás és meeting programálás oldalak

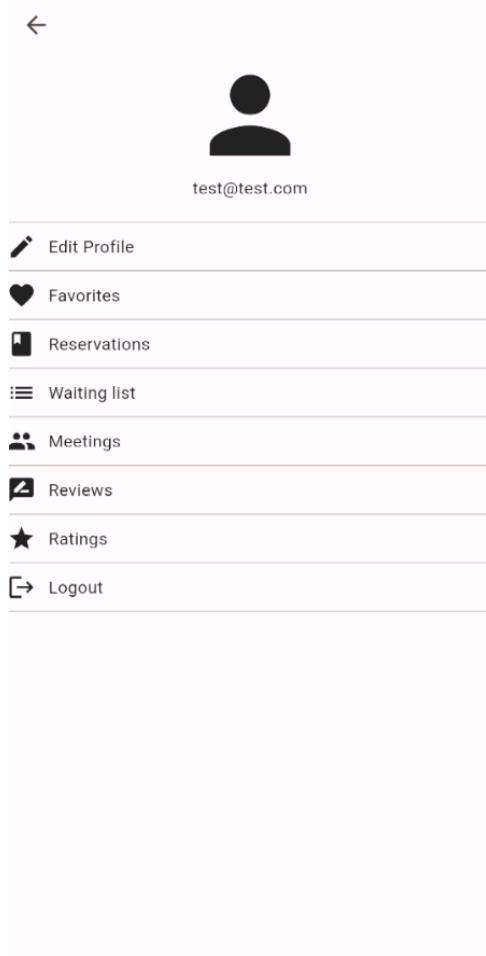
## 6.2.6. Profil oldal

A 6.3 ábrán látható a jobb felső sarokban egy profil ikon, erre kattintva érhető el a Profil oldal, amelyen a felhasználóval kapcsolatos információk érehetőek el, ezeknek a választéka a 6.7 ábrán látható.

Az *Edit profile* gombra kattintva a felhasználó a saját adatait szerkesztheti.

A *Favorites* gombra kattintva megjelenik az összes étterem listája, amelyet a felhasználó kedvencnek jelölt, itt ki is lehet őket törlni a kedvencek közül.

A *Reservations* gombbal megtekintheti azokat a foglalásait, amelyekre kapott választ az éttermektől, hogy elfogadták őket vagy sem. Itt elsősorban az elkövetkező foglalások jelennek meg, utána pedig egy History a régi foglalások listájával.



6.7. ábra. Profil oldal

## 6.3. Étterem

Az éttermek mivel két kategóriába sorolhatók, így néhány eltérő funkció is látható, de összességében nagyon hasonlóak.

### 6.3.1. Foglalás és Meeting oldal

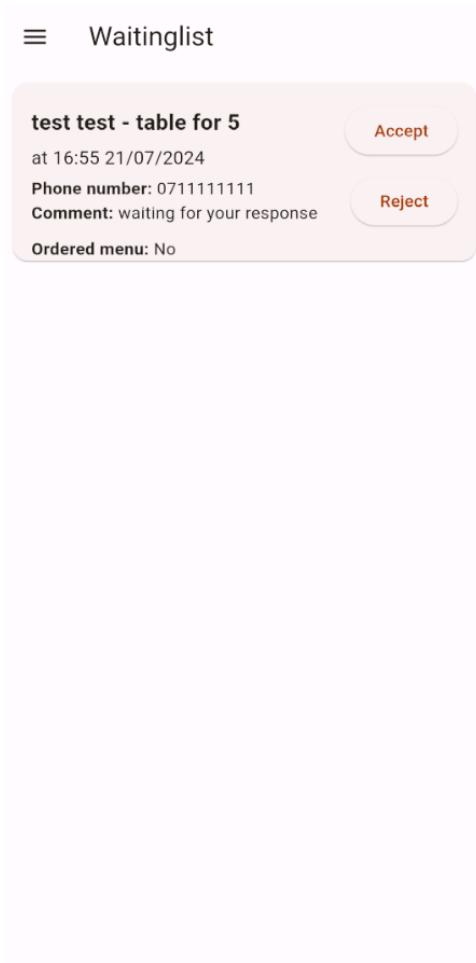
A 6.8 ábrán látható, hogyan jelennek meg az étterem foglalásai . Itt az elfogadott foglalásai láthatóak. Hárrom különféle nézetben tudja megtekinteni ezeket: havi, heti és napi. A napi nézetben, ha rákattint egy adott foglalásra, akkor láthatja a foglalás részleteit. Ha rendelt is a foglaló személy, akkor a *See orders* gombra kattintva meg lehet tekinteni a rendeléseit is. Hasonlóképpen jelennek meg a meetingek is.

