SAPIENTIA ERDÉLYI MAGYAR TUDOMÁNYEGYETEM
MAROSVÁSÁRHELYI KAR,
INFORMATIKA SZAK

SAPIENTIA
ERDÉLYI MAGYAR
TUDOMÁNYEGYETEM

A dolgozat címe

TastyGo: Alkalmazás egy éttermen belüli házhozszállításra

Témavezető:
Dr. Jánosi-Rancz Katalin-Tünde,
Egyetemi adjunktus

Végzős hallgató:
Kiss Krisztina Gyöngyvér

2023

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

UNIVERSITATEA
SAPIENTIA

Titlul lucrării

TastyGo: Aplicatie mobila pentru livrari la domiciliu pentru un restaurant

Coordonator științific:
Dr. Jánosi-Rancz Katalin-Tünde,
Lector universitar

Absolvent:
Kiss Krisztina Gyöngyvér

2023

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

SAPIENTIA
HUNGARIAN UNIVERSITY
OF TRANSYLVANIA

Title of the Bachelor thesis

TastyGo: Application for a restaurant with delivery

Scientific advisor:
Dr. Jánosi-Rancz Katalin-Tünde,
Lecturer

Student:
Kiss Krisztina Gyöngyvér

2023

Declaraţie

Subsemnatul/a KISS KRISZTINA GYÖNGYVÉR absolvent(ă) al/a specializării INFORMATICA, promoţia 2023, cunoscând prevederile Legii Educaţiei Naţionale 1/2011 şi a Codului de etică şi deontologie profesională a Universităţii Sapientia cu privire la furt intelectual declar pe propria răspundere că prezenta lucrare de licenţă/proiect de diplomă/disertaţie se bazează pe activitatea personală, cercetarea/proiectarea este efectuată de mine, informaţiile şi datele preluate din literatura de specialitate sunt citate în mod corespunzător.

Localitatea, Târgu Mureş
Data: 16.06.2023

Absolvent

Semnătura.....

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

LUCRARE DE DIPLOMĂ

Coordonator științific:	Candidat: **Kiss Krisztina Gyöngyvér**
dr. Jánosi Rancz Katalin Tünde	Anul absolvirii: 2023

a) Tema lucrării de licență: Proiectarea si implementarea unei aplicații mobile pentru preluare comenzi. Crearea unei aplicații pentru un restaurant cu livrare la domiciliu.

b) Problemele principale tratate: Proiectarea interfeței de utilizator (UI), implementarea funcționalităților de bază, cum ar fi afișarea meniului, adăugarea produselor în coșul de cumpărături și plasarea comenzii, plată online, gestionarea comenzilor primite și actualizarea stării acestora în timp real, integrarea serviciilor de localizare

c) Desene obligatorii: diagramă de cazuri de utilizare, diagramă de clasă, diagramă de secvență

d) Softuri obligatorii: Mediul de dezvoltare: Android Studio, Java Development Kit, Firebase. Aplicația este bazată pe tehnologii Java care utilizează Firebase Realtime Database pentru stocarea și gestionarea datelor în timp real, bazat pe modelul JSON
-utilizatorii pot căuta meniuri, plasa comenzi, efectua plăți
-aplicația include integrarea Google Maps API pentru a permite utilizatorilor să urmărească locația comenzilor pe hartă.

e) Bibliografia recomandată:
Android Developers: https://developer.android.com/about
„Comprehensive Study and Technical Overview of Application Development in iOs Android and Windows Phone8” Anuja H. Vaidya, Sapan Naik
„User Data Management System Using Firebase Cloud” Roshan R.Kolte, Ekansh Moundekar
„Location Based Reminder Using Android and Google Maps” Neelu Lalband
„Development of an Android Grocery Checklist Application” Ana Antoniette C. Illahi Gershom Rob Narag, Manuel Lorenzo Parro, Luis Paolo Wenceslao

f) Termene obligatorii de consultații: Studentul a început dezvoltarea aplicației în septembrie 2022 și ne am întâlnit la fiecare 2-3 săptămâni
g) Locul și durata practicii: Universitatea „Sapientia” din Cluj-Napoca,
Facultatea de Științe Tehnice și Umaniste din Târgu Mureș, sala / laboratorul 327A
Primit tema la data de: mai 2022
Termen de predare: 2 iulie 2023

Semnătura Director Departament

Semnătura coordonatorului

Semnătura responsabilului programului de studiu

Antal

Semnătura candidatului

Kivonat

Az éttermi szolgáltatások és az online rendelési lehetőségek rohamos fejlődése az elmúlt években egyre nagyobb keresletet teremtett a kényelmes és könnyű ételrendelés iránt. Az emberek az éttermi élményt otthonukba szeretnék hozni, anélkül hogy elhagynák a kényelmet és a kikapcsolódást. Ebben a digitális korban az étteremnek lépést kell tartania a fogyasztói igényekkel, és meg kell találnia a módját annak, hogy kiszolgálja az ügyfeleket az online térben is. Ezen a ponton jön képbe a házhoz szállító applikáció egy étteremnek, amely lehetővé teszi a könnyű és gyors ételrendelést, valamint a házhoz szállítást.

A cél egy olyan applikáció létrehozása, amely az étteremnek lehetőséget ad az ügyfelek számára, hogy kényelmesen és egyszerűen rendeljenek az étlapjáról. Az alkalmazás segítségével az ügyfelek böngészhetik az ételek széles választékát, megtekinthetik azok részletes leírását és árait, valamint könnyedén összeállíthatják a rendelésüket. Az alkalmazás lehetővé teszi az online fizetést is, hogy a folyamat minél gördülékenyebb legyen mind a felhasználók, mind az étterem számára.

A házhoz szállító applikáció előnyei nem csak az ügyfelek számára nyújtanak kényelmet, hanem az étterem számára is hatékonyabb működést eredményeznek. Az alkalmazás segítségével az étterem könnyedén kezelheti a rendeléseket, nyomon követheti a szállítási folyamatot és optimalizálhatja az erőforrásokat. Emellett az alkalmazás lehetőséget ad az ügyféladatok kezelésére és az ügyfélkapcsolatok építésére, amely hosszú távon hűséges ügyfelekhez és pozitív visszajelzésekhez vezethet.

A következő dolgozatban bemutatom a házhoz szállító applikáció tervezését és fejlesztését az étterem számára.

Rezumat

Serviciile de restaurante și posibilitățile de comandă online au cunoscut o dezvoltare rapidă în ultimii ani, generând o cerere tot mai mare pentru comenzi de mâncare ușoare și convenabile. Oamenii doresc să aducă experiența de a mânca într-un restaurant în confortul propriei case, fără a renunța la confort și relaxare. În era digitală, restaurantul trebuie să țină pasul cu nevoile consumatorilor și să găsească modalități de a servi clienții și în mediul online. Aici intră în scenă o aplicație de livrare la domiciliu pentru un restaurant, care permite comenzi rapide și ușoare de mâncare, precum și livrare la domiciliu.

Scopul este de a crea o aplicație care să ofere restaurantului posibilitatea de a permite clienților să comande ușor și simplu din meniul său. Prin intermediul aplicației, clienții pot naviga printr-o varietate largă de preparate, pot vedea descrieri și prețuri detaliate și pot crea cu ușurință comanda lor. Aplicația permite, de asemenea, plata online, pentru a face procesul cât mai fluent atât pentru utilizatori, cât și pentru restaurant.

Beneficiile aplicației de livrare la domiciliu nu oferă doar confort clienților, ci aduc și o funcționare mai eficientă și mai productivă pentru restaurant. Prin intermediul aplicației, restaurantul poate gestiona ușor comenzile, poate urmări procesul de livrare și poate optimiza resursele disponibile. De asemenea, aplicația oferă posibilitatea gestionării datelor clienților și construirii relațiilor cu aceștia, ceea ce poate duce la clienți fideli și feedback pozitiv pe termen lung.

În următorul document, va prezint în detaliu planificarea și dezvoltarea aplicației de livrare la domiciliu pentru un restaurant.

Abstract

The restaurant industry and online ordering options have seen rapid growth in recent years, creating a growing demand for convenient and easy food delivery. People want to bring the restaurant experience to their homes without sacrificing comfort and relaxation. In this digital age, restaurants need to keep up with consumer demands and find ways to serve customers in the online space. This is where a restaurant delivery app comes into play, allowing for easy and fast food ordering and delivery.

The goal is to create an application that enables the restaurant to provide customers with a convenient and simple way to order from their menu. Through the app, customers can browse a wide range of dishes, view detailed descriptions and prices, and easily create their order. The app also allows for online payment, ensuring a seamless process for both users and the restaurant.

The benefits of a delivery app extend beyond customer convenience and also result in more efficient and successful restaurant operations. With the app, the restaurant can easily manage orders, track the delivery process, and optimize resources. Additionally, the app enables customer data management and the building of customer relationships, which can lead to loyal customers and positive feedback in the long run.

In the following document, I will provide a detailed overview of the planning and development of a restaurant delivery app.

Tartalomjegyzék

1. fejezet

Bevezető

Az éttermi házhozszállításra, a TastyGo, egy rendkívül hasznos eszköz az éttermek számára, mivel lehetővé teszi a saját belső házhozszállítási rendszerük kiépítését. Az alkalmazás két fő részből áll: a kliensoldalból és a szerveroldalból.

A kliensoldal az Android platformra fejlesztett felhasználóbarát alkalmazás, amelyet a vásárlók telepíthetnek és használhatnak. Az alkalmazás lehetővé teszi számukra, hogy könnyedén böngésszék az étterem teljes étlapját, megtekinthessék az ételek részletes leírását, az árakat és a rendelési lehetőségeket. A kliensoldalon a felhasználók kényelmesen összeállíthatják a rendelésüket.

A szerveroldal a TastyGo alkalmazás "motorja", amely a rendelések kezeléséért és a házhozszállítás koordinálásáért felelős. Amikor egy vásárló leadja a rendelését a kliensoldalon keresztül, az információ a szerveroldalra kerül. Itt a rendelés rögzítésre kerül, és az étterem munkatársai áttekinthetik az érkező rendeléseket. A szerveroldal lehetővé teszi a rendelések hatékony kezelését, azok státuszának frissítését és a vásárlók értesítését a rendelés állapotáról. Emellett a szerveroldal a kiszállítóknak is lehetőséget nyújt az útvonalak és szállítási helyek megjelenítésére egy térképen, így könnyedén és hatékonyan tudják teljesíteni a kiszállításokat.

Az alkalmazás használatával mind a vásárlók, mind pedig az éttermek számos előnnyel járnak. A vásárlók kényelmesen és gyorsan rendelhetnek az étterem széles választékából, könnyedén kiválaszthatják a fizetési módokat, és nyomon követhetik rendelésük állapotát. Az alkalmazás segítségével a vásárlóknek nem kell telefonon vagy személyesen rendelést leadniuk, hanem egyszerűen és kényelmesen intézhetik azt az alkalmazás segítségével.

Az éttermek számára is számos előnyt nyújt a TastyGo alkalmazás. Először is, lehetőséget kapnak arra, hogy kibővítsék üzletüket és elérjék az online vásárlókat. Az alkalmazás segítségével az éttermek teljes étlapjukat bemutathatják, ami lehetőséget ad a nagyobb termékkínálatra és az ételek kiemelésére. Emellett az alkalmazás könnyű és hatékony rendeléskezelést biztosít az éttermeknek. A rendelések automatikusan beérkeznek a rendszerbe, és a szerveroldal segítségével könnyedén követhetik azokat. Ez megkönnyíti a folyamatokat és minimalizálja az emberi hibák lehetőségét.

