

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



**SAPIENTIA
ERDÉLYI MAGYAR
TUDOMÁNYEGYETEM**

Lakberendezési tárgyak vásárlása és 3D-s szobában való
megjelenítése, telefonos applikációban

DIPLOMADOLGOZAT

Témavezető:

Dr. Iclănanz Dávid Andrei,
Egyetemi docens

Végzős hallgató:

Mátyus-Borka Erzsébet

2023

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



**UNIVERSITATEA
SAPIENTIA**

Achiziționarea obiectelor de mobilier și afișarea acestora într-o cameră virtuală 3D prin intermediul unei aplicații mobile

LUCRARE DE DIPLOMĂ

Coordonator științific: Dr. Iclănzan Dávid Andrei, Conferențiar universitar
Absolvent: Mátyus-Borka Erzsébet
2023

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



SAPIENTIA
HUNGARIAN UNIVERSITY
OF TRANSYLVANIA

Visualizing furnishings in 3D room within a mobile application

BACHELOR THESIS

Scientific advisor: Dr. IclăNZan Dávid Andrei,
Associate professor Student: Mátyus-Borka Erzsébet

2023

Declarație

Subsemnatul/a MATYUS-BORKA ERZSEBET, absolvent(ă) al/a specializării INFORMATICĂ, promoția 2023 cunoscând prevederile Legii Educației Naționale 1/2011 și a Codului de etică și deontologie profesională a Universității Sapientia cu privire la furt intelectual declar pe propria răspundere că prezenta lucrare de licență/proiect de diplomă/disertație se bazează pe activitatea personală, cercetarea/proiectarea este efectuată de mine, informațiile și datele preluate din literatura de specialitate sunt citate în mod corespunzător.

Localitatea, Tg-Mureș
Data: 16.06.2023.

Absolvent

Semnătura.....Borka.....

LUCRARE DE DIPLOMĂ

Coordonator științific:
dr. Iclanțan David

Candidat: **Mátyus-Borka Erzsébet**
Anul absolvirii: 2023

a) Tema lucrării de licență: Dezvoltarea unei aplicații pe mobil pentru un magazin de mobilă online, care integrează o funcție de cameră cu realitate augmentată pentru a permite utilizatorilor să observe și să vizualizeze produsele într-un mod tridimensional, îmbunătățind astfel experiența de cumpărare și reducând riscul returnurilor.

b) Problemele principale tratate:

- Studiu bibliografic privind evoluția cumpărăturilor online și impactul lor asupra comerțului tradițional.
- Studiu bibliografic și analiza experienței utilizatorului în cumpărăturile online și factorii care contribuie la o experiență de succes.
- Studiul literaturii de specialitate în cea ce privește tehnologiile moderne utilizate în cumpărăturile online, cum ar fi aplicațiile mobile și funcționalitățile lor.
- Examinarea aspectelor de utilizabilitate și interfață utilizator pentru a crea o experiență prietenoasă și intuitivă.
- Proiectarea, prototipizarea, designul și implementarea unei aplicații mobile de cumpărături online, cu accent pe interfața utilizatorului și funcționalitățile relevante.
- Integrarea funcției de afișare a mobilierului 3D cu ajutorul tehnologiilor de realitate augmentată în aplicația de cumpărături online și evaluarea impactul acestora asupra experienței utilizatorului și proghnoza cu ajutorul unui chestionar a reducerii riscului returnurilor.
- Studiul și analiza rezultatelor obținute prin utilizarea aplicației mobile și a chestionarului completat în ceea ce privește satisfacția utilizatorilor și eficiența procesului de cumpărături online.
- Documentarea adecvată a stadiilor de proiectare, implementare și testare a aplicațiilor.

c) Desene obligatorii:

- Schema bloc a sistemului
- Diagrame de proiectare, implementare și testare pentru aplicația software realizată.

d) Softuri obligatorii:

- Aplicație mobilă de magazin online și cumpărături cu funcție de afișare augmentată a mobilierului selectat în spațiul utilizatorului.

e) Bibliografia recomandată:

Bonetti, F., Warnaby, G., & Quinn, L. (2018). Augmented reality and virtual reality in physical and online retailing: A review, synthesis and research agenda. *Augmented reality and virtual reality: Empowering human, place and business*, 119-132.

Cruz, E., Orts-Escalano, S., Gomez-Donoso, F., Rizo, C., Rangel, J. C., Mora, H., & Cazorla, M. (2019). An augmented reality application for improving shopping experience in large retail stores. *Virtual Reality*, 23, 281-291.

Kumar, T. S. (2021). Study of retail applications with virtual and augmented reality technologies. *Journal of Innovative Image Processing (JIIP)*, 3(02), 144-156.

David, A., Senn, W. D., Peak, D. A., Prybutok, V. R., & Blankson, C. (2021). The value of visual quality and service quality to augmented reality enabled mobile shopping experience. *Quality Management Journal*, 28(3), 116-127.

f) Termene obligatorii de consultații: săptămânal, preponderent online

g) Locul și durata practicii: Universitatea „Sapientia” din Cluj-Napoca,
Facultatea de Științe Tehnice și Umaniste din Târgu Mureș, sala / laboratorul 413

Primit tema la data de: 25.04.2022

Termen de predare: 06.07.2023

Semnătura Director Departament

Semnătura responsabilului
programului de studiu

Semnătura coordonatorului

Semnătura candidatului

Kivonat

Az online vásárlás napjainkban fénykorát éli. A vásárlók igyekeznek kényelmes megoldásokat választani otthonaik berendezésére, amelynek első lépése a bútorok megvásárlása. Az elmúlt években az online felületeken történő vásárlás egyre nagyobb előnyt élvez a fizikai kontaktussal történő vásárlással szemben. Ennek megvalósítására számos lehetőség áll rendelkezésre az érdeklődőknek, a weboldalakon publikált áruházaktól kezdve akár a telefonbeszélgetésen keresztül történő megrendelésen keresztül, egészen a konkrétan erre a célra fejlesztett mobilos applikációig.

Ezeket a szempontokat figyelembe véve egy olyan tematikát választottam dolgozatomnak, amely az aktuális trendekbe inkadrálódik, mégis hiánypótló, modern technológiákat használ a megvalósításra illetve egy kis csavarral újdonságot nyújt a felhasználói élmény növelése érdekében.

A dolgozatom egy online bútoráruházat valósít meg, egy Android applikáció keretein belül. Az alkalmazás kényelmes és könnyen használható felületet biztosít úgy az eladónak, aki az admin szerepkörét tölti be, mint a vásárlónak, aki pedig felhasználóként szerepel.

A felhasználói felület design szempontjából letisztult, könnyen értelmezhető és kezelhető mindenki számára, ugyanakkor ellát minden szükséges funkciót, amely a zökkenőmentes vásárláshoz szükséges. A felhasználó saját fiókot hozhat létre, amelyen keresztül belépve az alkalmazásba kedvére nézelődhet, illetve a vásárlás folyamatát is véghez viheti. Az online vásárlás talán egyetlen hátrányaként a fizikai jelenlét hiányából fakadó megfigyelés nehézségét említhetjük. Míg a fizikai vásárlás esetében a megfigyeléshez minden érzékszervünket használva válogathatunk, megtapinthatjuk, elforgathatjuk, minden szögből megtekinthetjük az adott árucikket, addig a webshopok általában ezekre a problémákra nem kínálnak megoldást.

Projektemben ezt a hátrányt igyekeztem kiküszöbölni, azáltal, hogy elérhetővé tettem egy virtuális szoba funkciót, amely segítségével a vásárló nem csupán egy képet tekinthet meg a megvásárolni kívánt termékről, illetve annak leírását olvashatja el, hanem 3D-ben is megfigyelheti, elforgathatja, elhelyezheti a szobában méretarányosan. Ez a funkció amellett, hogy fokozza a vásárlási élményt, csökkenti a visszaküldés kockázatát, hiszen alaposabb megfigyelés lehetőségét kínálja a vásárlónak.

Rezumat

Cumpărăturile online se bucură de perioada de glorie în aceste zile. Clienții încearcă să aleagă soluții confortabile pentru mobilarea locuințelor, primul pas fiind achiziționarea de mobilier. În ultimii ani, cumpărăturile pe platformele online au câștigat un avantaj din ce în ce mai mare față de cumpărăturile cu contact fizic. Există o serie de opțiuni disponibile părților interesate pentru a realiza acest lucru, de la magazinele publicate pe site-uri web până la comenzi telefonice, până la aplicații mobile dezvoltate special pentru acest scop.

Tinând cont de aceste aspecte, am ales o temă pentru teza mea care se încadrează în tendințele actuale, dar care folosește tehnologii moderne pentru a umple golarile și oferă ceva nou cu o mică întorsătură pentru a crește experiența utilizatorului.

Teza mea implementează un magazin de mobilă online în cadrul unei aplicații Android. Aplicația oferă o interfață convenabilă și ușor de utilizat atât pentru vânzător, care joacă rolul de administrator, cât și pentru cumpărător, care acționează ca utilizator.

Interfața cu utilizatorul este curată din punct de vedere al designului, ușor de înțeles și de utilizat pentru toată lumea și, în același timp, oferă toate funcțiile necesare pentru o experiență de cumpărături fără probleme. Utilizatorul își poate crea propriul cont, prin care poate naviga prin aplicație, precum produse la care se poate întoarce ulterior, și poate finaliza procesul de cumpărare.

Poate singurul dezavantaj al cumpărăturilor online este dificultatea monitorizării din cauza lipsei prezenței fizice. În timp ce în cazul cumpărăturilor fizice, ne putem folosi de simțurile pentru a observa, selecta, atinge, roti și vizualiza produsul dat din orice unghi, magazinele online de obicei nu oferă soluții la aceste probleme.

În proiectul meu, am încercat să elimin acest dezavantaj punând la dispoziție o funcție de cameră virtuală, cu ajutorul căreia clientul nu poate doar să vizualizeze o poză a produsului pe care dorește să-l cumpere, sau să citească descrierea acestuia, ci și să îl observe, să îl rotească și să îl plaseze în camera în 3D proporțional. Pe lângă îmbunătățirea experienței de cumpărături, această funcție reduce riscul returnurilor, întrucât oferă clientului posibilitatea unei monitorizări mai amănunțite.