The image shows two screenshots of a software application for managing reservations and meetings. The left screenshot displays a monthly calendar for June 2024. The days of the week are labeled M, T, W, T, F, S, S. Specific dates are highlighted: June 23rd is circled in red, and June 5th has a small blue double-dot icon above it. The right screenshot shows a detailed view of a reservation for June 5th at 10:52 AM. The reservation is for a table of 10 people at Piaf Cafe, with the name 'test test', party size 10, phone number 079999999, table number 3, and a comment 'string'. The 'Ordered' status is marked as 'Yes' with a 'See orders' link. The reservation is shown as a teal block from 11 AM to 1 PM, and the meeting is shown as a dark blue block from 1 PM to 3 PM. A 'Close' button is visible in the top right corner of the reservation details.

6.8. ábra. Foglalások megtekintése

### 6.3.2. Waitinglist oldal

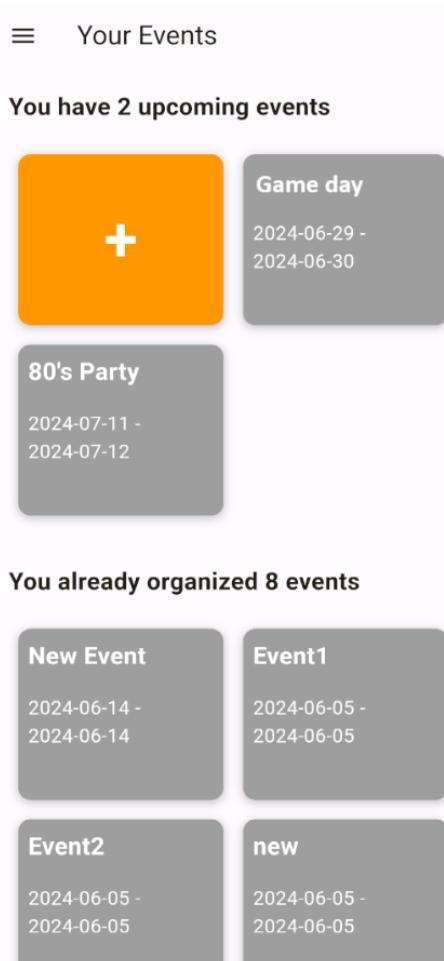
A waitinglist-be a válaszra váró foglalások és meetingek kerülnek be. Itt átnézheti őket az étterem és elfogadhatja vagy éppen elutasíthatja őket az Accept és Reject gombokkal. Ez látható a 6.9 ábrán.



6.9. ábra. Waitinglist oldal

### 6.3.3. Your Events oldal

Az eseményeket, amelyeket már megszerveztek az étteremben vagy éppen szerveznek, a menüsor *Your events* gombjára kattintva érheti el. Itt a régi eseményekre kattintva megtekinthetők a részleteik, az elkövetkező eseményekre kattintva pedig szerkeszteni is lehet őket. Ezen az oldalon lehet új eseményeket is meghirdetni a plusz gombra kattintva. Ez látható a 6.10 ábrán.



6.10. ábra. Your events oldal

### 6.3.4. Your Menu oldal

Az étteremnek lehetősége van menüket is hozzáadni, ehhez létrejött az az oldal, amelyet a menüsor *Your Menu* gombjára kattintva érhet el. Itt hozzáadhat új menüket és szerkesztheti a meglévőket, illetve törölheti is őket.