A TastyGo alkalmazás egyaránt előnyös a kiszállítóknak is. Az alkalmazás térképes megjelenítése lehetővé teszi számukra az optimális útvonalak kiválasztását és a szállítási helyek hatékony kezelését. Ez időt takarít meg nekik, és segít abban, hogy a rendeléseket időben és pontosan kézbesítsék a vásárlókhoz.

Tehát a TastyGo alkalmazás lehetőséget nyújt az éttermeknek, hogy kényelmes és hatékony házhozszállítási szolgáltatást nyújtsanak ügyfeleiknek. Az alkalmazás segítségével a vásárlók egyszerűen és gyorsan rendelhetnek, az éttermek könnyen kezelhetik a rendeléseket és nyomon követhetik azok állapotát, míg a kiszállítók optimális útvonalakat választhatnak. A TastyGo alkalmazás valódi előnyöket nyújt mind a vásárlóknak, mind pedig az éttermeknek, így a házhozszállítási élmény még kényelmesebbé és hatékonyabbá válik.

2. fejezet

Elméleti háttér

2.1. Android

Az Android egy mobil operációs rendszer, amelyet elsősorban okostelefonokhoz és táblagépekhez fejlesztettek ki. Az Android nyílt forráskódú, ami azt jelenti, hogy a fejlesztők szabadon hozzáférhetnek a forráskódhoz, és testreszabhatják az operációs rendszert az egyéni igényeik szerint. Az Androidot a Google fejleszti és támogatja, és a világ egyik legelterjedtebb mobil operációs rendszere.

Az Android rendkívül sokoldalú és számos előnnyel rendelkezik, amelyekért érdemes használni:

Nyílt forráskód: Az Android nyílt forráskódú jellege lehetővé teszi a fejlesztők számára, hogy szabadon hozzáférjenek és testre szabhassák az operációs rendszert. Ez nagyobb kreativitást és egyedi megoldásokat tesz lehetővé az alkalmazások fejlesztése során.

Széleskörű eszköztámogatás: Az Android kompatibilis számos különböző eszközzel, beleértve okostelefonokat, táblagépeket, okosórákat, intelligens tévét és egyéb eszközöket. Ez lehetővé teszi a fejlesztők számára, hogy alkalmazásaikat széles körben elérhetővé tegyék a felhasználók számára.

Nagy piaci részesedés: Az Android jelenleg a világ legnépszerűbb mobil operációs rendszere, amely jelentős piaci részesedéssel rendelkezik. Ez lehetőséget nyújt a fejlesztőknek széles körű felhasználói bázishoz való elérésre és üzleti lehetőségek kiaknázására.

Gazdag fejlesztői ökoszisztéma: Az Android rendelkezik egy gazdag fejlesztői közösséggel és számos eszközzel, amelyek segítségével könnyedén fejleszthetünk alkalmazásokat. A Google által nyújtott eszközök és platformok, mint például az Android Studio és a Firebase, segítséget nyújtanak az alkalmazások fejlesztéséhez, teszteléséhez és publikálásához.

Az Android rendszer folyamatosan fejlődik, és új funkciókkal és fejlesztői lehetőségekkel bővül. Az új verziók rendszeres kiadása révén az Android mindig a legújabb technológiai trendekre és felhasználói igényekre reagál. Ez biztosítja, hogy a fejlesztők mindig a legfrissebb eszközöket és funkciókat használhassák az alkalmazások létrehozásához.

Az Android rendkívül fejlett és sokoldalú operációs rendszer, amely lehetőséget nyújt a fejlesztőknek a teljesítményorientált alkalmazások készítésére, a grafikus felhasználói felületek megvalósítására, a hálózati kommunikáció kezelésére és sok más fejlett funkcióra. Az Android lehetővé teszi a fejlesztők számára az alkalmazások teljes körű testreszabását és személyre szabását, hogy a felhasználók számára egyedülálló élményt nyújthassanak.

Az Android használatával a fejlesztők széles körben elérhetik a felhasználókat, és fejlett alkalmazásokat hozhatnak létre a mobilplatformok számára.

Éppen ezért, az Android rendkívül előnyös platform egy ételrendelő alkalmazáshoz való használatra. Az Android az egyik legelterjedtebb mobil operációs rendszer, amely széles körben elterjedt a felhasználók körében. Ez azt jelenti, hogy az alkalmazásod potenciálisan elérheti a nagyobb felhasználói bázist, ami több letöltést, használatot és üzleti lehetőséget jelenthet. Számos különböző okostelefon- és táblagépmárkát támogat, ami azt jelenti, hogy az alkalmazás szélesebb körben elérhető lesz az Android-eszközök tulajdonosai számára. Ez magában foglalja különböző árkategóriákban elérhető eszközöket is, így elérhetőek a különböző vásárlói rétegek. Az Androidhoz elérhetőek kiváló fejlesztői eszközök és dokumentációk, amelyek segítségével könnyedén fejleszthető egy alkalmazás. Az Android Studio fejlesztői környezet számos funkciót és eszközt biztosít a fejlesztési folyamat egyszerűsítéséhez. Emellett a Google nyújt támogatást és erőforrásokat a fejlesztőknek, például a fejlesztői útmutatókat és az API-kat. Az Android rengeteg beépített funkcióval és lehetőséggel rendelkezik, ilyenek például a helymeghatározás, a fizetési rendszerek integrációja, értesítések kezelése, közösségi megosztás és még sok más. Az Android lehetővé teszi az alkalmazások számára a kényelmes, egyszerű és hatékony működést.

2.1.1. Java

A Java egy erőteljes és sokoldalú programozási nyelv, amely széles körben használatos a szoftverfejlesztés területén. Java-tanulás során felfedeztem, hogy milyen sok előnnyel jár ennek a nyelvnek a használata.

Először is, a Java platformfüggetlen, ami azt jelenti, hogy ugyanazt a Java kódot futtathatom különböző operációs rendszerek alatt, például Windows, macOS vagy Linux. Ez nagyon kényelmes, mert nem kell külön változatokat fejleszteni minden platformra, csupán egyszer kell megírni a kódot, és az bármilyen környezetben működik.

A Java objektumorientált nyelv, ami lehetővé teszi a kódbázis hatékony szervezését és újrafelhasználását. Az osztályok és az objektumok segítségével strukturálható és modularizálható a kód, ami javítja a fejlesztés hatékonyságát és karbantarthatóságát. Emellett a Java nagy figyelmet fordít az adatbiztonságra és a hibakezelésre is, ami hozzájárul a megbízható és stabil alkalmazások létrehozásához.

A Java egy JIT (Just-In-Time) fordítóval rendelkezik, amely lehetővé teszi a dinamikus kódkonverziót a futásidőben. Ez javítja a program futási idejű teljesítményét és optimalizálja a memóriakezelést. Emellett a Java beépített hibakezelési mechanizmusokkal rendelkezik, amelyek lehetővé teszik a megbízható és stabil alkalmazások fejlesztését.

A Java széles körben használt a vállalati szoftverfejlesztésben is. A Java EE (Enterprise Edition) keretrendszer lehetővé teszi a nagy méretű és összetett alkalmazások készítését, amelyek integrálódnak más rendszerekkel és biztosítják a skálázhatóságot és a megbízhatóságot.

A Java fejlesztői közössége óriási és aktív, így rengeteg erőforrás, dokumentáció és segítség áll rendelkezésre. Ez lehetővé teszi a folyamatos tanulást és fejlődést, valamint a problémák gyors megoldását. A Java nyílt forráskódú projektekben és keretrendszerekben is széles körben alkalmazható, így a közösségi fejlesztési lehetőségek is rendkívül gazdagok.

Összességében a Java egy olyan programozási nyelv, amely lehetővé teszi az átfogó alkalmazások készítését, kényelmes platformfüggetlen fejlesztést és megbízható működést biztosit.

2.1.2. Firebase Adatbázis

Firebase adatbázis egy felhőalapú NoSQL adatbázis-kezelő rendszer, amelyet a Google fejlesztett ki. Az adatbázis szolgáltatás része a Firebase platformnak, amely egy teljes körű fejlesztői platform a mobil- és webalkalmazásokhoz.

A Firebase adatbázis strukturálatlan dokumentumok tárolására és kezelésére szolgál. Az adatok JSON formátumban vannak tárolva, és a dokumentumokat gyűjteményekbe rendezhetjük. A Firebase adatbázis alapvetően NoSQL alapú, ami azt jelenti, hogy rugalmas szerkezetet kínál az adatok tárolásához és kezeléséhez, és nem igényel előre definiált sémát.

A valós idejű adatok szinkronizációja a Firebase adatbázis egyik kiemelkedő jellemzője, amely lehetővé teszi az adatok valós időben történő frissítését és szinkronizálását a különböző kliensek között. Ez azt jelenti, hogy amikor egy kliens oldalon módosítás történik az adatokban, az azonnal értesül és frissül a többi kliensen is, akik az adott adatokat figyelik. : A valós idejű adatok szinkronizációja lehetővé teszi az alkalmazások számára a valós időben történő frissítéseket és értesítéseket. Például, ha egy többfelhasználós chat alkalmazást vagy valós idejű frissítéseket kínáló alkalmazást használunk, az adatok azonnal megjelennek a kliens oldalon minden felhasználó számára, így gyors és interaktív élményt nyújt. Amikor több felhasználó dolgozik együtt egy alkalmazáson vagy dokumentumon, a valós idejű adatok szinkronizációja lehetővé teszi a valós időben történő együttműködést és azonnali visszajelzést. Például együttműködő szerkesztői alkalmazásban minden felhasználó láthatja a másik által végzett változtatásokat azonnal, így könnyebb és hatékonyabb az együttműködés. A valós idejű adatok szinkronizációja segít egyszerűen kezelni az adatokat az alkalmazásban. Az adatok közvetlenül a kliens oldalon módosíthatók, és azok azonnal frissülnek a szerver oldalon és a többi kliensnél. Ez kényelmes és hatékony adatmanipulációt tesz lehetővé anélkül, hogy különösen gondoskodnunk kellene az adatok frissítéséről vagy szinkronizálásáról.

A Firebase adatbázis lehetővé teszi az alkalmazások számára a könnyű skálázhatóságot. A Google infrastruktúráját használva az adatbázis automatikusan skálázódik az alkalmazás igényei szerint, így a nagy adatforgalommal és felhasználókkal rendelkező alkalmazások is hatékonyan kezelhetők. A Firebase adatbázis lehetővé teszi az alkalmazások számára az offline működést. Az adatokat helyileg is tárolja a kliens oldalon, és amikor újra elérhetővé válik az internetkapcsolat, a módosítások automatikusan szinkronizálódnak a szerverrel.

A Firebase integrációja Android alkalmazásokkal egy hatékony és megbízható módszer az alkalmazásfejlesztéshez. A Firebase olyan platform, amely számos fejlesztői szolgáltatást kínál, például adatbázis, autentikáció, tárolás, üzenetküldés és analitika. Az Android alkalmazásokhoz való összekapcsolása lehetővé teszi az alkalmazások funkcionalitásának bővítését és a felhasználói élmény javítását.

Egyrészt, a Firebase adatbázis lehetőséget nyújt a strukturálatlan adatok tárolására és kezelésére. Az Android alkalmazásban könnyedén csatlakoztathatjuk az alkalmazást a Firebase adatbázishoz, és használhatjuk azt a felhasználói adatok, termékinformációk

vagy chat-üzenetek tárolására. Ez a gyors és skálázható adatbázis lehetővé teszi az alkalmazásunk számára a hatékony adatmanipulációt és a valós idejű adatok szinkronizációját a felhasználók között.