Abstract

Online shopping is thriving in today's world. Customers strive to choose convenient solutions for furnishing their homes, and the first step is purchasing furniture. In recent years, online shopping has gained a significant advantage over in-person shopping with physical contact. To achieve this, there are numerous options available to interested individuals, ranging from online stores published on websites to ordering via phone calls and specifically developed mobile applications.

Taking these aspects into account, I have chosen a theme for my project that aligns with current trends while also incorporating innovative technologies to enhance the user experience. My project implements an online furniture store within an Android application. The app provides a convenient and user-friendly interface for both the seller, who fulfills the role of an administrator, and the buyer, who acts as a user.

From a design perspective, the user interface is clean, easily understandable, and manageable for everyone, while also offering all the necessary functions for a seamless shopping experience. Users can create their own accounts, allowing them to browse the application at their leisure, favorite products for future reference, and complete the purchase process.

Perhaps the only drawback of online shopping is the difficulty in physically observing the products due to the absence of physical presence. While physical shopping allows us to use all our senses to select and examine items from every angle, touch them, and rotate them, online shops generally do not offer solutions to these issues.

In my project, I have endeavored to overcome this drawback by implementing a virtual room feature that allows buyers to not only view images and read descriptions of desired products but also observe them in 3D, rotate them, and place them proportionally within a room. This functionality enhances the shopping experience and reduces the risk of returns by offering customers the opportunity for more thorough observation.

Tartalomjegyzék

1. Bevezető	11
2. Programok, technológiák bemutatása	13
2.1. Git + GitHub	13
2.1.1. GitHub	14
2.2. Postman	14
2.3. Visual Studio 2022	15
2.4. .Net Framework	15
2.5. SQL Server Management Studio	16
2.6. Kotlin	16
2.7. ARCore	17
2.8. Android	20
3. Adatbázis	21
3.1. MSSQL	21
3.2. Az adatbázis struktúrája	22
4. Funkcionális követelmények	24
4.1. Use Case Diagram	24
4.1.1. Aktorok - Szereplők	24
4.1.2. Use casek - Használati esetek	25
5. Felhasználói követelmények	27
5.0.1. Cél és áttekintés	27
5.0.2. Felhasználói csoportok	27
5.0.3. Felhasználói interfész	27
6. Nem funkcionális követelmények	30
7. Szerver	31
7.1. .NET Core WebAPI	31
7.2. API végpontok	31
8. Szerver arhitekúra	36
8.1. Konkrét megvalósítások	36
8.1.1. Táblák létrehozása	36
8.1.2. Servicek	37

8.1.3. Controllerek	37
8.1.4. Specifikus controllerek és servicek	38
9. Android	41
9.0.1. Arhitektúra	41
9.0.2. Fontosabb osztályok és könyvtárak	41
9.0.3. Konkrét megvalósítások	41
10. Felhasználói felület	43
10.1. Tervezés	43
10.2. Megvalósítás	43
11. Az alkalmazás működésének bemutatása képekkel	46
12. Továbbfejlesztési lehetőségek	53
Összefoglaló	54
Ábrák jegyzéke	56
Irodalomjegyzék	57

1. fejezet

Bevezető

Napjainkban az emberek egyre inkább arra törekednek, hogy a rohamos ütemben fejlődő tehnológiát arra használják fel, hogy az élet minél több területén megkönnyítse az ember munkáját. Lehet szó akár mesterséges intelligenciáról, amely az ember keze alá dolgozva tud segíteni számos területen, vagy akár a bizonyos tevékenységek automatizálásáról, amely az emberi erőforrásokat helyettesítve nyújtanak előnyt, viszont az átlag emberek életében az internet, illetve az online világ által nyújtott lehetőségek a legelérhetőbbek, illetve a hasonló célt ellátó mobil alkalmazások felhasználása is széles körben elterjedté vált.

Társadalmunkról általánosságban elmondható, hogy igyekszik minél kényelmesebben berendezkedni a minden nap életbe, időt és energiát spórolva ott, ahol csak tud. Erre tökéletes eszközként szolgál a rengeteg megoldás, amelyet a technika nyújt, az élet szinte minden területén, az által, hogy számos funkciót el tud látni az ember már otthonról is, egyszerűen egy internetes hozzáféréssel. Egyre gyakoribb napjainkban az otthonról történő munkavégzés, hiszen rengeteg iparág igyekszik áttérni az online terekbe. Ugyanakkor az ügyintézés is sok esetben lehetséges már online felületen, vagy applikáción keresztül, legyen szó itt akár hivatalos dokumentumokkal kacsolatban történő eljárásról, egyetemi vagy iskolai beíratkozáról, vagy akár csak egy egyszerű parkolási díj befizetéséről. Ezek a folyamatok az ember minden napjaiban időt megspórolva, egyszerűen és a legtöbbek számára elérhetően történik.

Az internet elterjedésével az emberek vásárlási szokásai is gyökeresen megváltoztak, hiszen az online felületek úgy a fogyasztóknak, mint a termelőknek kézenfekvő működést biztosítanak. Az online vásárlás rohamos elterjedése olyannyira befolyásolta a termelői piacot, hogy már nehéz lenne olyan terméket, vagy akár szolgáltatást megnevezni, amely ne lenne elérhető, rendelhető, de legalábbis foglalható az interntetes oldalak, vagy applikációk felhasználásával. Ebbe a téma körbe beleérthetünk egészen egyszerű dolgokat is, mint például az élelmiszerek házhoz rendelése, amely különös népszerűségenk örvendett a közelmúltban, a COVID-19 járvány terjedésének idején és mint szokás, teljesen észrevétlenül beleépült az emberek minden napjaiba. Az online vásárlás nem csak az egyszerű, minden nap termékekkel kapcsolatban lett népszerű, hanem az egyet nagyobb árucikkek is teret nyertek az internetes piacon, ide értve akár ingatlanokat, járműveket, vagy bútorokat is.

Mivel az internetes vásárlás elterjedésének legnagyobb oka a kényelem volt, ezért nem okoz meglepetést, hogy úgy a számítógépeken illetve laptopokon használt webol-

dalak, mint a telefonok esetében előnyös applikációk is igyekeznek lehetőséget kínálni a forgalmazóknak és a fogyasztóknak, hogy minél egyszerűbbé és elérhetővé tegyék a termékek forgalmazását.

A termelők szemszögéből nézve a kényelem mellett talán a legnagyobb előny, amely indokolttá teszi a virtuális felületeken való árusítást, az a globális piacra való betörés lenne. Azáltal, hogy a fizikai kontaktust nélkülözhetővé tesszük, lehetőséget nyújtunk arra is, hogy az adás-vétel folyamata a világ bármely két pontja között végbemehessen. Ezáltal a forgalmazó sokkal tágabb közönséget szólíthat meg, így már csak a sikeres és hatásos marketingre illetve reklámokra van szüksége ahoz, hogy sikeres vállalkozást tudhasson magáénak.

Ugyanezkről az előnyökről számolhat be a vásárló is, a saját szemszögéből, hiszen azáltal, hogy nem szükséges fizikai tereket körbejárnia a megfelelő termék kiválasztásának folyamata során, a befektető szabadon válogathat, egyrészt akár a saját tartózkodási helyének közelében található több áruházlánc kínálatából, anélkül, hogy akár egyetlen üzletet is körbe kellene járnia, másrészt pedig a föld bármely táján gyártott vagy forgalmazott termékek közül is válogathat, sokkal nagyobb piacot meglátogatva, csupán néhány kattintással. Ezáltal a vásárlás folyamata nem korlátozódik a helyi üzletek kínálatára.

Fontos kiemelnünk azt, hogy a virtuális vásárlás előnyt jelent a kistermelőknek, hiszen az online térben könnyebbenelfedezhetőek és fel tudják venni a versenyt akár a nagyobb áruházláncokkal is. A rengeteg előny mellett számos kihívással is szembe kell nézniük a vásárlóknak. Azokban a vásárlókban, akiknek ismerős lehet az adatlopás, adat-halászat fogalma, felmerülhet egy bizalmi kérdés az adott forgalmazó fizetési rendszereivel szemben, ezért kiemelt figyelmet kell fektetni az adatvédelemre, biztonságra.

Egy másik hátrány, amivel szembesülhet a vásárló, az akár az a pont is lehet, amelyet fennebb még előnyként említtettem, mégpedig a fizikai kontaktus hiánya. Ennek az árnyoldala befolyásolhatja a vásárlót, hiszen sokak számára szükséges a kézzel foghatóság a beruházás előtt, a méret, anyag, minőség és egyéb szempontok alapján történő megfelelő kiválasztásban.

Dolgozatom egy mobil applikáció fejlesztését foglalta magába, amely egy bútoráruházat valósít meg. A projekt különlegessége abban áll, hogy a fent említett felhasználói hátrányt, a való életben történő fizikai megfigyelés hiányát kiaknázza, azáltal, hogy rendelkezésre áll egy virtuális szoba használata, amelyben lehetőség van 3D-ben elhelyezni az árucikkeket, illetve a kamera használatával a saját környezetében is elhelyezheti a tárgyat, mindeneket a könnyebb megfigyelés, valósághűség, méretarányosság, illetve a több szögből való megtekintés lehetőségének érdekében.

2. fejezet

Programok, technológiák bemutatása

A projekt létrehozása során számos fejlesztői eszközre, platformra, fejlesztői környezetre, kezelőfelületre illetve keretrendszerre volt szükségem, amelyek nagyban megkönnyítették munkámat, vagy elengedhetetlen volt a használatuk a projekt megvalósításához. A következő részekben ezekről lesz szó röviden, használatuk illetve szükségességük megértése érdekében.

2.1. Git + GitHub

A Git egy nyílt forráskódú, elosztott verziókezelő szoftver, vagy másnéven egy szoftverforráskód-kezelő rendszer, amely a sebességre helyezi a hangsúlyt. minden Git munkamásolat egy teljes értékű repository teljes verziótörténettel és teljes revíziókövetési lehetőséggel, amely nem függ a hálózat elérésétől vagy központi szervertől.

A Git egy verziókezelő, arra szolgál, hogy fileok (programok, dokumentációk, stb) különböző verzióit kordában tartsa, elkönyvelje, tárolja és megossza. Git annyit csinál, hogy amikor azt mondjuk neki (commit), akkor egy directoryről csinál magának egy helyi adatbázist a .git nevű könyvtárba. Ezekkel az adatbázisokkal nyomon tudja követni, hogy mikor hogyan változott a könyvtárunk, vissza tudja állítani bármelyik korábbi (commit-olt) állapotát a könyvtárnak, illetve szinkronizálni tud egy másik gépen levő hasonló könyvtárral, közben intelligensen átvezeti a változásokat, illetve jelez, ha nem megy neki.