### 6.3.5. Restaurant oldal

Az étterem a saját információit szerkesztheti a *Restaurant* gombra kattintva a menüből. Ez az oldal látható a 6.11 ábrán. Itt megadhat frissebb információkat az étteremről és képeket töölhet fel vagy törölhet.

### ≡ Restaurant

#### Edit Restaurant Profile

Name —

Address —

Phone Number —

Email —

Description —

Max Table Capacity —

**Price category** Expensive ▾

**Update**

#### Photos



### ≡ Restaurant



**Add Photo**

#### Edit Restaurant Details

##### Openings

Monday	7:00 - 20:00
Tuesday	Closed
Wednesday	7:00 - 20:00
Thursday	7:00 - 20:00
Friday	7:00 - 20:00
Saturday	12:30 - 13:40
Sunday	7:00 - 20:00

**Edit Openings**

##### Cuisines

**+ Add Cuisine**

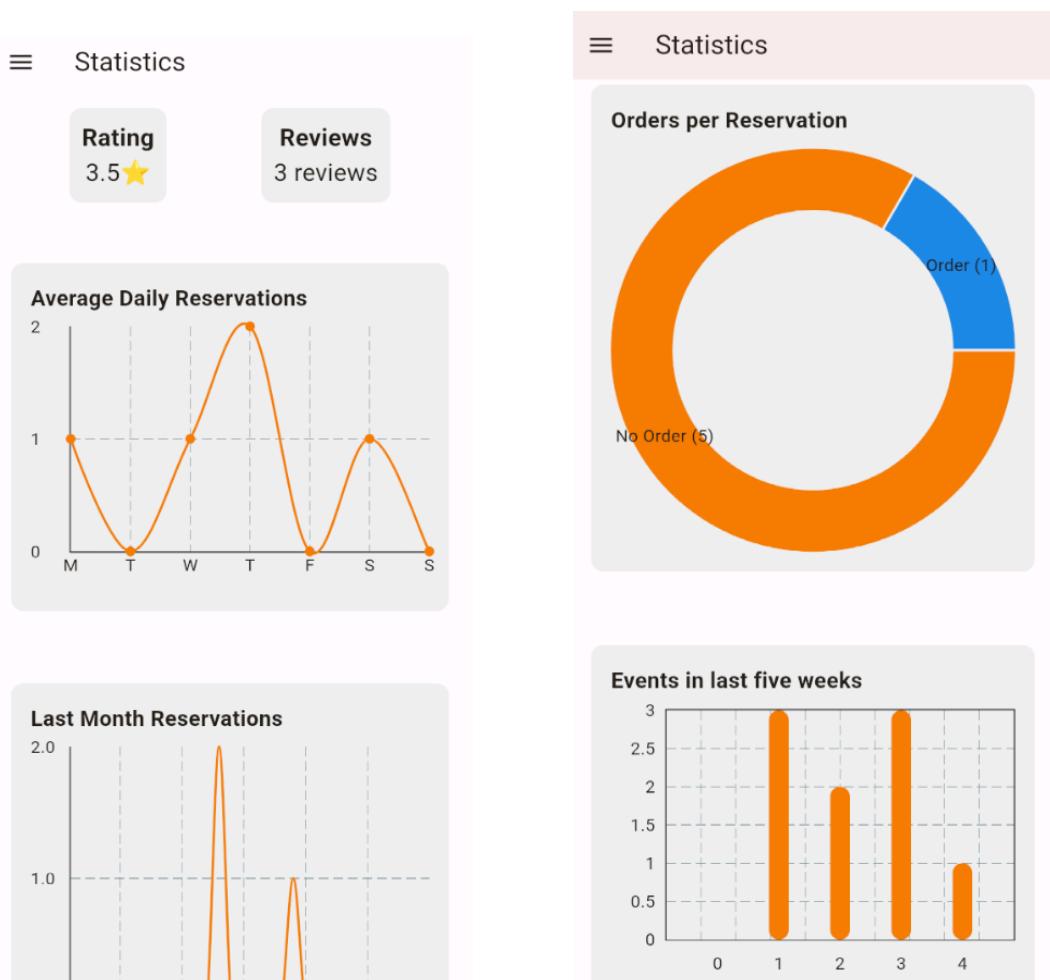
##### Categories

**+ Add Category**

6.11. ábra. Étterem profil oldal

### 6.3.6. Statistics oldal

Az étteremről készül néhány statisztikai mérés is, ami ezen az oldalon érhető el. Ha a *for event* kategóriába sorolandó étterem információit vizsgáljuk, akkor ezen az oldalon jelenik meg az átlag napi meetingek statisztikája, az elmúlt hónap meetingjeinek statisztikája, egy statisztika arról, hogy melyik órában a legforgalmasabb az étterem, és az elmúlt öt hét eseményeinek száma oszlopdiagrammal. Az átlag éttermek annyiba különböznek ettől, hogy mérve van a rendelések száma is foglalásonként, illetve az előzőekben felsoroltakban nem a meetinget, hanem a foglalásokat méri. Ugyanakkor itt látható az étterem értékelése is és megtekinthetők a visszajelzések is. Ezeket a visszajelzéseket itt törölni is lehet, ha nem tetszene a tartalma.



6.12. ábra. Statisztikák oldal

# Összefoglaló

A DineEase projekt egy modern asztalfoglalórendszer megvalósításával foglalkozik, amely megoldást nyújt úgy a foglalni vágyó embereknek, mint az éttermeknek a foglalások egyszerűbb kezelésére. Lehetőséget nyújt az alkalmazás a felhasználói vélemények megnyilvánulásának is, amely alapján könnyebben dönthet az éttermek között. Ugyanakkor az éttermeknek is egy olyan felületet