Másrészt, a Firebase autentikációs szolgáltatása egyszerűvé teszi a felhasználók bejelentkezését és regisztrációját az alkalmazásban. Az Android alkalmazásba integrálva a felhasználók különböző bejelentkezési lehetőségekkel, például e-mail és jelszó, Google, Facebook vagy más közösségi hitelesítő szolgáltatásokkal léphetnek be az alkalmazásba. Ez biztosítja a felhasználók biztonságát és kényelmét, miközben lehetővé teszi a személyre szabott felhasználói élményt és a felhasználói adatok kezelését.

A Firebase tárolási szolgáltatása lehetővé teszi a felhasználók számára a médiafájlok (például képek, videók) feltöltését és tárolását a felhőben. Az Android alkalmazásban könnyedén használhatjuk ezt a szolgáltatást a felhasználók által feltöltött fájlok tárolására és elérésére. Ezáltal optimalizálhatjuk az alkalmazás méretét és biztosítjuk a felhasználók tartalmának biztonságát és elérhetőségét.

Végül, a Firebase analitikai szolgáltatása lehetővé teszi az alkalmazások teljesítményének és felhasználói viselkedésének elemzését. Az Android alkalmazásban könnyedén beállíthatjuk az analitikát, hogy nyomon kövessük a felhasználói aktivitást, az alkalmazás használatát és az üzleti célok teljesítését. Ez az információ fontos lehet a felhasználói élmény optimalizálásában és az alkalmazásunk továbbfejlesztésében.

A Firebase és az Android alkalmazások integrációja tehát rendkívül előnyös a hatékony és sikeres alkalmazásfejlesztéshez. A Firebase széleskörű szolgáltatásai lehetővé teszik az alkalmazások bővítését, a felhasználói élmény javítását és az üzleti célok elérését. A könnyű integráció és a részletes dokumentáció révén a fejlesztők gyorsan és hatékonyan alkalmazhatják a Firebase szolgáltatásait az Android alkalmazásaikban.

2.1.3. Google API

A Google Helymeghatározás API egy hatékony eszköz az Android alkalmazások számára, amely lehetővé teszi a felhasználók helyzetének meghatározását és a helyalapú szolgáltatások integrálását az alkalmazásba. Ez az API lehetővé teszi az alkalmazások számára, hogy hozzáférjenek a felhasználók földrajzi helyzetéhez, és használják ezt az információt a különböző funkcionalitásokhoz, például helymegjelölések, navigáció, térképek, helyi keresés vagy helyspecifikus tartalom megjelenítése céljából.

Az API segítségével az Android alkalmazások könnyedén meghatározhatják a felhasználó helyzetét GPS, mobilhálózat vagy Wi-Fi alapján. Az API lehetőséget nyújt az aktuális földrajzi koordináták lekérdezésére, a felhasználó mozgásának nyomon követésére és a helyzetváltozások figyelésére. Ezáltal az alkalmazások pontosabb és személyre szabottabb funkcionalitást nyújthatnak a felhasználóknak, például helyspecifikus ajánlatokat, közeli helyek keresését vagy térképen történő navigációt.

Az API további előnyei közé tartozik a különböző helyzeti adatokhoz való hozzáférés. Az alkalmazások képesek lehetnek megszerezni a felhasználói címeket, a pontos helyrajzi információkat, a sebességet és az irányt, valamint más földrajzi jellemzőket. Ezek az adatok lehetővé teszik az alkalmazások számára, hogy személyre szabottabb élményt nyújtsanak, például az alkalmazás nyelvének és tartalmának automatikus beállításával a felhasználó aktuális helyzetéhez.

Az API továbbá lehetőséget nyújt a helyalapú szolgáltatások integrálására is. Az alkalmazások könnyedén használhatják a Google Térkép szolgáltatásait, például a térkép megjelenítését, a helyek keresését, a navigációt vagy a helyspecifikus információk lekérdezését. Ezáltal az alkalmazások felhasználóbarátabb és funkcionalitásban gazdagabb élményt nyújthatnak a felhasználóknak.

Az Android alkalmazások számára a Google Helymeghatározás API tehát rendkívül előnyös. Az API segítségével az alkalmazások pontosan meghatározhatják a felhasználó helyzetét, és használhatják ezt az információt a helyalapú szolgáltatások és személyre szabott funkcionalitások nyújtásához. Ezáltal az alkalmazások sokoldalúbbak, interaktívabbak és felhasználóbarátabbak lehetnek.

3. fejezet

Funkcionalitások bemutatása

3.1. ábra. Logo

3.1. Ügyfél oldal (Client-side)

Az alkalmazás ügyfél oldala (Client-side) a TastyGo alkalmazásban a vásárlók számára elérhető felületet és funkciókat foglalja magában. Ez a rész felelős az étterem menüjének böngészéséért, az ételek rendeléséért és az összes többi interakcióért, amelyet a felhasználó az alkalmazáson keresztül végez.

Az alkalmazás ügyfél oldala intuitív és felhasználóbarát felülettel rendelkezik, amely lehetővé teszi a vásárlók számára a könnyű navigációt és az ételek keresését. A felhasználók megtekinthetik az étterem teljes választékát, beleértve a különböző kategóriákat, leírásokat és árakat. Az ételekhez tartozó képek és egyéb információk segítenek a vásárlóknak a választásban.

Az alkalmazás ügyfél oldala lehetőséget biztosít a rendelések összeállítására és testreszabására. A vásárlók könnyedén kiválaszthatják a kívánt ételeket a menüből, hozzáadhatják azokat a kosárhoz, és testre szabhatják a rendelést a különböző opciók (pl. hozzávalók, mennyiség) segítségével. Az alkalmazás lehetőséget nyújt az online fizetésre is, így a vásárlók a rendelés leadása után azonnal rendezhetik a számlát.

Az alkalmazás ügyfél oldala lehetőséget ad a vásárlók számára, hogy teljes mértékben kihasználják a TastyGo alkalmazás előnyeit és kényelmét, hogy könnyedén és gyorsan rendelhessenek az étterem kínálatából, és élvezhessék a finom ételeket a saját otthonukban vagy bárhol, ahonnan kényelmesen megrendelhetik a házhozszállítást.

3.1.1. Felhasználói regisztráció és bejelentkezés

A felhasználói regisztráció egy alapvető és elengedhetetlen funkciója a TastyGo alkalmazásnak, amely lehetővé teszi az új felhasználók számára, hogy hozzáférjenek az ételrendelési szolgáltatásokhoz és élvezhessék az alkalmazás nyújtotta előnyöket. Ez a folyamat a következő lépésekből áll:

1.Regisztrációs űrlap kitöltése: Az új felhasználóknak meg kell adniuk bizonyos információkat, mint például a teljes nevüket, e-mail címüket, jelszavukat és telefonszámukat. Az űrlap kitöltésekor fontos, hogy a felhasználók minden kötelező mezőt helyesen és teljes egészében töltsenek ki.

2.Regisztrációs adatok mentése az adatbázisban: Ha az összes ellenőrzés sikeres volt, az alkalmazás el menti a felhasználó regisztrációs adatait az adatbázisba. Ez magában foglalja az e-mail címet, jelszót és felhasználónevet.

3.Regisztráció visszaigazolása: Miután a regisztráció sikeresen megtörtént, a felhasználó használhatja a fiókját és élvezheti az alkalmazást.

A felhasználói bejelentkezés fontos része a TastyGo alkalmazásnak, amely lehetővé teszi a regisztrált felhasználók számára, hogy hozzáférjenek fiókjukhoz és élvezhessék az alkalmazás funkcióit. Az alábbiakban bemutatom a Sign In (Bejelentkezés) osztály néhány kulcsfontosságú részletét:

1.UI elemek inicializálása: Az osztály inicializálja a bejelentkezéshez szükséges UI elemeket, mint például a telefonszám és jelszó mezőket, valamint a bejelentkezés gombot.

2.Firebase konfiguráció: Az osztály inicializálja a Firebase adatbázis kapcsolatot, amely a felhasználói adatok tárolásáért és hitelesítéséért felelős.

3.Bejelentkezési folyamat kezelése: Az osztály létrehoz egy ClickListener-t a bejelentkezés gombhoz, amely meghívja az onClick metódust a gomb megnyomásakor.

4.Adatellenőrzés és hitelesítés: Az onClick metódusban az alkalmazás ellenőrzi a felhasználó által megadott adatokat. Először ellenőrzi, hogy van-e internetkapcsolat, majd folytatja a bejelentkezési folyamattal, ha van. Ezután lekéri az adatbázisból a felhasználót az általa megadott telefonszám alapján.

5.Hitelesítés eredményének kezelése: Az osztály figyeli a Firebase adatbázist az adatváltozásokra. Ha a felhasználó létezik az adatbázisban, összehasonlítja a megadott jelszót a tárolt jelszóval. Ha egyezést talál, sikeres bejelentkezést jelzi a felhasználónak, és átirányítja a kezdőképernyőre (Home Activity). Ellenkező esetben hibás jelszó üzenetet jelenít meg.

3.1.2. Menü böngészése és keresése

A felhasználók számára az étterem teljes menüjének böngészését és keresését a FoodList illetve a Home osztályok teszik lehetővé. A menü különböző kategóriái között navigálhatnak, és képesek kereséseket végezni az ételek között. A Home o osztály megjeleníti a kategóriákat, mig a FoodList egy adott katergóriához tartozó ételek információit, mint például a kép az ételről és az étel neve..

Az ételek listájának megjelenítéséért a RecyclerView felelős.

A RecyclerView egy hatékony és rugalmas osztály az Android Java-ban, amely lehetővé teszi a nagy adathalmazok hatékony megjelenítését listanézetben. A RecyclerView az előző verziók ListView-jának továbbfejlesztett változata, és számos előnnyel rendelkezik.

A RecyclerView használatával lehetőség van a listaelemek sokoldalúabb és testreszabottabb megjelenítésére, valamint hatékonyabb adatkezelésre. Hatékony megjelenítés és újrafelhasználás: A RecyclerView csak a képernyőn megjelenített elemeket tölti be, és a listát görgetve újrahasznosítja a listaelemeket. Ez javítja a teljesítményt és csökkenti az erőforrásfogyasztást, különösen nagy adathalmazok esetén. A RecyclerView elemeinek elrenezéséért a LayoutManager felel. Különböző LayoutManager típusok állnak rendelkezésre, melyek lehetővé teszik az elemek rugalmas elhelyezését és megjelenítését a RecyclerView-ban.

A RecyclerView-hoz tartozik egy Adapter osztály, amely felelős az adatok átadásáért a RecyclerView-nak. Az Adapter feladata az adatok forrásának, például egy adatbázis vagy lista, kezelése, valamint az elemek megfelelő nézetekkel való összekapcsolása. Az Adapter meghatározza, hogy melyik nézetet kell megjeleníteni az egyes elemekhez, és kezeli a nézetek újrafelhasználását.

A Home Activityben a loadMenu() metódust hívjuk meg, amely betölti az ételek listáját a Firebase adatbázisból. Ehhez inicializáljuk a RecyclerView-t (*recycler_menu*), beállítjuk annak méretét, a LinearLayoutManager-t használjuk a megfelelő elrendezéshez, majd meghívjuk a loadMenu() metódust.