2.1. ábra. Git

2.1.1. GitHub

A GitHub egy olyan felhőplatform, amely a Gitet használja alapvető technológiaként. Leegyszerűsíti a projekteken való együttműködés folyamatát, és egy webhelyet, parancssori eszközöket és egy átfogó folyamatot biztosít, amelyekkel a fejlesztők és a felhasználók együttműködhettek. A GitHub a Git szakaszában korábban említett „távoli adattárként” működik.



2.2. ábra. GitHub

2.2. Postman

Postman egy fejlesztői eszköz és egyben platform, amely lehetővé teszi a webes API-k tesztelését, dokumentálását és megosztását. A Postman lehetővé teszi fejlesztőknek és API-tervezőknek, hogy könnyedén kommunikáljanak a különböző szerverekkel és azok API-ival.



2.3. ábra. Postman

A Postman számos célra hasznos eszköz a fejlesztők számára: API tesztelés, A Postman lehetővé teszi a webes API-k tesztelését és ellenőrzését. Könnyedén küldhetsz kéréseket az API-végpontokhoz, és megkapod a válaszokat. Ez segít az API-k helyes működésének ellenőrzésében, hibák felderítésében és hibakeresésben.

Kérések létrehozása és megosztása: A Postman segítségével egyszerűen létrehozhatsz és konfigurálhatsz HTTP kéréseket, például GET, POST, PUT, DELETE stb. A kérésekhez paramétereket, fejléceket és testet adhatsz hozzá. Ezeket a kéréseket később megoszthatod más fejlesztőkkel vagy csapatokkal.

2.3. Visual Studio 2022

A Visual Studio egy teljes körű fejlesztői környezet (IDE), amelyet a Microsoft fejlesztett ki. Ez egy szoftvercsomag, amely különböző eszközöket, funkciókat és szolgáltatásokat kínál a szoftverfejlesztőknek az alkalmazások létrehozásához, teszteléséhez, hibakereséséhez és üzemeltetéséhez.

A Visual Studio 2022 egy fejlesztői környezet, amelyet a Microsoft készített a szoftverfejlesztők számára. Ez az új verzió a korábbi Visual Studio verziók fejlesztésén alapul, és számos új funkciót, fejlesztést és javítást tartalmaz. A Visual Studio 2022 támogatja a .NET Framework-öt, valamint a .NET Core és az újgenerációs .NET 5+ keretrendszereket is.



2.4. ábra. Visual Studio 2022

.NET fejlesztési támogatás: A Visual Studio 2022 támogatja a .NET Framework-öt és a legújabb .NET Core és .NET 5+ keretrendszereket is. Ez lehetővé teszi a modern .NET alkalmazások fejlesztését és futtatását, valamint a régebbi .NET alapú projektek karbantartását.

A Visual Studio nagyon népszerű a fejlesztők körében, mivel egy egységes és kényelmes környezetet biztosít az alkalmazásfejlesztéshez, függetlenül attól, hogy milyen platformon vagy nyelven dolgoznak.

2.4. .Net Framework

.NET Framework lehetőséget biztosít a .NET Web API fejlesztéséhez is. A Web API egy olyan alkalmazásprogramozási felület (API), amely lehetővé teszi a kommunikációt és az adatcsere-t az ügyfél és a szerver között webes alkalmazásokban.



2.5. ábra. .NET Framework

A .NET Framework számos előnyt és hasznos jellemzőt kínál a fejlesztőknek, amik segítenek a hatékony és megbízható Web API-k fejlesztésében. Főbb jellemzői:

Könnyű és hatékony fejlesztés: A .NET Framework számos előre elkészített osztálykönyvtárt és beépített funkcionálitást biztosít a Web API fejlesztéséhez. Ez lehetővé teszi a gyorsabb és hatékonyabb kódolást, mivel nem kell minden részletet saját magadnak megírnod. Az osztálykönyvtárak széles választékban tartalmazzák az adatkezelést, az autentikációt, az autorizációt, a hibakezelést és más gyakori feladatokat.

.NET fejlesztési támogatás: A Visual Studio 2022 támogatja a .NET Framework-öt és a legújabb .NET Core és .NET 5+ keretrendszeret is. Ez lehetővé teszi a modern .NET alkalmazások fejlesztését és futtatását, valamint a régebbi .NET alapú projektek karbantartását.

2.5. SQL Server Management Studio

Az SSMS egy olyan szoftver, amely magába foglalja úgy a szkripteszközöket, mint a grafikus eszközöket, amelyek a szerver objektumaival és azok tulajdonságaival dolgoznak. Az SQL Server egyik felügyelő eszköze, lekérdezésekhez, tervezésekhez használják személyi számítógépen, vagy felhőn keresztül. A Microsoft lehetővé tette a visszafele kompatibilitást, így az SSMS újabb verzióin keresztül is elérhetjük az SQL Server példányok régebbi verzióit. A 11-es verziótól kezdődően az applikáció 2010-es Visual Studio héjon alapszik, WPF-et használva felhasználói felületként. A 18-as verzió pedig Visual Studio 2017 izolált héjra alapszik.

2.6. Kotlin

A Kotlin egy statikusan tipizált, objektumorientált programozási nyelv, amely inter-operábilis a Java virtuális géppel (JVM), a Java osztálykönyvtárrakkal és az Androiddal. A Kotlin programozási nyelvet eredetileg a Java programozási nyelv továbbfejlesztésére terveztek, és gyakran használják a Javával együtt. Annak ellenére, hogy a Kotlin az Android preferált fejlesztői nyelve, a Java-val való átjárhatósága miatt számos alkalmazástípushoz használják. A Kotlin egy általános célú fejlesztői nyelv, amelyet elsősorban Android mobilalkalmazások fejlesztéséhez használnak. Az Android-alkalmazások mellett a Kotlin a következőkben is hasznos: Szerveroldali fejlesztés. A back-end webalkalmazás-fejlesztés hagyományosan Java-t használ. A Kotlin a Java mellett használható a szer-veroldali fejlesztéshez. A Kotlin támogatja a Java osztálykönyvtárakat. Full-stack webes



2.6. ábra. SQL Server management Studio

fejlesztés. A fejlesztők a Kotlin for JavaScriptet használják a Kotlin kódsorok JavaScript-re történő lefordítására a front-end webes fejlesztéshez. Ez a megközelítés lehetővé teszi számukra, hogy ugyanazt a kódot használják a front- és a backendeken. Többplatformos mobilfejlesztés. A fejlesztők a Kotlin for Androidot és más mobilplatformokat használnak, beleértve az Apple iOS-t, az Apple watchOS-t és a Linuxot. Kotlin vs Java Kotlin és Java mindkettő nagyon népszerű programozási nyelv az Android alkalmazásfejlesztésben, és mindenkorral növekvően elterjedt. A nyelvnek előnyei között a hagyományos Java-val szemben állnak. A szintaxis tiszta és intuitív (könnyen használható és érthető). Közöttük, hogy nagyon hasonlít a Scalára, de egyszerűbb. Automatikus konverzió A JetBrains egy új funkciót integrált az IntelliJ-be, amely konvertál Kotlinra, és jelentős időt takarít meg. És azt is tartja Summing Things Upus, hogy újraírja a hétköznapi kódot. A null-biztonsága nagyszerű Ezáltal megszabadulhatunk a NullPointerExceptionktól. Ez a fajta rendszer segít nekünk elkerülni a null pointer kivételeket. Ebben a rendszer megtagadja a nullát hozzárendelni vagy visszaadni próbáló kód lefordítását.

Elsődlegesen az olvasható szintaxisra koncentrál, így a kód átnézése A kód átnézések nem jelentenek problémát, azokat továbbra is elvégezhetik azok a csapattagok, akik nem ismerik a nyelvet. Kevesebb kódot igényel Minden benne írt kóddarab sokkal kisebb, mint a Java-ban írtak, mivel kevésbé szószátyár. Tehát "kevesebb kód egyenlő kevesebb hibával." és kevesebb időt fordít az ütemezésre és a projektköltségek megtakarítására. Funkcionális programozás A legfontosabb dolog, ami bennük van, hogy ez egy funkcionális programozási nyelv. Sok hasznos módszerből áll, többek között magasabb rendű függvények, lambda kifejezések, operátor túlterhelés, lista kiértékelés, operátor túlterhelés és még sok más.

2.7. ARCore

Az ARCore a Google platformja a kiterjesztett valóság élményeinek létrehozására. Az ARCore különböző API-k segítségével lehetővé teszi, hogy a telefon érzékelje a környezetét, megértesse a világot és interakcióba lépjen az információkkal. Az API-k egy része And-

roid és iOS rendszereken keresztül is elérhető, hogy lehetővé tegye a közös AR-élményeket. Az ARCore három kulcsfontosságú képességet használ a virtuális tartalom és a telefon kameráján keresztül látott valós világ integrálásához: A mozgáskövetés lehetővé teszi, hogy a telefon megértse és nyomon kövesse a világhoz viszonyított helyzetét. A környezet megértése lehetővé teszi a telefon számára, hogy érzékelje minden típusú felület méretét és helyzetét: vízszintes, függőleges és ferde felületek, például a talaj, a dohányzóasztal vagy a falak méretét. A fénybecslés lehetővé teszi a telefon számára, hogy megbecsülje a környezet aktuális fényviszonyait. Támogatott eszközök Az ARCore-t úgy terveztek, hogy az Android 7.0 (Nougat) vagy újabb Android 7.0-t futtató, minősített Android telefonok széles skáláján működjön. Hogyan működik az ARCore? Az ARCore alapvetően két dologt csinál: követi a mobileszköz helyzetét mozgás közben, és felépíti saját megértését a valós vilagról. Az ARCore mozgáskövető technológiája a telefon kameráját használja az érdekes pontok, az úgynevezett jellemzők azonosítására, és követi, hogy ezek a pontok hogyan mozognak az idő múlásával. E pontok mozgásának és a telefon inerciális érzékelőinek leolvasásainak kombinációjával az ARCore meghatározza a telefon helyzetét és tájolását is, ahogy az a térben mozog. A kulcspontok azonosítása mellett az ARCore képes felismerni a sík felületeket, például az asztalt vagy a padlót, és képes megbecsülni a körülötte lévő terület átlagos megvilágítását is. Ezek a képességek együttesen lehetővé teszik, hogy az ARCore felépítse saját megértését a körülötte lévő világról. Az ARCore valós vilagról alkotott megértése lehetővé teszi, hogy objektumokat, megjegyzésekkel vagy más információkat úgy helyezzen el, hogy azok zökkenőmentesen illeszkedjenek a valós világba. Elhelyezhet egy szunyókáló cicát a dohányzóasztal sarkán, vagy megjegyzést fűzhet egy festményhez a művész életrajzi információival. A mozgáskövetés azt jelenti, hogy bármilyen szögből mozoghatsz és nézheted ezeket a tárgyakat, és még ha meg is fordulsz és elhagyod a szobát, amikor visszatérsz, a cica vagy a megjegyzés ott lesz, ahol hagyad. Az ARCore számos SDK-t kínál a legnépszerűbb fejlesztői környezetekhez. Ezek az SDK-k natív API-kat biztosítanak az összes alapvető AR-funkcióhoz, például a mozgáskövetéshez, a környezet megértéséhez és a fénybecsléshez. Ezekkel a képességekkel teljesen új AR-élményeket hozhat létre, vagy meglévő alkalmazásokat bővíthet AR-funkciókkal.