biztosít, amelyen egyszerűen kezelheti foglalásait, vagy akár meetingjeit és eseményeit is könnyedén hirdetheti, illetve elemzéseket is végezhet a foglalások alapján. Hozzájárul az olyan éttermeknek az eléréséhez is, amelyek nagyobb eseményekre specializálódottak, így könnyebbé téve az éttermekkel való kommunikációt.

A dolgozatom során ismertettem a hasonló alkalmazásokat és összemértem azokat a DineEase alkalmazásal funkcionálitás és technológia szempontjából is. A biztonsági lépéseket is leírtam dolgozatom során, amellyel hozzájárulhattam a felhasználók adatainak védelméhez. Majd a funkcionálisokat is tárgyaltam, amelyek megvalósításra kerültek, illetve a funkcionális és nem funkcionális rendszerkövetelményekben is megfogalmaztam, hogy mik az elvárások a rendszerrel kapcsolatban. Ezekután tárgyaltam, hogyan épül fel a rendszer, amely egy .NET keretrendszerrel készített API-ból, egy MSSQL adatbázisból és Flutter keretrendszerrel készült frontend-ből tevődik össze. Itt bemutatásra került a verziókövető rendszer és a projektmenedzsment eszköz is. A megvalósítás részben tárgyalva van, hogyan épülnek fel a különböző komponensek. Mindezek után pedig a felhasználói felület is bemutatásra került.

Összességében úgy gondolom, sikerült egy jól használható alkalmazást fejleszteni, amely könnyebbséget jelenthet a felhasználók életében. Ugyanakkor több különböző tervvel járulnék hozzá az alkalmazás további fejlesztéséhez:

- **Google térkép:** ezzel hozzájárulva a könnyebb helymeghatározáshoz, esetleg legközelebbi hely szerinti kereséshez.
- **Kép feltöltés:** a menü, esemény és a felhasználó táblában helyet biztosítani a képeknek, amelyek által jobb élményt nyújthatunk a felhasználóknak.
- **Meghívók küldése:** a foglalásban megemlítve azokat a felhasználókat, akikkel tervez az étterembe menni, meghívók küldődnének ki, hogy emlékeztessék őket.
- **Értesítések:** a foglalás elfogadásáról vagy elutasításáról, ajánlott eseményekről, esetleg a foglalás közeledtérről szóló értesítések küldése.