A loadMenu() metódusban a FirebaseRecyclerOptions osztályt használjuk az ételek listájához, amelyeket a Category osztályból szeretnénk lekérni. A FirebaseRecyclerAdapter segítségével beállítjuk az adatok megjelenítését a MenuViewHolder osztályban. Az onBindViewHolder() metódusban beállítjuk a nézetek tartalmát a megfelelő adatokkal, például az ételek nevével és képével. A setItemClickListener() metódust használjuk, hogy reagáljunk a menüelemek kattintására, és elindítsunk egy FoodList Activity-t az adott kategória azonosítójával. Az onCreateViewHolder() metódusban pedig beállítjuk a nézetet, amelyet használni fogunk a menüelemek megjelenítésére. Végül beállítjuk az adaptert a RecyclerView-ra és elkezdjük figyelni a változásokat az adapterben a startListening() metódussal.

Az "Intent service = new Intent(Home.this, ListenOrder.class);" részben egy szervizt indítunk, amely figyeli az új rendeléseket.

Ha egy kategóriára kattintunk a RecyclerView-n belül, akkor meghívódik a következő sor: Intent foodList = new Intent(Home.this, FoodList.class); foodList.putExtra("CategoryId", adapter.getRef(position).getKey());

Ezzel átadjuk a FoodListnek a categoryId-t és innen fogja tudni, hogy melyik kategória ételeit kell betöltse az adatbázisból.

A FoodViewHolder osztály a RecyclerView része, és felelős a listaelemek nézetének tárolásáért és hozzáféréséért. A ViewHolder segíti a hatékony megjelenítést és görgetést a listában. Az Adapter felelős a ViewHolder objektumok létrehozásáért és kezeléséért.

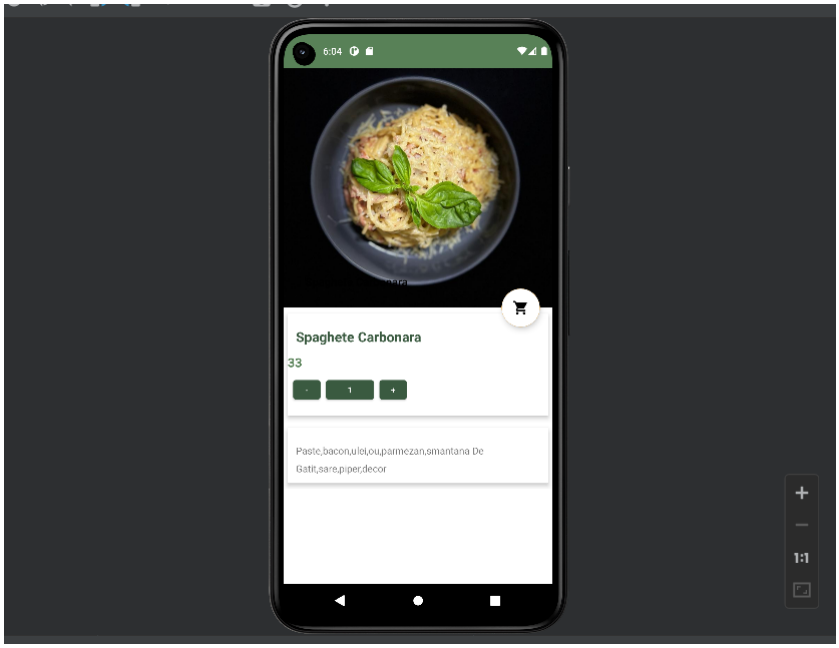
A FoodList Activity lefutásakor meghivódik az onCreate() metódus. A setContentView(R.layout.$activity_food_list$) sorral beállítjuk a megfelelő layout fájlt, amely meghatározza az Activity kinézetét és tartalmát. Inicializáljuk a RecyclerView-t, beállítjuk annak a méretét (setHasFixedSize(true)), majd létrehozzuk és beállítjuk a LinearLayoutManager-t a RecyclerView-hoz. Ellenőrizzük, hogy van-e intent. Az intentek az Activity-k közötti kommunikáció eszközei. Ha van intent, akkor kinyerjük belőle a kategória azonosítóját (categoryId). Az Intent egy olyan osztály az Android Java-ban, amely lehetővé teszi az alkalmazások közötti kommunikációt. Az Intent osztály segítségével az alkalmazások képesek kommunikálni egymással és adatokat megosztani.

Ellenőrizzük az internetkapcsolatot a Common.isConnectedToInternet(getBaseContext()) metódussal. Ez a metódus ellenőrzi, hogy a készülék csatlakozva van-e az internethez. Ha nincs internetkapcsolat, akkor megjelenítünk egy Toast üzenetet, amely arra kéri a felhasználót, hogy ellenőrizze a kapcsolatát.

Ha van érvényes kategória azonosító (categoryId), akkor meghívjuk a loadListFood() metódust ezzel az azonosítóval. Ez a metódus betölti az ételek listáját a megadott kategória alapján.

Az első sorban létrehozunk egy lekérdezést (Query), amely az adatbázisból kiválasztja azokat az ételeket, amelyeknek a menuId mezője megegyezik a megadott categoryId paraméterrel. Ez azért szükséges, hogy csak az adott kategóriához tartozó ételek kerüljenek lekérdezésre. Az options változóban beállítjuk a FirebaseRecyclerAdapter konfigurációs lehetőségeit, a setQuery metódussal megadjuk a lekérdezést és a Food osztályt, amelyre az eredményeket illeszteni szeretnénk. Létrehozunk egy adapter objektumot, amely az ételek megjelenítéséért felelős. Az onBindViewHolder metódusban beállítjuk az egyes ételek nevét és képét a megfelelő ViewHolder-en keresztül. Egy ItemClickListener-t hozunk létre, amely figyeli az ételekre kattintásokat. Ha egy ételre kattintunk, akkor elindítunk egy új Activity-t (FoodDetail), amely megjeleníti az étel részleteit. Az Intent-ben átadjuk az étel azonosítóját (FoodId), amelyet az adapter getRef(position).getKey() metódusa ad vissza. Végül elindítjuk az adapter figyelését (adapter.startListening()), hogy frissítse az elemeket az adatbázisban történő változások esetén.

Egy étel részleteinek megjelenitéséért és betöltéséért a felhasználói felületen a FoodDetail osztály felelős. Az osztály kezdetén inicializálódnak a szükséges adattagok, például a szövegmezők, kép és gombok. Az onCreate metódusban történik az Activity inicializálása és a felhasználói felület elemeinek beállítása. Például az eseménykezelők hozzáadása a gombokhoz, illetve az étel részleteinek betöltése az adatbázisból. A decreaseValue metódus csökkenti a darabszám értékét eggyel, amennyiben az érték nagyobb, mint 1, az increaseValue metódus pedig növeli a darabszám értékét eggyel. A getDetailFood metódus lekéri az étel részleteit az adatbázisból a foodId azonosító alapján. Az adatokat a Food objektumba menti, majd megjeleníti az étel képét, címét, árát és leírását a felhasználói felületen. Lehetőséget nyújt a felhasználónak az étel darabszámának kiválasztására, a kosárba helyezésre, valamint az étel részleteinek megtekintésére.



3.2. ábra. Food Detail

3.1.3. Rendelés

A FoodDetail osztályban szereplő gombokkal tudjuk irányitani a mennyiséget, amit a kosárba szeretnék helyezni. A felhasználó kosarának megjelenítését és kezelését a Cart osztáy végzi az alkalmazásban. Az osztály felelős a kosár tartalmának betöltéséért, az étel rendeléséért, valamint a kosár törléséért és az értékek megjelenítéséért. Az onCreate metódusban inicializáljuk az osztályváltozókat, például a RecyclerView-t, az adatbázist és a hivatkozásokat. Beállítjuk a RecyclerView elrendezését és az eseményfigyelőket. Ezután meghívjuk a loadListFood metódust, amely betölti és megjeleníti a kosárban lévő ételeket.

A showAlertDialog metódusban egy párbeszédpanelt jelenítünk meg, ahol a felhasználó megadhatja a szállítási címet. Ha a felhasználó elfogadja a rendelést, létrehozunk egy Request objektumot a felhasználó adataival, a kosár tartalmával és a szállítási címmel. Ezután a rendelést feltöltjük a Firebase adatbázisba, töröljük a kosarat és visszajelzést adunk a felhasználónak a sikeres rendelésről.

A loadListFood metódusban betöltjük a kosár tartalmát a helyi adatbázisból, létrehozzuk a kosár adaptert és beállítjuk a RecyclerView-nak. Kiszámítjuk a teljes árat a kosárban lévő tételek árai és mennyisége alapján, majd formázzuk a megfelelő pénznem és számlálási formátum szerint. Az összesített árat beállítjuk a megfelelő TextView-n.

Az onContextItemSelected metódusban kezeljük a kosárelemekre kattintást és a "Törlés" opciót. Amikor a felhasználó törli egy elemet a kosárból, az deleteCart metódus eltávolítja az elemet a kosárból, tisztítja a kosár tartalmát a helyi adatbázisban, majd újra hozzáadja a megmaradt elemeket a kosárhoz és frissíti a megjelenítést.

Az CartViewHolder osztály felelős a kosár elemeinek nézetének kezeléséért a RecyclerView-ben. A nézet tartalmazza a kosárban lévő elemek nevét, árát és mennyiségét. Az osztály implementálja a View.OnClickListener és View.OnCreateContextMenuListener interfészeket, hogy kezelje a kattintásokat és a kontextmenüt.

Az osztály konstruktorában inicializáljuk a nézeteket, például a TextView-kat, amelyek megjelenítik az elemek nevét, árát és mennyiségét. Beállítjuk a nézetekhez tartozó eseményfigyelőket, hogy reagálhassanak a kattintásokra és a kontextmenüre.

Az onCreateContextMenu metódusban beállítjuk a kontextmenü fejlécét és hozzáadjuk a "Törlés" opciót. Az opció kiválasztása esetén a getAbsoluteAdapterPosition() metódus segítségével megkapjuk az adott elem abszolút pozícióját a listában.

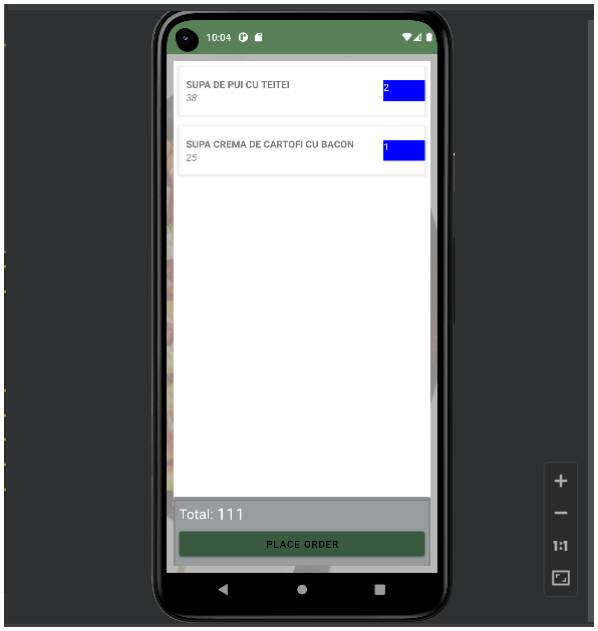
A CartAdapter osztály felelős a kosár elemeinek megjelenítéséért a RecyclerView-ban. Az osztály példányosításakor megkapja a kosár elemeinek listáját és a kontextust. Az osztály implementálja a RecyclerView.Adapter<CartViewHolder> interfészt, és így meghatározza a nézetek létrehozását és adataikkal való feltöltését.