A GLB (Graphics Library Binary) egy kompakt, bináris formátum 3D modellek és jelenetek tárolására. Az ARCore egy olyan Google által fejlesztett keretrendszer, amely lehetővé teszi az Android készülékek számára az augmentált valóság (AR) élmények megjelenítését és interakcióját. Az ARCore támogatja a GLB formátumot, amely lehetővé teszi az ARCore alkalmazások számára, hogy 3D modelleket és jeleneteket jelenítsenek meg az augmentált valóságban. A GLB formátum előnye, hogy a modell geometriáját, anyagát és animációt egyetlen fájlba csomagolja, így könnyen kezelhető és átvihető. Az ARCore segítségével a GLB fájlokat az AR alkalmazásokban az alábbiak szerint használhatjuk: Modell importálása: Az ARCore lehetővé teszi a GLB fájlok importálását az alkalmazásba. Ezek a modellek lehetnek például karakterek, tárgyak vagy épületek, amelyeket az AR környezetben szeretnénk megjeleníteni. Az importált modellt a GLB fájlból tárolt geometria, anyagok és textúrák alapján jeleníti meg az alkalmazás. Helymeghatározás: Az ARCore segítségével a GLB modellt lehet a valós környezethez rögzíteni, vagyis pozícionálni és orientálni az AR térben. Ez lehetővé teszi a modelltnek, hogy kövesse a környezeti viszonyokat, például az asztalon, a padlón vagy más felületen való elhelyezést. Interakció és animáció: A GLB modell lehetőséget ad interaktív elemek beillesztésére az AR alkalmazásokba. Ez lehetővé teszi például a felhasználók számára, hogy közelről



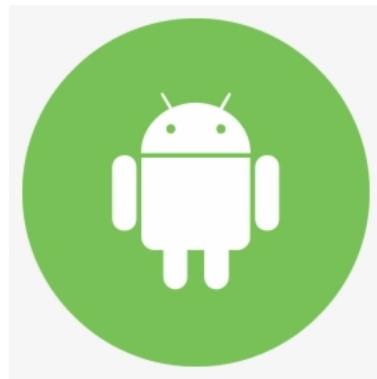
2.7. ábra. AR Core

megtekintsék és forgassák a modellt, vagy akár az AR környezetben animációkat jelenítsenek meg a modell alapján. Az ARCore és a GLB kombinációja izgalmas és kreatív lehetőségeket nyújt az augmentált valóság alkalmazások fejlesztéséhez. A GLB formátum kompaktsága és a modell teljes tartalmának tárolása egyetlen fájlban lehetővé teszi az egyszerűbb és hatékonyabb tartalomkezelést az AR alkalmazásokban.

A GLTF (Graphics Library Transmission Format) egy másik, szintén bináris formátum 3D modellek és jelenetek tárolására. A GLTF az OpenGL és az OpenGL ES API-k támogatását célozza meg, és célja az, hogy könnyen kezelhető és hatékony legyen a 3D tartalom átvitele és megjelenítése között különböző alkalmazások és platformok között. A GLTF egy nyílt szabvány, amelyet az Khronos Group fejlesztett ki, és széles körben támogatott a 3D grafikus szoftverekben és keretrendszerekben. Az ARCore is támogatja a GLTF formátumot, és lehetővé teszi az AR alkalmazások számára, hogy 3D modelleket és jeleneteket jelenítsenek meg az augmentált valóságban. Az ARCore segítségével a GLTF fájlokat az AR alkalmazásokban az alábbiak szerint használhatjuk: Modell importálása: Az ARCore lehetővé teszi a GLTF fájlok importálását az alkalmazásba. Ezek a modellek tartalmazhatnak geometriát, anyagokat, textúrákat és animációkat, amelyeket az AR környezetben szeretnénk megjeleníteni. Az importált modellt a GLTF fájlban tárolt adatok alapján jeleníti meg az alkalmazás. Helymeghatározás: Az ARCore segítségével a GLTF modellt lehet a valós környezethez rögzíteni, vagyis pozícionálni és orientálni az AR térben. Ez lehetővé teszi a modellnek, hogy kövesse a környezeti viszonyokat, például az asztalon, a padlón vagy más felületen való elhelyezést. Interakció és animáció: A GLTF modell lehetőséget ad interaktív elemek beillesztésére az AR alkalmazásokba. Ez lehetővé teszi például a felhasználók számára, hogy közelről megtekintsék és forgassák a modellt, vagy akár az AR környezetben animációkat jelenítsenek meg a modell alapján. A GLTF formátum kompakt és hatékony, mivel csak a szükséges adatokat tartalmazza a 3D tartalom leírásához. Ez lehetővé teszi az egyszerűbb és gyorsabb adatátvitelt és feldolgozást az AR alkalmazásokban.

2.8. Android

Az Android szoftverfejlesztés az a folyamat, amelynek során az Android operációs rendszert futtató eszközökre alkalmazásokat hoznak létre. Az Android-alkalmazások Kotlin, Java és C++ nyelveken írhatók" az Android szoftverfejlesztő készlet (SDK) segítségével, de más nyelvek használata is lehetséges. minden nem Java virtuális gépen (JVM) futó nyelv, például Go, JavaScript, C, C++ vagy assembly, JVM nyelvi kód segítségét igényli, amelyet eszközök szolgáltathatnak, valószínűleg korlátozott API-támogatással. Egyes programozási nyelvek és eszközök lehetővé teszik a keresztpalatformos alkalmazástámogatást (azaz Android és iOS számára is). A harmadik féltől származó eszközök, fejlesztőkörnyezetek és a nyelvi támogatás is folyamatosan fejlődött és bővült a kezdeti SDK 2008-as megjelenése óta. Az Android-alkalmazások hivatalos terjesztési mechanizmusa a végfelhasználók számára a Google Play; ez lehetővé teszi az alkalmazások fokozatos, fokozatos kiadását, valamint a tesztelők számára az alkalmazás előzetes verzióinak terjesztését is.



2.8. ábra. Android

3. fejezet

Adatbázis

3.1. MSSQL

A Microsoft SQL Server egy saját tulajdonú relációs adatbázis-kezelő rendszer, amit a Microsoft fejlesztett ki. Mint adatbázis szerver, ez egy szoftvertermék, aminek fő funkciója az adatok tárolása és lekérdezése más szoftveralkalmazások kérése alapján - melyek lehetnek ugyanazon a számítógépen vagy egy másik számítógépen hálózaton keresztül (beleértve az internetet is). A Microsoft legalább egy tucat különböző kiadást kínál a Microsoft SQL Serverból, amelyek különböző célcsoportokra és terhelésre tervezettek, kis egyszámítógépes alkalmazásuktól kezdve a nagy internetes alkalmazásokig, amelyek sok egyidejű felhasználót szolgálnak ki.



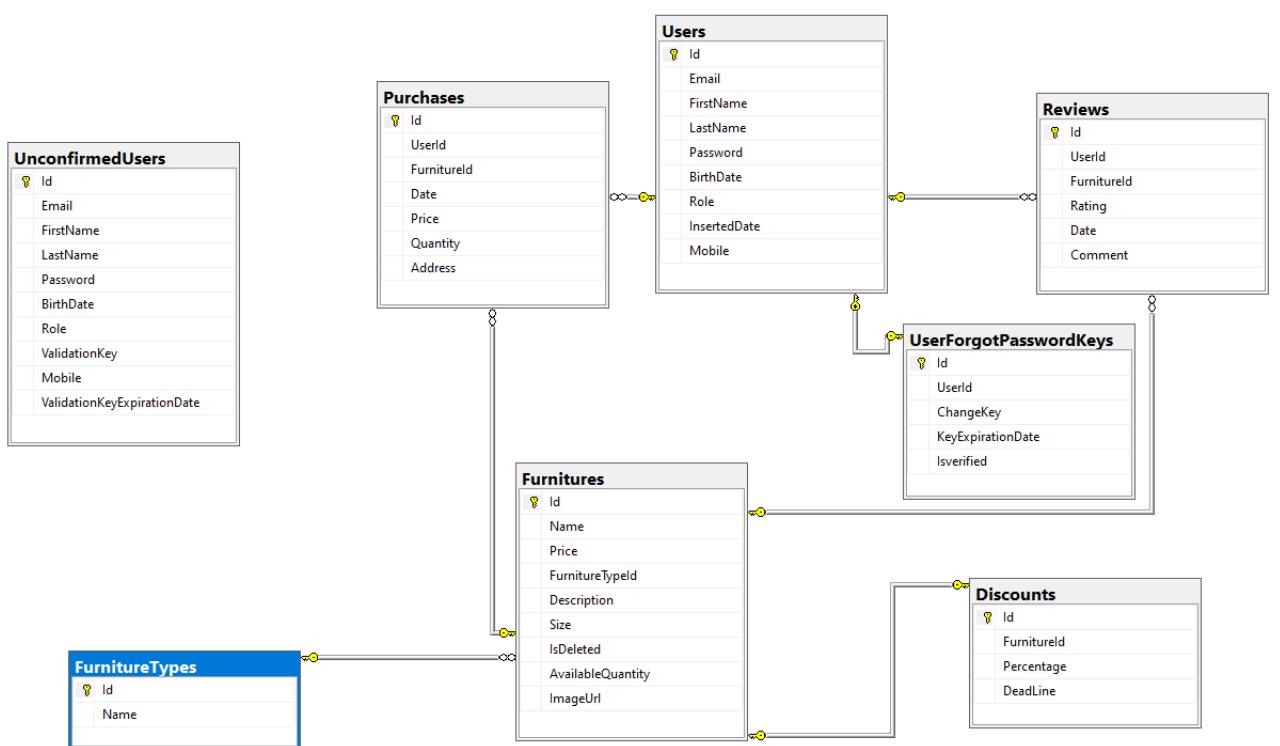
3.1. ábra. Microsoft SQL Server

3.2. Az adatbázis strukúrája

Az adatbázis tábláiban tárolt adatok lekérdezéséhez illetve módosításához az SQL Server Management Studio-t használtam, amely megfeleő kezelőfelületet biztosított a fent említett tevékenységek elvégzéséhez. Az adatbázis táblái a következő logika mentén épültek fel:

- UnconfirmedUsers tábla, amely azokat az adatokat tárolja, amelyeket a regisztráció során ad meg az új felhasználó.
- A Users táblába kerülnek az előbb említett tábla adatai, szigorúan csak akkor, ha a regisztráció során megadott email cím ellenőrizve volt az emailben kapott verifikációs linkre való kattintással.
- Ha az adott felhasználó elfelejtené a jelszavát, akkor lehetősége van a UserForgotPasswordKeys segítségével visszaállítani a jelszót.
- A FurnitureTypes tábla tárolja az adott bútorfajtákat.
- A Furnitures pedig magát a bútorokat, amelyek elérhetőek az áruház kínálatában.
- Az adott bútorokhoz az admin hozzáadhat leárazásokat, amelyek a Discounts táblában találhatóak.
- A felhasználó megrendelhet egy adott terméket és ennek a vásárlásnak az adatait a Purchases táblában tároljuk.
- Végül pedig a felhasználó értékelheti a megvásárolt termék minőségét illetve megjegyzést is fűzhet hozzá, ezt tartalmazza a Reviews tábla.

A fent leírt struktúra az alábbi diagramon tekinthető meg, amely bemutatja a táblákat, azok beszédes névvel ellátott mezőit is, illetve a kulcsokat, amelyekkel kapcsolódnak egymáshoz.



3.2. ábra. Adatbázis diagramja

4. fejezet

Funkcionális követelmények

Ebben a fejezetben bemutatom a rendeszer funkcionális követelményeit a Use Case Diagram segítségével (lásd 4.1. ábra). Az alábbi diagram bemutatja a rendszer funkcionálitásait, felhasználói interakciókat és a felhasználók által végrehajtott feladatokat. Szemléltetni fogom a szereplők, úgynévezett "aktorok" és a use cas-ek közötti kapcsolatokat.

4.1. Use Case Diagram

4.1.1. Aktorok - Szereplők

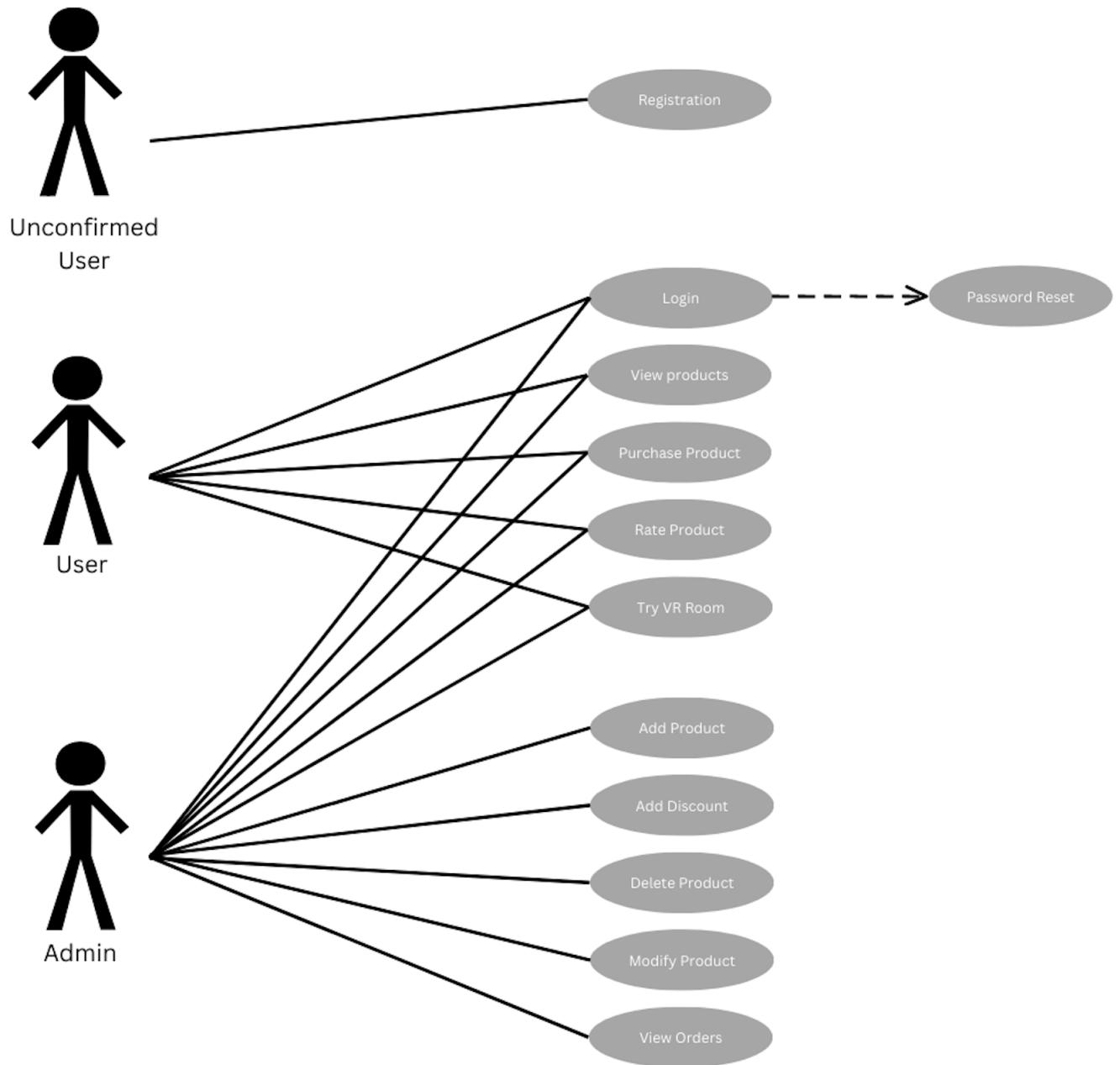
Az alábbi diagram három szereplőt jelenít meg, mégpedig azt a három fajta felhasználót, aki kapcsolatba léphet a rendszerrel: még nem regisztrált felhasználó, felhasználó illetve az admin.

- A még nem regisztrált felhasználó (Unconfirmed User) egyetlen feladatot tud végrehajtani, mégpedig a regisztrációt. Tekintve, hogy az alkalmazás használatához elengedhetetlen az email címmel és jelszóval való belépés, ezért az a felhasználó, aki először fogja használni az alkalmazást köteles egy regisztráció alkalmával létrehozni a saját fiókját. Adatainak megadása után egy emailben kapott linkre kattintva tudja megerősíteni az adatainak helyességét. Ez által automatikusan bekerül a userek listájába és így már élhet a felhasználó funkcionálitásaival.
- A felhasználó(User), aki már rendelkezik érvényes belépési adatokkal, különböző használai esetekkel rendelkezik, amelyek tulajdonképpen egy egyszerű vásárló rendelkezésére álló feladatokat foglalják magukba.
- Az admin az a típusú felhasználó, aki ugyanazokat a feladatokat képes elvégezni, mint az egyszerű felhasználó, illetve ezen kívül kezeli a rendszer adminisztrációs részét is.

4.1.2. Use casek - Használati esetek

A use case-ek olyan leírások vagy modellek, amelyek bemutatják, hogy hogyan használják a rendszert a felhasználók konkrét feladatok vagy célkitűzések elérésére. Az alábbiakban bemutatom, a diagram megértése érdekében, hogy saját projektem esetében melyek lennének a használati esetek.

- Registration: Amikor egy új felhasználó szeretné használni az alkalmazást, regisztrálnia kell, annak érdekében, hogy a felhasználói adatai érvényesítésre kerüljenek és elkészüljön a fiókja, amelyet a későbbiekben a belépés során használhat.
- Login: Az email cím illetve jelszó megadásával tudunk belépni az áruházba.
- Password Reset: Elfelejtett jelszó esetén kérhetjük a jelszó visszaállítását, emailben kapott ellenőrző kód segítségével.
- View Products: A felhasználók megtekinthetik a bútoráruház termékeinek listáját, egyenként rákattintva pedig részletesebb leírást is kaphatnak az adott bútordarabról.
- Purchase Product: A felhasználó megvásárolhatja a terméket, beszúrva a vásárlások listájába.
- Rate Product: Vásárlás után a felhasználó értékelheti a terméket egy 5-ös skálán illetve opcionálisan mejegyzésekkel is fűzhet az értékeléshez.
- Try VR Room: Az adott bútordarab elhelyezhető egy viruális szobában, amely jobb képet mutat a felhasználónak a bútor valódi méretéről, anyagának textúrájáról illetve az elforgatás segítségével több szövőgből is megtekintheti.
- Add Product: Az Admin típusú felhasználó hozzáadhat új bútordarabot a terméklistához, amelyet a felhasználók megvásárolhatnak.
- Add Discount: Az Admin létrehozhat leárazásokat adott termékekre, így az áruk automatikusan csökken a megadott százalékkal.
- Delete Product: Az Admin törölhet egy adott terméket az áruházból, így ez már semmilyen formában nem lesz elérhető
- Modify Product: Az Admin módosíthatja az adott terméket, pontosabban annak az adatait például leírását, méreteit, stb.
- View Orders: Az Admin hozzáfér az összes megrendelés listájához, annak érdekében, hogy ki tudja szállítani azokat.



4.1. ábra. Use Case Diagram

5. fejezet

Felhasználói követelmények

5.0.1. Cél és áttekintés

Az alkalmazás célja, hogy egy olyan bútoráruházat valósítson meg, amely úgy a forgalmazó, mint a felhasználó, vásárló számára megadja azokat a szolgáltatásokat, amelyekre azoknak szükségük van.