A részletes munkafolyamat elérhető a következő GitHub és Trello linken:

- <https://github.com/aksza/DineEase>
- <https://trello.com/b/FlzsShfn/dineease>

# Ábrák jegyzéke

2.1. Tokenizálás . . . . .	17
2.2. Jelszó hash-elés . . . . .	18
3.1. Felhasználói use-case diagram . . . . .	21
3.2. Étterem use-case diagram . . . . .	23
3.3. Asztalfoglalás szekvencia diagram . . . . .	27
4.1. Rendszer architektúrája . . . . .	30
4.2. Adatbázis terv . . . . .	32
4.3. Felhasználó felület terv . . . . .	34
4.4. Étterem felület terv . . . . .	35
4.5. Verziókövető rendszer . . . . .	36
4.6. Branch-ek . . . . .	36
4.7. Projektmenedzsment . . . . .	37
5.1. API mappa struktúra . . . . .	40
5.2. Példa tábla létrehozására . . . . .	41
5.3. Flutter mappa struktúra . . . . .	41
6.1. Bejelentkezés oldal . . . . .	44
6.2. Regisztrációs oldalak . . . . .	45
6.3. For you oldal . . . . .	46
6.4. Restaurant és Event oldalak . . . . .	47
6.5. Étterem részleteinek oldala . . . . .	48
6.6. Foglalás és meeting programálás oldalak . . . . .	49
6.7. Profil oldal . . . . .	50
6.8. Foglalások megtekintése . . . . .	51
6.9. Waitinglist oldal . . . . .	52
6.10. Your events oldal . . . . .	53
6.11. Étterem profil oldal . . . . .	54
6.12. Statisztikák oldal . . . . .	55

# Irodalomjegyzék

- [AM19] Salman Ahmed and Qamar Mahmood. An authentication based scheme for applications using json web token. In *2019 22nd international multitopic conference (INMIC)*, pages 1–6. IEEE, 2019.
- [Cor24a] Microsoft Corporation. Asp.net core documentation. <https://learn.microsoft.com/en-us/aspnet/core/?view=aspnetcore-7.0>, 2024. Látogatás dátuma: 2024-06-03.
- [Cor24b] Microsoft Corporation. Microsoft sql server documentation. [https://learn.microsoft.com/en-us/sql/?view\(sql-server-ver16](https://learn.microsoft.com/en-us/sql/?view(sql-server-ver16), 2024. Látogatás dátuma: 2024-06-03.
- [HO06] Jane Huffman Hayes and Jeff Offutt. Input validation analysis and testing. *Empirical Software Engineering*, 11:493–522, 2006.
- [LLC24] Google LLC. Flutter documentation. <https://docs.flutter.dev/>, 2024. Látogatás dátuma: 2024-06-03.
- [LT95] Peter Loborg and A Torne. A layered architecture for real time applications. In *Proceedings Seventh Euromicro Workshop on Real-Time Systems*, pages 11–16. IEEE, 1995.
- [Mic24] Microsoft. Entity framework documentation. <https://learn.microsoft.com/en-us/ef/>, 2024. Látogatás dátuma: 2024-06-06.
- [RJ07] Ekaterina Razina and David S Janzen. Effects of dependency injection on maintainability. In *Proceedings of the 11th IASTED International Conference on Software Engineering and Applications: Cambridge, MA*, page 7, 2007.
- [RSB23] Palash Rambhia, Parth Shinde, and Kalyan Bamane. Securing flutter applications: A comprehensive study. In *2023 7th International Conference On Computing, Communication, Control And Automation (ICCUBEA)*, pages 1–5. IEEE, 2023.
- [Sta11] Mark Stamp. *Information security: principles and practice*. John Wiley & Sons, 2011.
- [tea23] Hive team. Unlocking the power of javascript: A guide to its benefits. *Devs-Hive Insights*, May 15 2023. Látogatás dátuma: 2024-06-03.

- [Tea24] Flutter Dev Team. Flutter supported platforms. <https://flutter-ko.dev/development/tools/sdk/release-notes/supported-platforms>, 2024. Látogatás dátuma: 2024-06-05.
- [ZCSC19] Yi Zeng, Jinfu Chen, Weiyi Shang, and Tse-Hsun Chen. Studying the characteristics of logging practices in mobile apps: a case study on f-droid. *Empirical Software Engineering*, 24:3394–3434, 2019.