Az onCreateViewHolder metódusban a LayoutInflater segítségével betöltjük a *cart_layout*.xml elnevezésű elrendezést, amely tartalmazza a kosár elemeinek nézetét. Ezután létrehozunk egy új CartViewHolder példányt és visszaadjuk azt.

A onBindViewHolder metódusban a megfelelő elemhez tartozó adatokat állítjuk be a nézetekben. Beállítjuk a mennyiséget, háttérszínt és szövegszínt a megfelelő TextView-ban. Kiszámítjuk az árat a mennyiség és az ár alapján, majd beállítjuk a formázott árat és az elem nevét a megfelelő TextView-ba.

A getItemCount metódusban visszaadjuk a kosár elemeinek számát, ami meghatározza a RecyclerView-ban megjelenített elemek számát.

A CartViewHolder és CartAdapter osztályok együttműködve felelősek a kosár elemeinek megjelenítéséért és kezeléséért a RecyclerView-ban. A CartAdapter felelős a nézetek létrehozásáért és az adatok betöltéséért, míg a CartViewHolder kezeli a nézetek eseményeit és a kontextmenüt.



3.3. ábra. Cart

3.2. Szerveroldal (Server-side)

3.2.1. Bejelentkezés

A Szerveroldal egy bejelentkezési képernyővel indul, amelyet a SignIn osztály valósít meg. Az alkalmazás célja az, hogy az étteremhez tartozó adminitsztrációs feladatokat lehessen végrehajtani rajta. A SignIn osztály felelős az adminok bejelentkezéséért az alkalmazásban.

Az osztályban szerepelnek EditText mezők, amelyeket a felhasználók használnak a telefonszámuk és jelszavuk megadásához. Emellett van egy bejelentkezés gomb, amelyet a felhasználók megnyomhatnak a bejelentkezési folyamat elindításához.

Az alkalmazás itt is Firebase adatbázist használ, amelyben a felhasználók adatai tárolódnak.

Amikor a felhasználó megnyomja a bejelentkezés gombot, a signInUser() metódus hívódik meg. Ez a metódus előkészíti a bejelentkezési folyamatot. Először megjelenik egy ProgressDialog, amely arra figyelmezteti a felhasználót, hogy várnia kell amig betöltődnek az adatok, ezután a felhasználó által megadott telefonszám és jelszó a localPhone és localPassword változókban tárolódik.

Ezután a users referenciához hozzáadódik egy ValueEventListener-t . Amikor az adatbázisban változás történik, az onDataChange() metódus hívódik meg. Itt ellenőrizzük, hogy az adott telefonszám már szerepel-e az adatbázisban. Ha igen, akkor a felhasználó adatait lekérem és ellenőrzöm, hogy az admin fielde true-e. Ha igen, összehasonlítom a megadott jelszóval. Ha egyezik, akkor sikeres bejelentkezés történt, és a felhasználót átirányítom a Home osztályba. Ha a jelszó nem egyezik, hibaüzenet jelenik meg. Ha az admin mező false, akkor pedig egy másik hibaüzenet jelenik meg, amely arra kéri a felhasználót, hogy admin fiókkal jelentkezzen be. Ha a telefonszám nem található az adatbázisban, akkor szintén hibaüzenet jelenik meg, hogy a felhasználó nem létezik.

3.2.2. Menü módosítása, törlése és editálása

Elsősorban itt is, akárcsak a Client-sideon, megjelenítjük a kategóriákat, majd a kategóriákhoz tartozó ételeket. Az onCreate() metódusban inicializálódnak és összekapcsolódnak a különböző felhasználói felületi elemek és adatforrások.

Első lépésként létrehozásra és beállításra kerülnek a szükséges adatbázis és tárolóhivatkozások (FirebaseDatabase, DatabaseReference, FirebaseStorage, StorageReference).

Ezután inicializálódik és beállításra kerül a RecyclerView, amin a különböző ételek listája jelenik majd meg. Ehhez egy LayoutManager is létrejön, amely felelős az elemek elrendezéséért a RecyclerView-n belül. A LinearLayoutManager azt határozza meg, hogy a lista elemei függőlegesen jelennek meg egymás alatt.

Ezt követően inicializálásra kerülnek a gombok és más felhasználói felületi elemek, például a FloatingActionButton, amelynek az eseménykezelője a showAddFoodDialog() metódust hívja meg.

Ha az Intent nem üres és tartalmaz egy "CategoryId" kulcsú extra adatot, akkor az értékét kinyerjük és betöltjük a listát az adott kategóriával rendelkező ételekkel a loadListFood() metódus segítségével.

A loadListFood() metódusban egy lekérdezés (Query) kerül végrehajtásra a Firebase adatbázisban, amely az ételek listáját adja vissza a megadott kategória alapján rendezve. Az eredményt egy FirebaseRecyclerOptions objektumba helyezzük, majd ezt az objektumot használjuk egy FirebaseRecyclerAdapter inicializálásához. Az adapter felelős az adatok megjelenítéséért és az elemek megfelelő kezeléséért a RecyclerView-n belül.

Az adapter beállítása után a RecyclerView megkapja az adaptert és elindul az adatok figyelése a startListening() metódus meghívásával.

A Floating Action Button (FAB) egy kör alakú ikon gomb, amely a tartalom fölött lebegve elősegíti az alkalmazás elsődleges műveleteinek elvégzését. Általában a legfontosabb funkciókhoz kapcsolódik, és lehetővé teszi az alkalmazások számára, hogy egyszerűbbé és könnyebben használhatóvá tegyék azokat. A FloatingActionButtonhoz egy onClick() eseménykezelő függvény van rendelve, amikor a felhasználó rákattint erre a gombra, egy párbeszédpanel (AlertDialog) jelenik meg, amelyben az admin új ételeket adhat hozzá a menühöz.

A showAddFoodDialog() metódus inicializálja az AlertDialog.Builder objektumot, beállítja a párbeszédpanel címét és üzenetét, majd létrehoz egy nézetet (View), amely tartalmazza a dialogpanelen megjelenő felhasználói felületi elemeket.

A metódusban megtörténik a dialogpanelen megjelenő felhasználói felületi elemek inicializálása (pl. EditText-ek, gombok), valamint az ezekhez tartozó eseménykezelők beállítása.

A "Select" gombra való kattintásra az onClick() metódusban meghívódik a chooseImage() metódus, ami a kép kiválasztásáért felelős.

Az "Upload" gombra való kattintásra az onClick() metódusban meghívódik az uploadImage() metódus, ami a kép feltöltését végzi.

Az AlertDialog.Builder objektumhoz hozzáadódik a létrehozott nézet, és beállításra kerül az ikon is.

Az "Igen" gombra kattintva a dialóguspanel elrejtésre kerül, majd ellenőrzésre kerül, hogy az új étel objektum (newFood), megnézzük ha ez nem üres-e. Ha nem üres, akkor az új étel hozzáadásra kerül az adatbázishoz (foodList.push().setValue(newFood)), és egy Snackbar üzenet jelenik meg, amely visszajelzi, hogy az új étel sikeresen hozzáadásra került.

A "Nem" gombra kattintva a dialóguspanel egyszerűen bezáródik.

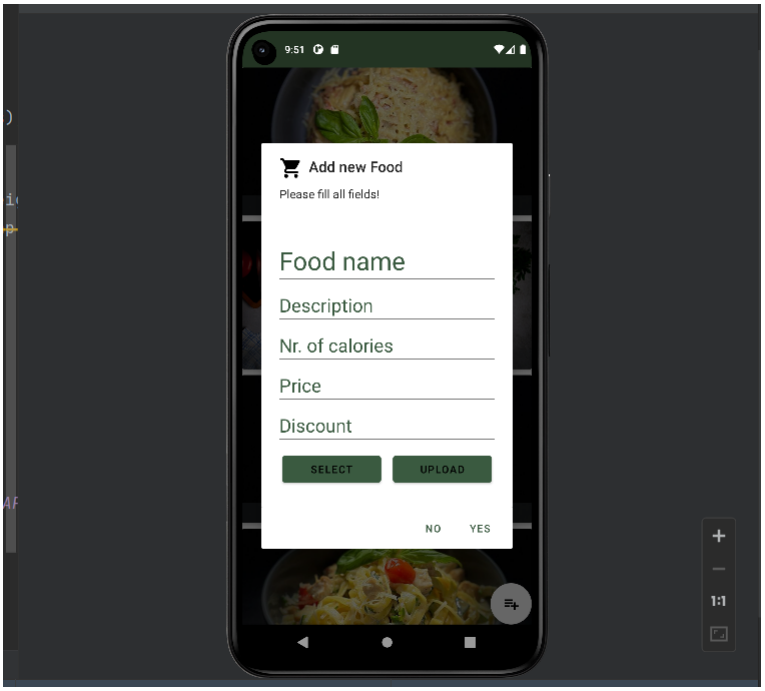
Az Uploadhoz hasonlóan, az Edit és Delete metódusok is implementálásra kerülnek, annak érdekében, hogy a az ételek információi napra készek legyenek.

3.2.3. Rendelés státusza

A OrderStatus osztály az aktuális rendelések állapotát jeleníti meg.

Az osztályban található onCreate() metódus a kezdőállapotot inicializálja. Először elérjük a Firebase adatbázist a FirebaseDatabase.getInstance() metódussal, majd hozzárendeljük a "Requests" csomóponthoz a db.getReference("Requests") segítségével. Ezután beállítjuk a RecyclerView-t, annak elrendezését és az adatok betöltéséért felelős loadOrders() metódust hívjuk meg.

A loadOrders() metódusban létrehozunk egy FirebaseRecyclerOptions objektumot, amely beállítja az adatok lekérdezését és a Request osztály használatát. Ezután inicializáljuk az adaptert egy FirebaseRecyclerAdapter objektummal, amelynek onBindView-



3.4. ábra. Food Detail

Holder() metódusában töltjük be az adatokat a megfelelő nézeti elemekbe. A nézet elemek inicializálása és a kattintás eseménykezelése itt történik meg.

Az onCreateViewHolder() metódusban egy új nézetet hozunk létre a $R.layout.order_layout$ layout fájlból, amely a RecyclerView elemek megjelenítéséért felelős.

Az adapter elindítása és beállítása a RecyclerView-nél a adapter.startListening() és recyclerView.setAdapter(adapter) metódusokkal történik.

Az onContextItemSelected() metódusban az elemekre való kattintás eseményét kezeljük. Ha a kiválasztott menüelem "UPDATE", akkor a showUpdateDialog() metódust hívjuk meg, amely dialóguspanelt jelenít meg a rendelés státuszának frissítéséhez, itt változtathatja meg az admin, ha a rendelés már úton van, vagy mér kézbesitve van. Változtatás után a kliens értesitést kap a rendelése státuszáról. Ha a kiválasztott menüelem "DELETE", akkor a deleteOrder() metódus segítségével töröljük az adott rendelést.

3.2.4. Track Order

A TrackOrder a rendelések nyomon követéséért felelős, és hogy az étteremben lévő alkalmazottak könnyebben megtalálhassák a rendelés cimét. Az onCreate az aktivitás létrehozásakor fut le. Beállítja a megfelelő nézetet ($activity_tracking_order.xml$), inicializálja a Google Térképet, és ellenőrzi a helymeghatározás engedélyeit. Ahhoz hogy helymeghatározást tudjunk végre hajtani szükséges két engedélyt megadni a Manifest fileban. Az android.permission.$ACCESS\_COARSE\_LOCATION$ és android.permission.$ACCESS\_FINE\_LOCATION$ engedélyek azért fontosak, mert lehetővé teszik az alkalmazás számára, hogy hozzáférjen a felhasználó helyzetéhez. Ezek az engedélyek különböző pontossági szinteket jelölnek a helymeghatározás során.