Az alkalmazás hasznossága abban áll, hogy míg a forgalmazó számára biztosít egy egyszerűen kezelhető admin felületet, ahonnan nyomon követheti vállalkozásának forgalmát illetve elérheti a számára hasznos funkcionálitásokat, ugyanakkor a vásárlók számára is egy kényelmes, könnyen kezelhető felületet níjt a bútorok böngészéséhez illetve ezek megvásárlásához, illetve további funkcionálitások segítségével javítja a felhasználói élményt és segít a vevőnek a beruházás döntésének meghozatalában.

5.0.2. Felhasználói csoportok

Az alkalmazás felhasználót három csoportba tudnánk kategorizálni, funkciójuk illetve eddigi tevékenységeik alapján. Az admin felhasználó, aki az áruház tulajdonosa, a bútorok forgalmazója lenne, olyan funkcionálitásokra van lehetősége, amelyekhez más, átlagos felhasználóknak nincs hozzáférésük. Az ő szerepe teljesen eltér minden más felhasználó szerepéktől, hiszen ő arra használja az alkalmazást, hogy az áruházat naprakészen tartsa, az új termékeket feltölthesse, régieket frissíthesse, illetve a rendelések nyomon követésére is lehetősége van.

Ezzel ellentétben a felhasználó az alkalmazást csupán böngészésre, keresgélésre használja, megtekintheti az elérhető bútorokat illetve megvásárolhatja őket.

A harmadik típusú felhasználó, akit meg kell különböztetnünk a fent említettektől, az a még nem regisztrált felhasználót jelentené, akinek regisztrálnia kell a belépés előtt, hiszen az ő adatai még nincsenek eltárolva az adatbázisunkban, így rögtön első lépésként elkerüljük adatait. Miután ezek ellenőrzésen mentek keresztül és eltároltuk őket, azután ez a felhasználó is sima felhasználóvá válik és elkezdheti a böngészést az áruházban.

5.0.3. Felhasználói interfész

A bútoráruház esetében a felhasználói interfést a mobilalkalmazás biztosítja, amelynek megvalósításában célom, hogy könnyen átlátható legyen, illetve a kezelése ne okozzon

nehézséget a sok különböző életkorú, digitális felkészültségű, különböző igényekkel rendelkező vásárlónak, akik megfordulhatnak az áruházban.

- Az alkalmazás megnyitása egyenesen a login képernyőre dob, amely lehetőséget nyújt a vevőnek a saját felhasználói adataival belépni az áruházba, vagy hogyha nem rendelkezik elmentett adatokkal, akkor a regisztrációhoz navigálva ezt a hiányosságot könnyen pótolhatja. Elfelejtett jelszó esetén a fiók könnyen megmenthető egy jelszóemlékeztető kérésével, amelyet a felhasználó emailben fog megkanni a rendszertől, hogy új jelszót állíthasson be.
- Amennyiben a belépés megtörtént, a felhasználó végignézheti a termékek listáját a fő oldalon, rákereshet egy termékre név szerint, vagy kategória szerint illetve rendezheti a termékek listáját különböző sorrendekbe, a saját igényei szerint.
- Egy termékre kattintva a listában, megtekinthetjük az adott árucikk adatait részletesbben, elolvashatjuk a leírását, illtetve megnézhetjük, hogy milyen visszajelzéseket küldtek azok a felhasználók, akik korábban már megvásárolták a terméket, majd értlkelték, illetve hozzájárultak a termékhez. A termék adatainak képernyőjén érhető el az a funkcionálitás is, amely által a vásárló átléphet egy virtuális szobába, ahol megtekintheti a termék 3D-s modelljét, amelyet elforgathat, kedve szerint letehet belőle a szobába több darabot is, ez által több szögből megtekintheti a terméket, ez által jobb összképe lesz az adott bútor darabról, így könnyebben eldöntheti, hogy valóban meg akarja-e vásárolni.
- Ha a felhasználó úgy dönt, hogy megvásárolja a terméket, akkor a vásárlás gombra kattintva a termék bekerül a kosárba. A kosár tartalmát a képernyő alján látható menüből érhetjük el, amely tartalmazza és kilistázza az összes olyan elemet, emelyet betettünk a kosarunkba illetve a végösszeg mellett látható checkout gombbal befejezhetjük a vásárlást.
- A következő kötelező lépés a vásárlás véglegesítéséhez a kiszállítási cím megadása, ami után már csak egy gombnyomás választ el a vásárlástól.
- A felhasználó az utolsó menüpontban megtekintheti a saját profilját az adataival illetve lehetősége van kijelentkezni az alkalmazásból.

Az alkalmazás funkciói teljesen átalakulnak, amikor a felhasználó az admin szerepkörrel rendelkezik. Az ő fő tevékenysége az áruház naprakészen tartása, az eladások követése illetve a

- Az admin számára a főoldalon a termékek mellett látható egy hozzáadás gomb is, amelyre rágatva hozzáadhat egy új bútor darabot az áruházhöz, megadva annak tulajdonságait, illetve feltöltve hozzá egy képet, a telefon galériájából. Ha sikeresen megtörtént a feltöltés, akkor a bútor darab bekerül a főoldalon megjelenített termékek közé.
- Egy termékre kattintva az adminnak lehetősége van az adott tárgy adatainak módosítására, illetve leárazás hozzáadására is.

- Az alsó navigációs menüben a kosárra kattintva az adminnak megjelennek a más felhasználók által feladott rendelések minden fontos adattal együtt. A rendelések között lehet kereni a felhasználónév, vagy email cím alapján illetve rendezni is lehet a rendeléseket különböző szempontok alapján.
- A profilra kattintva az admin megtekintheti az összes felhasználót azok adataival illetve ő is ki tud jelentkezni, akárcsak egy egyszerű felhasználó.

6. fejezet

Nem funkcionális követelmények

Az alkalmazás futtatásához szükséges a 9-es, vagy annál nagyobb Adnroid verzió illetve a telefonra szükséges telepíteni a Googel Playből a Google Play Services for AR nevű alkalmazást a virtuális valóság működédéhez. A zökkenőmentes működés szükséges feltétele a megfelelő internet kapcsolat

7. fejezet

Szerver

7.1. .NET Core WebAPI

Az API feljesztésére a .NET Core WebAPI keretrendszer használtam, amely lehetővé tette az API különböző platformokon való futtatását. A .Net Core WebAPI segítségével könnyen interakcióba léphettem az APIval, elősegítette az adatok hatékony kezelését. Itt definiáltam a különböző végpontokat, a HTTP metódusokat, amelyekről a későbbiekben lesz szó. Az említett API végpontok elérhetőek a mobilalkalmazásból, így az app használhatja ezeket a adatlekérdezésekre illetve a felhasználói interakcióra.

7.2. API végpontok

Az API végpontok azok a HTTP kérések, amelyek a kliensek közötti kommunikációt teszik lehetővé. A kliensek meghatározott kéréseket küldhetnek az API-hoz illetve ezekre a végpontok által meghatározott válaszokat kapják.

A végpontok attribútumait és konfigurációját a kontrollerekben definiálhatjuk, a controller metódusok fölött, illetve megadhatjuk a kérések típusát(GET, POST, PUT, DELETE), útvonalakat([Route("api/users")]) illetve egyéb paramétereit.

Az alábbiakban láthatunk egy példát, amely bemutatja a projekttemben levő UsersController tartalmának egyik részletét, ahol egy olyan Http metódust hozunk létre, amely visszaadja a felhasználónak a saját adatait, illetve megengedi azt, hogy az Admin is hozzáférhessen ezekhez. A példa jól szemlélteti a fent említett paraméterek megadásának módját az alábbi három paraméterre: Azon szerepkörök megadása, akiknek jogosultságot adunk a metódus használatára: [Authorize(Roles = "User,Admin)]

A metódus típusának megadása: [HttpGet]

Az útvonal megadása: [Route("GetMyUser")]

Továbbá a kép szemlélteti, hogy hogyan érhetjük el a Fejlécben megadott tokent, amelyből kiderül majd, hogy az adott felhasználónak van-e jogosultsága használni az adott metódust, illetve szemlélteti, hogy a UserService által visszaadott választ vagy hibaüzeneteket hogyan kezeli.

A továbbiakban bemutatom a projektemben létrehozott Http végpontokat, illetve ezek funkciójait.

Négy fajta HTTP kérést implementáltam, amelyek a következő élokat szolgálták: 1. GET kérés: Az adatok szerverről való lekérésére használjuk. 2. POST kérés: Az új erőfor-

```

[Authorize(Roles = "User,Admin")]
[HttpGet]
[Route("GetMyUser")]
0 references
public async Task<IActionResult> GetMyUser()
{
    try
    {
        StringValues values;
        var res = Request.Headers.TryGetValue("Token", out values);
        var user = await _userService.GetUserFromToken(values);
        if (user == null)
            return NotFound(new BackendResponse<string>("User with given email not found"));
        return Ok(new BackendResponse<UserModel>(_mapper.Map<UserModel>(user)));
    }

    catch(Exception ex)
    {
        return StatusCode(500, new BackendResponse<string>(ex.Message));
    }
}

```

7.1. ábra. HttpGet példa szemléltetése

rások létrehozására használjuk. 3. PUT kérés: Az erőforrások frissítésére használjuk. 4. DELETE kérés: Az erőforrások törlésére használjuk. A végpontokat három kategóriába soroltam, az alapján, hogy melyik entity-vel kapcsolatban látja el feladatát:

A bútorokkal kapcsolatos kérések:

GET/api/Furnitures : A bútorok listáját kérdezi le.

POST/api/Furnitures : Új bútor hozzáadására ad lehetőséget.

PUT/api/Furnitures : A listában már meglévő bútorok szerkesztése.

GET/api/Furnitures/:id : A bútorok listájából megkereshetünk egy adott id-val rendelkező bútor.

GET/FurnitureTypes: Lekéri a bútorok kategóriáit tartalmazó listát./

Reviews- A visszajelzésekkel és hozzászólásokkal kapcsolatos kérések:

POST/api/Reviews : Új hozzászólás írására használjuk.

GET/api/Reviews/FurnitureReviews/:furnitureid : Megkapjuk egy adott id-vel rendelkező bútorhoz fűzött hozzászólásokat.

DEL/api/Reviews/:id :Adott id-val rendelkező hozzászólás törlését teszi lehetővé.

Users mappában találhatóak a felhasználókkal kapcsolatos kérések:

POST/api/Users/sendRegistrationCode :Email-ben történő regisztrációs kód küldésére használjuk.

GET/api/Users/Registration : A regisztráció kivitelezésére ad lehetőséget az új felhasználónak.

POST/api/Users/Login : Belépés.

POST/api/Users/SendForgotPasswordKey : Elfelejtett jelszó esetén a felhasználó email-ben kap egy biztonsági kódöt amellyel visszaállíthatja majd a jelszavát.

POST/api/Users/VerifyForgotPasswordKey : A fent említett kód megadása és helyességének ellenőrzése.

PUT/api/Users/ChangePassword : Új jelszó megadása.

GET/api/Users/GetMyUser : A felhasználó saját adatainak lekérése (Annak a felhasználónak az adatai amellyel történt a belépés).

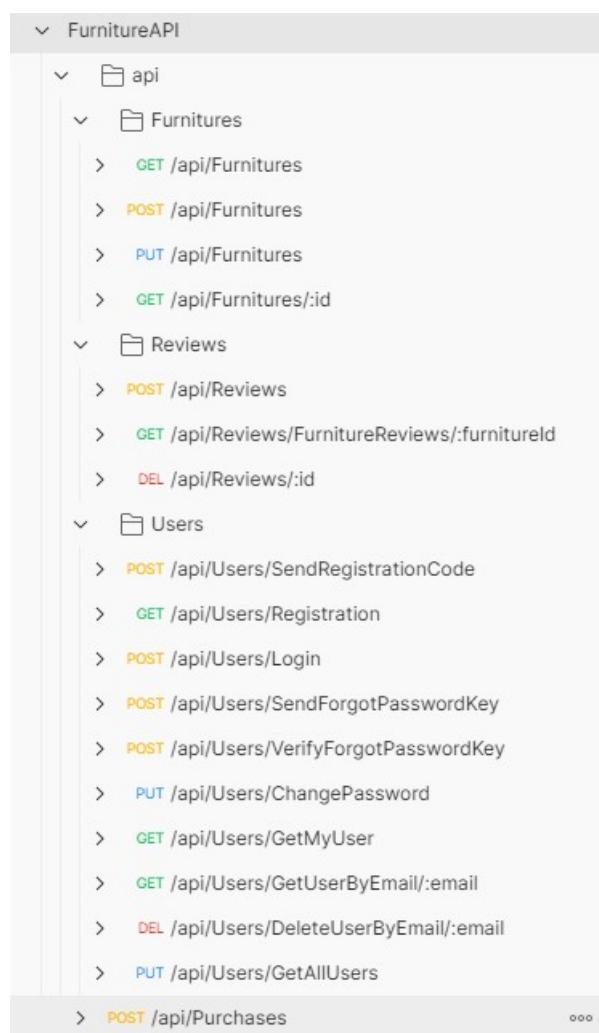
GET/api/Users GetUserByEmail/:email : Email alapján keresendő felhasználó adatainak lekérése.

DEL/api/Users/DeleteUserByEmail/:email : Adott email címmel rendelkező felhasználó törlése

GET/api/Users GetAllUsers : Az összes felhasználó adatainak lekérése a táblából

Továbbá látható még a vásárlásokkal, rendelésekkel kapcsolatos kérés:

GET/api/Purchases : A vásárlás adatainak hozzáadása a táblához.



7.2. ábra. API végpontok

Fontos figyelembe vennünk azt, hogy ezeknek a végpontoknak a használata korlátozva van, annak függvényében, hogy a felhasználónak mi a szerepköre. A következő két típusú felhasználót különböztetünk meg: "Admin" és "User". Az endpointok definiálá-

sánál figyeltem arra, hogy mindenki csak a saját feladatkörének megfelelő végpontokat tudja használni, amelyet a [Authorize(Roles = "User")], vagy [Authorize(Roles = "Admin")] segítségével tudtam megvalósítani, illetve ha olyas valaki akarta volna használni az endpointot, akinek nem volt erre jogosultsága, akkor ő "403, Forbidden" hibaüzenetet kapta.

Az endpointok különböző Status codeokat térítenek vissza, annak érdekében, hogy tudtunkra adják, hogy milyen eredménnyel zárult a kérés. A leggyakrabban használt status codeok a következők lennének: 200 OK: A kérés sikeres volt, és a szerver visszatért a kért adatokkal.

201 Created: A kérés sikeres volt, és új erőforrás lett létrehozva.

204 No Content: A szerver sikeresen feldolgozta a kérést, de nem kell tartalmat visszaadnia.

400 Bad Request: A szerver nem tudta értelmezni a kérést a hibás szintaxis vagy érvénytelen paraméterek miatt.

401 Unauthorized: A kéréshez hitelesítés szükséges. A kliensnek érvényes hitelesítő adatokat kell megadnia a kért erőforrás eléréséhez.

403 Forbidden: A szerver megértette a kérést, de megtagadja az engedélyezést. A kliensnek nincs jogosultsága a kért erőforráshoz.

404 Not Found: A szerver nem találta meg a kért erőforrást.

500 Internal Server Error: Általános hibaüzenet, amely arra utal, hogy váratlan állapot lépett fel a szerveren.



7.3. ábra. Példa a kérésre kapott status codera

Ezeket az általánosan használt eredmény jelzéseket használtam én is a megvalósítás során illetve a postman felületén láthattam az adott kérésre érkező eredményeket.

Az endpointok a kérésre adott válaszként a status codeok mellett visszatéríthatnak az adott kérésnek megfelelő json-t, amely a kért mezőkből és azok tartalmából áll, ezeket az elempárokat vesszőkkal elválasztva, jól értelmezhető szerkezetként adja vissza.

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON ↻

```
1   "result": [
2     {
3       "id": 1,
4       "email": "matyus.erzsebet@student.ms.sapientia.ro",
5       "firstName": "Betty2",
6       "lastName": "Matyus2",
7       "birthDate": "1950-09-07T01:42:24.716",
8       "role": "Admin"
9     },
10    {
11      "id": 2,
12      "email": "matyusb.erzsebet@gmail.com",
13      "firstName": "Betty",
14      "lastName": "Matyus",
15      "birthDate": "2006-06-19T00:00:00",
16      "role": "User"
17    }
18  ]
19
20
```

7.4. ábra. Példa JSON válaszra

8. fejezet

Szerver arhitekúra

A .NET szerver legfelsőbb elemei, amivel a felhasználó direkt módon kommunikálhat azok a controllerek. 4 fő controllert használtunk, amelyek a felhasználók, vásárlások, értékelések, illetve a bútorokhoz kapcsolódó adatlekérések, illetve módosításokért felelősek. minden controller tartalmazza a hozzá társuló service-t. Ezek a servicek felelősek az üzleti logikáért, továbbítják a controllereknek az adott logika végrehajtásának végeredményét, majd a controllerek ezek alapján ad választ a felhasználónak. A servicek direkt módon használják az adatbázisunkat, code-first megközelítés segítségével könnyedén megvalósítható, hogy az adatbázisomat egy DbContext osztály reprezentálja. A DbContext minden DbSet<T> tagja az adatbázisunk egy táblájára mutat, ennek pedig minden eleme az adatbázisom egy sorára. A szerverem rendelkezik HostedServicekkal, melyek bizonyos időközönként futtatnak feladatokat, legtöbbször adatok törléséhez, például már érvénytelen leárazások esetében.

8.1. Konkrét megvalósítások

8.1.1. Táblák létrehozása

Az adatbázisunk code-first megközelítéssel lett létrehozva. Ezt a Microsoft.EntityFrameworkCore libraryjának a Migrations osztályával értem el. A táblákra mutató osztályok annotációk, illetve a DbContext létrehozásánál beállított kritériumokból állítja elő azt a kódsort, amely ezeknek megfelelően hozza létre, vagy akár módosítja az adatbázisunkat.

Parancsok:

Migration hozzáadása: Add-Migration InitialCreate – létrehozza az InitialCreate nevű migrationt

Update-Database: frissíti az adatbázisunk felépítését egy adott migration függvényben.

Azt, hogy a web apim melyik adatbázishoz kapcsolódik az appsettings.json fájlbanban adjuk meg connection stringként, amely tartalmazza a szerver nevét, az adatbázis nevét, ha szükségés bejelentkezést akkor az adatbázis kezelőjének felhasználónevét és jelszavát.

```
    "AllowAnonymous": true,
    "connectionStrings": {
        "DefaultConnection": "Server=localhost;Database=FurnituresDB;Trusted_Connection=True;MultipleActiveResultSets=true"
    }
},
```

8.1. ábra. Connection String

A táblák azonosítói egyediek és a rendszer által vannak generálva, így nem kell figyelni arra hogy ne legyen többször előforduló azonosító. A primary illetve foreign keyek közötti kapcsolatot meg lehet adni az adott táblához kapcsolódó osztályban annotációk segítségével, illetve a DbContext-ünk OnModelCreating felülírható metódusában. A több a többhöz kapcsolatokat leginkább itt konfiguráltam be, valamint egyes mezőket egyediségét is itt kérélmeztem.

```
[Required]
[ForeignKey(nameof(Furniture))]
7 references
public int FurnitureId { get; set; }
2 references
public FurnitureEntity Furniture { get; set; }
```

8.2. ábra. Idegen kulcs kapcsolat

```
[Required]
[ForeignKey(nameof(Furniture))]
7 references
public int FurnitureId { get; set; }
2 references
public FurnitureEntity Furniture { get; set; }
```

8.3. ábra. Adatbázis létrehozásának szabályozása

8.1.2. Servicek

A servicek felelnek az üzleti logikáért. minden service implementálja a hozzá rendelt interfacet dependency injection, így absztraktizáljuk a webszerverünket teljesítve bizonyos osztályszintű tervezési elveket. Ez az implementálás három féle lehet, minden egyes kérés esetén letrejöhet egy új service (transient), az egész alkalmazáson belül egyszer jön létre az adott service (singleton) vagy scoped, amely scopeolt függőséget hoz létre. A GET controller-függvényekben meghívódó service metódusok legtöbbször adatokat küldenek a controllereknek, míg a POST vagy PUT controllerek servicének függvényei enumokat, így megoldható, hogy több esetet is lekezeljék a controllerek szintjén.