Az $ACCESS_COARSE_LOCATION$ engedély a durvább helymeghatározást jelenti. Az alkalmazás az ilyen engedéllyel elérheti az úgynevezett "közelítő" helyzetet, ami alapvetően a mobiltelefon toronyállomásainak és a Wi-Fi hozzáférési pontoknak az adataira támaszkodik. Ez az engedély lehetővé teszi az alkalmazás számára, hogy meghatározza a felhasználó közelítő tartózkodási helyét anélkül, hogy pontos GPS koordinátákra lenne szükség. Az $ACCESS_FINE_LOCATION$ engedély pedig pontosabb helymeghatározást biztosít. Az alkalmazás a GPS rendszeren keresztül lekérdezheti a felhasználó pontos tartózkodási helyét. Az engedély megadása lehetővé teszi az alkalmazás számára, hogy magasabb pontossággal határozza meg a felhasználó helyzetét, amennyiben a készülék rendelkezik GPS funkcióval. Fontos megjegyezni, hogy az engedélyek megszerzése a felhasználó beleegyezését igényli. Amikor az alkalmazás először kérik ezeket az engedélyeket, a rendszer felhívja a felhasználó figyelmét, és lehetőséget nyújt neki az engedélyek elfogadására vagy elutasítására. Az engedélyek hiánya esetén az alkalmazás nem férhet hozzá a felhasználó helyzetéhez, és nem tudja megjeleníteni vagy követni az útvonalat a térképen. A displayLocation metódus felelős a felhasználó aktuális helyzetének megjelenítéséért a térképen, ellenőrzi a helymeghatározás engedélyeit, majd lekéri a felhasználó aktuális helyzetét a LocationServices API segítségével. Ezután a helyzetet megjeleníti egy jelölővel a térképen.

A drawRoute metódus felelős az útvonal megrajzolásáért a felhasználó aktuális helyzetétől az ételrendelés helyéig. Az ételrendelés helyét a getGeoCode API hívásával lekéri a szolgáltatástól, majd az útvonalat a getDirections API hívásával kéri le, az útvonalat végül a PolylineOptions segítségével rajzolja a térképre.

A requestRuntimePermission: metódus kéri a helymeghatározás engedélyét a felhasználótól.

Az onRequestPermissionsResult metódus kezeli a helymeghatározás engedélyének kérése utáni választ. Ha a felhasználó engedélyezte a helymeghatározást, akkor inicializálja a Google API kliensét és megjeleníti a felhasználó helyzetét.

ParserTask: Ez egy belső osztály, amely az útvonalak feldolgozásáért felelős. Az AsyncTask osztályból származik, és háttérfolyamatban dolgozza fel a JSON választ. A feldolgozott útvonalat a térképre rajzolja a PolylineOptions segítségével.

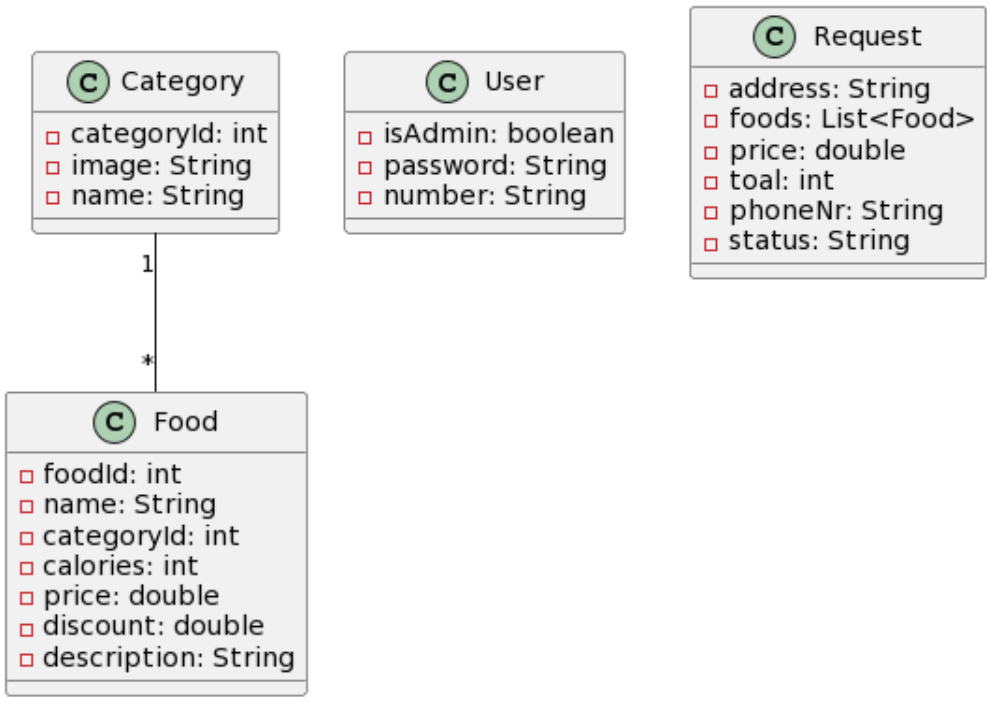
Ezeken kívül a kód tartalmaz import utasításokat, valamint néhány segédfüggvényt és adattagot. Az alkalmazás működéséhez szükséges a Google Play szolgáltatások és a helymeghatározás engedélye. A kódban használt funkciók segítségével az alkalmazás megjeleníti a felhasználó helyzetét, valamint az ételrendelés helyzetét és az útvonalat a két hely között.

4. fejezet

Diagramok

4.1. Class Diagram

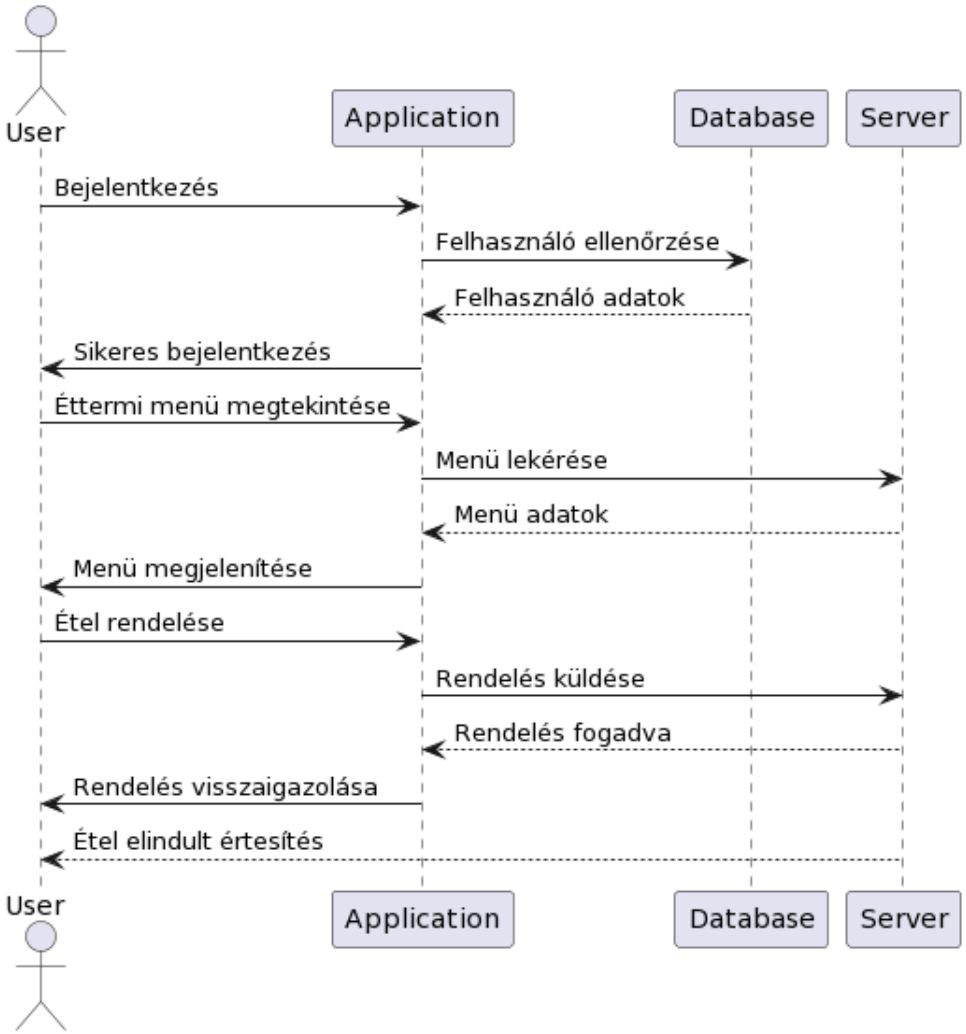
Az osztálydiagram (class diagram) az objektumorientált tervezés egyik típusú UML diagramja, amely bemutatja az osztályokat, azok attribútumait és metódusait, valamint az osztályok közötti kapcsolatokat. Az osztálydiagramok segítenek megérteni egy rendszer osztályainak struktúráját, hierarchiáját és viszonyait. Segítenek a fejlesztőknek és tervezőknek átlátni a rendszer struktúráját és könnyebben megtervezni, implementálni és karbantartani a szoftvert.



4.1. ábra. Class Diagram

4.2. Sequence Diagram

A Sequence Diagram bemutatja a TastyGo alkalmazásban található folyamatokat és azok kölcsönhatásait a felhasználó és az admin között. A diagramban látható, hogy a felhasználó regisztrál vagy bejelentkezik, majd rendelést ad le és szerkeszti a kosarat. A rendelés elkészítése után a felhasználó leadja a rendelést és nyomon követheti annak állapotát. Az admin menükezelést végez, kezeli a rendeléseket, bejelentkezik és megtekinti a helyzetet a térképen.



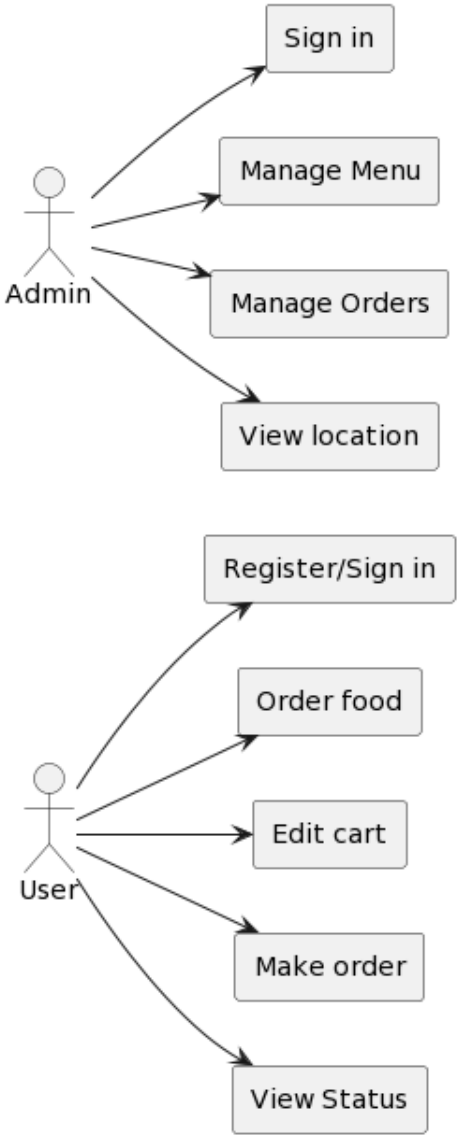
4.2. ábra. Sequence Diagram

4.3. Use Case Diagram

A Use Case Diagram egy grafikus eszköz, amely bemutatja, hogy a felhasználók (hogyan használják a rendszert és milyen funkciókat hajtanak végre. Az alábbi Use Case Diagram a TastyGo alkalmazás használati eseteit mutatja be:

Az Use Case Diagramon látható, hogy a vásárló (User) regisztrálhat az alkalmazásban, bejelentkezhet, böngészheti az ételeket, rendelést készíthet, lekérdezheti a rendelés

állapotát, módosíthatja a rendelést, megtekintheti az étel részletes leírását és véglegesítheti a rendelést. Az adminisztrátor (Admin) bejelentkezhet, kezelheti a menüt, kezelheti a rendeléseket és megtekintheti a helyzetet a térképen. Az alkalmazás a vásárlók és az adminisztrátor közötti interakciókat koordinálja és lehetővé teszi a kényelmes és hatékony ételrendelést.



4.3. ábra. Use Case Diagram

5. fejezet

Projekt felépítése

A projekt felépítése a következő képpen néz ki: Category: Ez az osztály tárolja az ételek kategóriáit, például pizza, pasta, soup stb. kategóriákat. A Category osztály hasznos, mert lehetővé teszi az ételek könnyebb csoportosítását és kategorizálását, Ezenkívül segít a felhasználóknak a kívánt kategóriák alapján böngészni az étlapot és könnyebben megtalálni a kívánt ételeket.

Food: Ez az osztály tárolja az ételek adatait, mint például az étel nevét, kalóriatartalmát, árát, leírását stb. A Food osztály fontos a projektben, mert lehetővé teszi az ételobjektumok kezelését, listázását és rendelését. Az ételeknek különféle tulajdonságai vannak, amelyekre szükség lehet a rendelés folyamatában, például az ár vagy a kalóriatartalom.

Order: Ez az osztály tárolja a rendelések adatait, mint például az összes rendelt étel, a szállítási cím, az ár stb. Az Order osztály segítségével kezelhetjük és nyomon követhetjük a felhasználók által leadott rendeléseket. Ez az osztály elengedhetetlen a házhozszállítási alkalmazásban, mivel lehetővé teszi a rendelések rögzítését, frissítését és kezelését.

Request: Ez az osztály tárolja a felhasználók által leadott rendelések kérését, mint például a rendelt ételek, a szállítási cím stb. A Request osztály segítségével kezelhetjük a felhasználók által küldött rendeléseket, és az éttermek vagy a kiszállítók számára elérhetővé teszi azokat. Ez lehetővé teszi a hatékonyabb kommunikációt és a rendelések sima folyamatát.

User : Ez az osztály tárolja a felhasználók adatait, mint például a felhasználónév, jelszó, telefonszám stb. A User osztály lehetővé teszi a felhasználói fiókok kezelését, beleértve a regisztrációt, bejelentkezést és adatmódosítást.

Az Activity-k az Android alkalmazások különböző képernyőinek reprezentációi, ezek az osztályok felelősek a felhasználói felület kezeléséért és az interakciók kezeléséért: CartActivity: Ez az Activity felelős a felhasználó kosarában található elemek kezeléséért. Itt a felhasználó megtekintheti és szerkesztheti a kosár tartalmát, például hozzáadhat vagy eltávolíthat elemeket.

FoodDetailActivity: Ez az Activity felelős egy adott étel részletes adatainak megjelenítéséért. Itt a felhasználó részletes információkat kaphat egy ételről, például az árát, leírását és képét. Továbbá lehetősége van az étel hozzáadására a kosarához vagy rendelésére.

FoodListActivity: Ez az Activity felelős az ételek listázásáért és megjelenítéséért. Itt a felhasználó áttekintheti az elérhető ételek listáját és különböző szűrők segítségével böngészheti azokat.

HomeActivity: Ez az Activity a főképernyő vagy kezdőképernyő az alkalmazásban. Itt a felhasználó áttekintheti az alkalmazás főbb funkcióit és navigálhat más Activity-k között.

MainActivity: Ez az Activity felelős az alkalmazás indításakor először megjelenő képernyőért. Általában itt történik az alkalmazás inicializálása és a felhasználó bejelentkezési/regisztrációs felületének megjelenítése.

OrderStatusActivity: Ez az Activity felelős a felhasználó rendelésének állapotának megjelenítéséért. Itt a felhasználó nyomon követheti a rendelése állapotát, például "feldolgozás alatt", "kiszállítás alatt" vagy "teljesítve".

SignInActivity: Ez az Activity felelős a felhasználó bejelentkezési felületének megjelenítéséért. Itt a felhasználó megadhatja a bejelentkezéshez szükséges adatokat, például felhasználónevet és jelszót.

SignUpActivity: Ez az Activity felelős a felhasználó regisztrációs felületének megjelenítéséért. Itt a felhasználó megadhatja az új fiókhoz szükséges adatokat, például felhasználónevet, e-mail címet és jelszót.

A ViewHolder mappában található osztályok olyan osztályok, amelyek segítenek az adatok megjelenítésében és kezelésében az alkalmazásban.

CartAdapter: Ez az osztály felelős a kosárban található elemek megjelenítéséért és kezeléséért. A CartAdapter segítségével az alkalmazás a kosárban található elemeket megfelelő módon jeleníti meg a felhasználói felületen. Továbbá lehetőséget nyújt az elemek szerkesztésére és eltávolítására.

CartViewHolder: Ez az osztály egy nézet (View) tartalmát tárolja a kosárban található elemek megjelenítéséhez. A CartViewHolder felelős az adatok megjelenítéséért és kezeléséért a felhasználói felületen. Például egy elem nevét, árát vagy mennyiségét tárolhatja.

FoodViewHolder: Ez az osztály egy nézet (View) tartalmát tárolja az ételek megjelenítéséhez. A FoodViewHolder felelős az ételek adatainak megjelenítéséért és kezeléséért a felhasználói felületen. Például egy étel nevét, leírását vagy képét tárolhatja.

MenuViewHolder: Ez az osztály egy nézet (View) tartalmát tárolja a menüelemek megjelenítéséhez. A MenuViewHolder felelős a menüelemek adatainak megjelenítéséért és kezeléséért a felhasználói felületen. Például egy menüelem nevét, ikonját vagy leírását tárolhatja.

OrderViewHolder: Ez az osztály egy nézet (View) tartalmát tárolja a rendelések megjelenítéséhez. Az OrderViewHolder felelős a rendelések adatainak megjelenítéséért és kezeléséért a felhasználói felületen. Például egy rendelés állapotát, címét vagy árát tárolhatja.

Ezek az osztályok jelentős segítséget nyújtanak az adatok hatékony és rendezett megjelenítésében az alkalmazásban. A ViewHolder mappában található osztályok felelősek a nézetek és adatok összekapcsolásáért, valamint a felhasználói felület kezeléséért. Ezáltal biztosítják a felhasználó számára egyértelmű és interaktív élményt az alkalmazás használata során.

A Common osztály a közös vagy általános funkciókat és adatokat tartalmazza, amelyekre az alkalmazás különböző részeiben szükség lehet. CurrentUser: Ez egy statikus User

objektum, amely az aktuális felhasználót tárolja. Az alkalmazás más részeiben használható, hogy hozzáférjen a bejelentkezett felhasználóhoz és annak adataihoz.

ConvertCodeToStatus(): Ez a metódus egy szöveges státuszértéket alakít át ember által olvasható formára. A metódus egy státusz kódot kap paraméterként, és visszatéríti a megfelelő szöveges státuszt. Például a "0" státuszkód esetén visszaadja a "Placed" szöveget.

isConnectedToInternet(): Ez a metódus ellenőrzi, hogy az eszköz csatlakozva van-e az internethez. Az alkalmazás kap egy Context objektumot paraméterként, és az erre a célra létrehozott ConnectivityManager segítségével ellenőrzi az internetkapcsolat állapotát. Ha csatlakozva van az internethez, akkor igaz értékkel tér vissza, egyébként hamissal.

DELETE: Ez egy konstans String, amelyet "Delete" értékkel inicializálnak. Az alkalmazás más részeiben használható a törlési műveletek jelzésére.

A Common osztály hasznos segédosztály az alkalmazásban, amely közös funkcionalitást és adatokat biztosít a különböző részek számára. Azt segíti elő, hogy a kód egy helyen legyen elérhető és újrafelhasználható legyen, ami hatékonyabb és karbantarthatóbb fejlesztést eredményez.

Ugyanakkor van egy Interface is, az ItemClickListener, amely egyetlen metódust tartalmaz: onClick(), amely 3 paramétert fogad. Az interfész célja, hogy lehetőséget biztosítson az eseménykezelőknek az elemekre történő kattintások figyelésére.

5.1. A szoftver bemutatása

Az Android TastyGo alkalmazás egy modern és felhasználóbarát megoldás az éttermi házhozszállításra. Az alkalmazás lehetővé teszi a vásárlók számára, hogy egyszerűen és kényelmesen rendeljenek ételt az okostelefonjukon keresztül.

Az alkalmazás teljes étlapot kínál, amelyben a vásárlók böngészhetnek, kategóriák szerint szűrhetnek és részletes információkat kaphatnak az ételekről, mint például a kalóriatartalom, az ár, a kedvezmény és a leírás.

A rendelési folyamat egyszerű és intuitív. A vásárlók könnyedén összeállíthatják a kosarukat és megadhatják a kiszállítási címet.

Az alkalmazás szolgáltatásai nemcsak a vásárlóknak nyújtanak előnyöket, hanem az éttermeknek is. Az éttermek széles választékban prezentálhatják étlapjukat, ahol az ételek képekkel és részletes leírásokkal vannak bemutatva.

Az alkalmazás segíti az éttermeket a rendelések hatékony kezelésében és a kiszállítások koordinálásában. A rendelések automatikusan beérkeznek a rendszerbe, ahol az éttermek áttekinthetik és frissíthetik azok státuszát.

A felhasználók számára az alkalmazás lehetőséget nyújt a rendelés állapotának követésére. Értesítéseket kapnak az aktuális helyzetekről, mint például a rendelés kiszállításra került vagy úton van.

Az Android TastyGo alkalmazás egyszerű és intuitív felhasználói felülettel rendelkezik, amely megkönnyíti a vásárlók számára a rendelési folyamatot és a navigációt az alkalmazásban.

5.2. A szoftver megírásához használt könyvtárak

A szoftver elkészítésénél szükségem volt néhány előre megírt osztálykönyvtárra, amelyek megkönnyítették a munkámat. Ezekről tudni kell, hogy nyílt forráskódúak, tehát bárki számára elérhetőek az interneten, továbbá azt is, hogy ezek is Java nyelvben íródtak, hasonlóan, mint az általam írt alkalmazás. A továbbiakban szeretném bemutatni ezeket a könyvtárakat és azt, hogy mire- és hogyan használtam fel őket.

- Android SDK (https://developer.android.com/studio):
  - Android API-k: Az Android API-k olyan interfészek és osztályok gyűjteménye, amelyek lehetővé teszik az alkalmazások számára, hogy hozzáférjenek az Android operációs rendszer funkcióihoz. Ezek az API-k különböző kategóriákba sorolhatók, mint például a felhasználói felület, hálózat, adatbázis, szenzorok stb
  - Az Android SDK tartalmaz olyan eszközöket, amelyek segítenek az alkalmazások fejlesztésében és tesztelésében. Ezek közé tartozik az Android Studio, amely az Android alkalmazások integrált fejlesztői környezete, valamint a szimulátorok, emulatorok, debugger és profilozó eszközök.
  - Kiterjesztett könyvtárak: Az Android SDK számos kiterjesztett könyvtárat tartalmaz, amelyek megkönnyítik az alkalmazások fejlesztését. Ilyen könyvtárak például a Retrofit, Gson, Picasso, RecyclerView stb. Ezek a könyvtárak további funkciókat és lehetőségeket biztosítanak az alkalmazásokhoz, például hálózati kommunikáció, adatkezelés, képfeldolgozás stb.
  - Emulátorok és tesztelési eszközök: Az Android SDK lehetővé teszi az alkalmazások tesztelését és hibakeresését különböző emulátorok és fizikai eszközök segítségével. Ez lehetővé teszi a fejlesztők számára, hogy valós környezetben teszteljék az alkalmazást és ellenőrizzék annak kompatibilitását különböző Android verziókkal és eszközökkel.
- Firebase (https://www.crane.hu/amit-a-google-firebase-rol-tudni-erdemes):
  - Valósidejű adatbázis: A Firebase Realtime Database lehetővé teszi az alkalmazások számára, hogy valós időben kommunikáljanak és szinkronizáljanak adatokat a felhőben. Ez nagyon hasznos lehet például chat alkalmazásokban vagy valós idejű frissítéseket igénylő funkciókban.
  - Felhasználókezelés: A Firebase Authentication szolgáltatás segítségével egyszerűen kezelhetjük a felhasználók bejelentkezését és regisztrációját. Támogatja az e-mail, jelszó, Google, Facebook és más bejelentkezési lehetőségeket, és biztonságosan kezeli a felhasználói azonosítókat.
  - Felhőtárolás: A Firebase Cloud Storage lehetővé teszi a felhasználók számára, hogy fájlokat tároljanak és megosszanak a felhőben. Ez kényelmes lehet például képek, videók vagy dokumentumok feltöltésére és megosztására az alkalmazásba
- Retrofit

- Kényelmes API-kommunikáció: A Retrofit egyszerű és kényelmes módot kínál az alkalmazások számára a webszolgáltatásokhoz történő kommunikációhoz. Az API hívások egyszerűen megadhatók és kezelhetők a Retrofit által biztosított annotációkkal és metódusokkal.
- Adatkonverzió: A Retrofit automatikusan kezeli az adatkonverziót a kliens és a szerver között. A könyvtár lehetővé teszi a különböző adatformátumok (például JSON, XML) használatát, és képes az objektumok közötti könnyű átalakításra.
- Hálózati forgalom kezelése: A Retrofit beépített hálózati forgalomkezeléssel rendelkezik, amely segít az adatok hatékony kezelésében. Ez magában foglalja az adatok gyorsítótárazását, az adatforgalom optimalizálását és a hálózati hibák kezelését.
- Interceptorok támogatása: Az interceptorok lehetővé teszik az alkalmazás számára, hogy befolyásolja és módosítsa a hálózati kéréseket és válaszokat. A Retrofit támogatja az interceptorokat, amelyek rugalmasságot biztosítanak az adatkérések módosításához vagy a hálózati forgalom logolásához.
- Picasso
  * Kép letöltése: A Picasso lehetővé teszi a képek letöltését a hálózatról, legyen szó HTTP vagy HTTPS protokollról. Egyszerűen megadhatod a kép URL-jét, és a könyvtár automatikusan elvégzi a letöltést.
  * Gyorsítótárkezelés: A Picasso automatikusan kezeli a képek gyorsítótárazását, ami azt jelenti, hogy letöltés után a képek tárolódnak a gyorsítótárban. Ez lehetővé teszi, hogy a következő alkalommal, amikor a kép megjelenítésre kerül, nem kell újra letölteni, hanem a gyorsítótárból kerül beolvasásra
  * Képméretezés és méretező arány: A Picasso lehetővé teszi a képek méretezését és az aránytartás beállítását a megjelenítés során. Így könnyedén illesztheted a képeket a kívánt nézethez és felülethez.
  * Helyi képek kezelése: A könyvtár nemcsak a hálózatról történő letöltést és megjelenítést teszi lehetővé, hanem kezeli a helyi tárolóban lévő képeket is. Tehát helyi fájlok, például a készülék memóriájában vagy SD-kártyán található képek is könnyedén megjeleníthetők a Picasso segítségével.
- Google Maps Android API
  * Térképek megjelenítése: A Google Maps API lehetővé teszi a térképek megjelenítését az alkalmazásban. Az API segítségével beállíthatjuk a térkép pozícióját, nagyítását, forgatását és döntését, valamint megjeleníthetünk címkéket, útvonalakat és más térképes elemeket.
  * Helymeghatározás: Az API támogatja a felhasználók aktuális tartózkodási helyének meghatározását a GPS vagy hálózati helymeghatározás segítségével. Ez lehetővé teszi az alkalmazás számára, hogy például megjelenítse a felhasználó aktuális pozícióját a térképen vagy keresse meg a közelben lévő helyeket.

* Helyek keresése és kijelölése: Az API segítségével keresni lehet helyeket, például címeket, városokat vagy nevezetes helyeket. Az alkalmazás könnyedén megjelenítheti ezeket a helyeket a térképen, vagy interaktív módon lehetővé teheti a felhasználók számára, hogy kijelöljenek helyeket a térképen.
* Útvonaltervezés: Az API lehetővé teszi az útvonaltervezést két vagy több pont között. Az alkalmazás megadhatja a kiinduló és célhelyeket, valamint a köztes pontokat, és megjelenítheti az optimális útvonalat a térképen. Ezenkívül az API segítségével információkat is lekérdezhetünk az útvonal hosszáról, időtartamáról és közlekedési információkról.

Összefoglaló

A TastyGo alkalmazás nemcsak kényelmes és hatékony házhozszállítást biztosít, hanem számos további előnnyel jár mind az éttermek, mind a vásárlók számára. Az alkalmazás intuitív felhasználói felületének köszönhetően könnyedén böngészhető az étterem teljes étlapja, ahol részletes leírások és képek segítik a vásárlókat a döntésben. Emellett az alkalmazás figyelembe veszi a diétás és allergiás igényeket is, így rugalmas rendelési lehetőségeket kínál.

A vásárlók számára a TastyGo alkalmazás egyszerű és gyors rendelési folyamatot biztosít. Az étel kiválasztása után könnyedén hozzáadható a kosárhoz, és a különböző fizetési módok közül választhatnak. Az alkalmazás emellett lehetőséget nyújt a rendelési előzmények és kedvenc ételek mentésére, ami még kényelmesebbé teszi a rendelést.

Az éttermek számára a TastyGo alkalmazás nagyobb láthatóságot és üzleti növekedést jelent. Az online jelenlét révén az éttermek elérhetik az online vásárlókat, és bemutathatják teljes étlapjukat, valamint különleges ajánlataikat. A rendeléskezelés is egyszerű és hatékony a szerveroldalon, ahol rögzíthetik, áttekinthetik és frissíthetik a rendeléseket, minimalizálva a hibák lehetőségét.

A kiszállítók számára a TastyGo alkalmazás optimalizált szállítási folyamatot biztosít. A térképes megjelenítés segítségével a kiszállítók könnyen kiválaszthatják az optimális útvonalakat és hatékonyan kezelhetik a szállítási helyeket. Az alkalmazás nyomon követi a kiszállítók tevékenységét és lehetőséget nyújt a teljesítményük értékelésére, ezzel növelve a hatékonyságot és minőséget.

Az alkalmazás felhasználja a Java programozási nyelvet, ami egy népszerű és rugalmas nyelv az Android alkalmazásfejlesztésben. A Java nyelv lehetővé teszi a hatékony és megbízható alkalmazások fejlesztését, valamint könnyen integrálható más technológiákhoz és könyvtárakhoz.

Összességében a TastyGo alkalmazás átalakítja és javítja az éttermi házhozszállítási élményt, mind a vásárlók, mind az éttermek, mind a kiszállítók számára. Az intuitív felület, a széles választék, a kényelmes rendelési folyamat és az optimális szállítás mind hozzájárulnak egy kellemesebb és hatékonyabb élményhez, amely a vendéglátóipar szereplőinek jobb üzleti eredményeket és a fogyasztók elégedettségét eredményezi.

Ábrák jegyzéke

Irodalomjegyzék

1. „Comprehensive Study and Technical Overview of Application Development in iOs Android and Windows Phone8” Anuja H. Vaidya, Sapan Naik

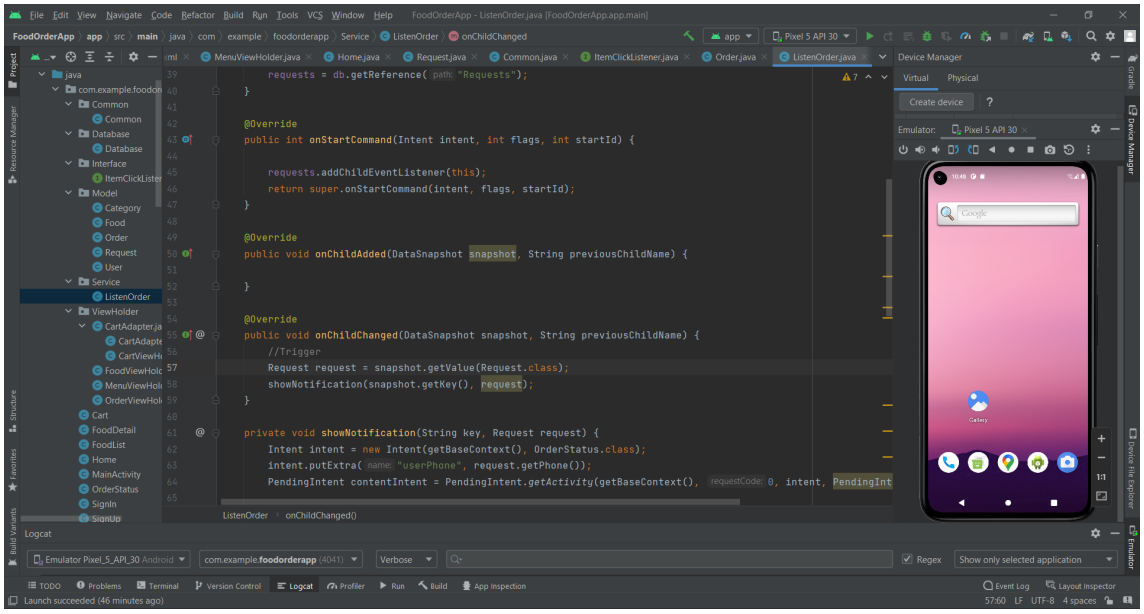
2.„User Data Management System Using Firebase Cloud” Roshan R.Kolte, Ekansh Moundekar 3.„Location Based Reminder Using Android and Google Maps” Neelu Lalband

4.„Development of an Android Grocery Checklist Application” Ana Antoniette C. Illahi Gershom Rob Narag, Manuel Lorenzo Parro, Luis Paolo Wenceslao

5.Android Developers: https://developer.android.com/about

Függelék

F.1. Android Studio felülete



F.1.1. ábra. AndroidStudio.