8.1.3. Controllerek

A servicek által visszatérített adatokat AutoMapper könyvtár használatával alakítottam a felhasználó számára értelmezhető üzenetté, valamint ennek segítségével értem el, hogy egyes információk rejtve maradjanak felhasználói szinten. A controllerek POST

és PUT metódusokhoz igényelnek egy kérés bodyt. Ahhoz hogy bizonyos hibásan megadott adatok a servicehez eljutás előtt lekezelődjenek ugyancsak annotációk segítségével valósítottam meg. Ilyen hibák esetén allíthatunk a felhasználó által érhető üzeneteket. Bizonyos controllerek igényelnek szerepkörhöz kötött jogosultságot. Az autorizáláshoz használt információkat a kérés fejlécében adtuk meg "Token" név alatt, hozzá társítva a Json Web Tokent. A szerepkört a tokenhez Claim formájában adtam hozzá. A token bejelentkezéskor érkezik meg válaszként, majd elmentve tudjuk használni adott kérések futtatásához.

```
[HttpGet]
[Authorize(Roles = "Admin")]
0 references
public async Task<IActionResult> GetPurchases()
```

8.4. ábra. Szerepkörhöz kötött bejelentkezés

8.1.4. Specifikus controllerek és servicek

Users

Ez a controller-service páros a bejelentkezett, illetve a nem bejelentkezett felhasználókhöz kötött adatlekérések, beszúrások illetve módosításokért felelősek. A belépés, regisztráció illetve jelszó elfelejtéshez szükséges email küldése SMTP segítségével illetve ennek validálása nem szükséges JWT token, így bárki hozzáférhet ezekhez. A regisztrációkor a felhasználó megadja a személyes adatait, ezek ha érvényesek illetve még nem regisztrált ezzel az email címmel, akkor hitelesítő emailt kap, amely egy bizonyos ideig érvényes. Emailben megkapja az adott linket, amely segítségével hitelesítheti magát. A linkben tárolva van az adott felhasználó email címe, illetve a hozzá rendelt hitelesítő kód. A metódus GET formájú a paraméterek átadása miatt. Az email küldése SMTP segítségével történik, meg kell adni az adott hostot, portot, illetve a küldő email címét, valamint a Google környezetben generált kódját a külső applikációkból való email küldésre. Bejelentkezéskor a felhasználó email és jelszó alapján kérést küld, válaszként visszakapja a tokenet. Elfelejtett jelszó esetén a felhasználó egy linket kódot az emailjére, amelyet megadva az új jelszóval együtt módosíthatja a jelszavát. Ez a kód egyedi kell legyen, ezért addig generáltatom egy while ciklusban, ameddig nem kapok egy érvényes kódot. A felhasználó lekérheti a saját adatait, az azonosítóját a fejlécben átadott tokenből adja át. Az admin hozzáférhet az összes felhasználó listájához, természetesen a jelszó rejtve marad.

```
StringValues values;
var res = Request.Headers.TryGetValue("Token", out values);
var user = await _userService.GetUserFromToken(values);
```

8.5. ábra. Fejlécből kapott felhasználó

Furnitures

A felhasználók lekérhetik az adatbázisunkban szereplő bútorok listáját, amelyben a specifikus adatok mellett szerepel hogy hányas az adott példány átlagértékelése, hány

darab van raktáron, hányan vásárolták, illetve hányan értékelték az adott búttrot. Az hogy a bútorhoz milyen kép van rendelve egy string tárolja, amely az adott jpg vagy png publikus elérési útvonalát mutatja. Ha adott azonosítójú bútor adatait szeretnénk lekérni, akkor válaszként megkapjuk hogy hányadik legjobb értékelésű az adott kategóriában, illetve hogy hányadik legjobban vásárolt az adott kategóriában. Ezt úgy oldottam meg hogy kiszámoltam minden bútor értékelésének az átlagát majd ezek alapján rendeztem (bútorId, átlag) párosban, majd rendeztem átlag alapján és visszatérítettem a páros pozícióját. Ugyanakkor megkapjuk a képhez rendelt összes értékelés listáját. Hogyha van tárolva 3D-s model is tárolva akkor annak elérési útvonalát megkapjuk. A hiteles méret beállításához egy adott virtuális szobában szükséges a skálázás értéke is olykor, ehhez az értékhez is hozzáférhetünk. Az admin hozzáadhat illetve módosíthat bútrokat, rendelhet leárazást is hozzá, valamint módosíthatja a raktáron lévő darabszámát. A hozzá társított kép útvonalát is módosíthatja.

Reviews

A felhasználó értékelést adhat egy adott termékre. 1-től 5-ös skálán adhatja meg a minőségét, valamint opcionálisan megjegyzést is. Az admin illetve a felhasználó is megkapja az értékelések listáját.

Purchases

Az admin megtekintheti az összes vásárlást, illetve hogy milyen termékek voltak vásárolva ezen belül. Egy vásárlás egy adott felhasználó és vásárlás idejének párosából jön létre. A vásárolt bútorok darabszáma és ára is szerepel a listában, amely változhat leárazás esetén. Az összár is szerepel a listában. A felhasználó vásárláskor egy (bútorId, darabszám) párosból álló listát ad meg, hogyha bármelyik kép nincs az adatbázisban vagy nincs elég darab belőle raktáron, akkor a vásárlás nem valósul meg. A kiszállításhoz szükséges lakkím megadása is szükséges.

```
public async Task<PurchaseServiceResponses> AddPurchases(List<PurchaseEntity> Purchases, int userId, string Address)
{
    var PurchaseDate = DateTime.Now;
    foreach (var p in Purchases)
    {
        var furniture = await _context.Furnitures.AsNoTracking().FirstOrDefaultAsync(f => f.Id == p.FurnitureId);
        if (furniture == null)
            return PurchaseServiceResponses.FURNITURENOTFOUND;
        if (furniture.AvailableQuantity < p.Quantity)
            return PurchaseServiceResponses.NOTEENOUGHQUANTITY;

        var discount = await _context.Discounts.FirstOrDefaultAsync(d => d.FurnitureId == furniture.Id);

        p.Price = p.Quantity * furniture.Price;
        if (discount != null)
            p.Price -= (p.Price * discount.Percentage)/100;
        p.UserId = userId;
        p.Date = PurchaseDate;
        p.Address = Address;

        _context.Purchases.Add(p);
        furniture.AvailableQuantity -= p.Quantity;
        _context.Entry(furniture).State = EntityState.Modified;
    }
}
```

8.6. ábra. Bútorok vásárlása

9. fejezet

Android

9.0.1. Arhitektúra

Az felhasználói felület android keretrendszerben van megírva. Két fő activitynk van, az activitykhez fragmentek vannak inflatelve, amelyek figyelik a viewmodelben lévő LiveDatakat. A livedatak legtöbbször api hívás során kapnak értéket az adott viewmodel repositotyjából, a repositoryk hajtják végre az api hívást Retrofit könyvtár segítségével.

9.0.2. Fontosabb osztályok és könyvtárak

Retrofit

A Retrofit egy olyan könyvtár, amelynek segítségével api hívásokat hajthatunk végre. A Retrofit egyed létrehozásakor szükséges megadni az api URL-jét, valamint a hozzá kapcsolódó servicet, amely tartalmazza az api hívásokat.

ViewModel

A viewmodellek segítenek megőrizni az adatokat konfigurációs változások, például a képernyő orientációjának változásakor. A viewmodellek lehetnek úgy az activity, mint a viewmodel szintjén, activity szinten minden fragmenten keresztül ugyanaz marad, nem fog többször létrejönni. Azért, hogy ne az először megkapott adat legyen mindig az applikációban, gyakran alkalmazhatunk api hívást.

Coroutines

Aszinkron programozáshoz használható szálak, amelyeknek többféle scopeja lehet, például globális, vagy éppen a saját scopejához kötött. Használatukkal megoldható, hogy hosszabb művelet során ne blokkolódjon az applikációnk fő szála.

Google AR Core

Egy olyan eszközökészlet, amely lehetővé teszi a valós és virtuális világ kombinációját a felhasználó számára. Támogatja a 3D-s modellek használatát, akár GLB, akár GLTF fromátumban.

9.0.3. Konkrét megvalósítások

Applikációmban minden api híváskor a viemodel megkapja a repositoryból származó api hívás eredményét, ha hibát észlel kivétel dobódik, amit elkap a viewmodel. A viewmodel értéket ad a tartalmazott LiveData-nak. Exception esetén a hibaüzenet Livedata kap értéket, a fragment ezt figyelve általában Toast formájában jeleníti meg az üzenetet. Lo-

ginActivity minden bemeneti adatot igénylő mező helyessége le van kezelve felhasználói szinten, a hibákat a mezők körül jelenítettem meg. Csak akkor küld kérést a felhasználó ha nincs hiba UI szinten. Sikeres bejelentkezéskor megkapjuk a felhasználó adatait, illetve a szerepkörökhez szükséges tokent. Ezeket egy adott SharedPreferencesben tároljuk, hogy később alkalmazhassuk az applikációban őket.

MainActivity

Az activitynk elindulásakor a bútorok listáját tartalmazó fragment kerül betöltésre. Az elemek recyclerviewban vannak elhelyezve, a.recyclerviewnak van egy specifikus adapttere, ez felel az adatok megjelenítéséért a képernyőn. Ezen a képernyőn a keresést a searchview onTextChangeListener metódusának segítségével valósítottam meg, illetve az elemek rendezését spinner segítségével oldottam meg. Ha a felhasználó admin szerepkörű, megjelenik egy gomb amire kattintva átnavigálhat a bútor hozzáadása képernyőre. A szerepkört a SharedPreferencesból kapom meg, s ennek fülfényében változik a gomb láthatósága. Egy adott elemre kattintva a képernyőn megjelennek az adott bútor adatai, ha Adminként kattint rá akkor megjelennek az adatai EditTextekben, ezeket módosítva frissíti a bútor, felhasználóként pedig csak megjeníti az adatakat az adott viewmodelből api hívás következtében. Adminként ezen a képernyőn tartalmat tölthet fel a bútor vizuális tartalmához Cloudinary segítségével. A feltöltött kép url-jét az adatbázisban tárolja. Ha a bútorhoz van rendelve 3D-s objetum, akkor átkerülünk az ArFragmentre, ahol valós kamera vagy éppen virtuális szoba segítségével elhelyezhetjük az adott bútor a térből. Ezt a Google ARCore szoftver segítségével valósítottam meg. A kiválasztott 3D-s model elérési útvonalából létrehoz egy objetumot, amelyet kattintáskor elhelyez egy fix helyen a térből AnchorNode és az ehhez rendelt Chil node segítségével.

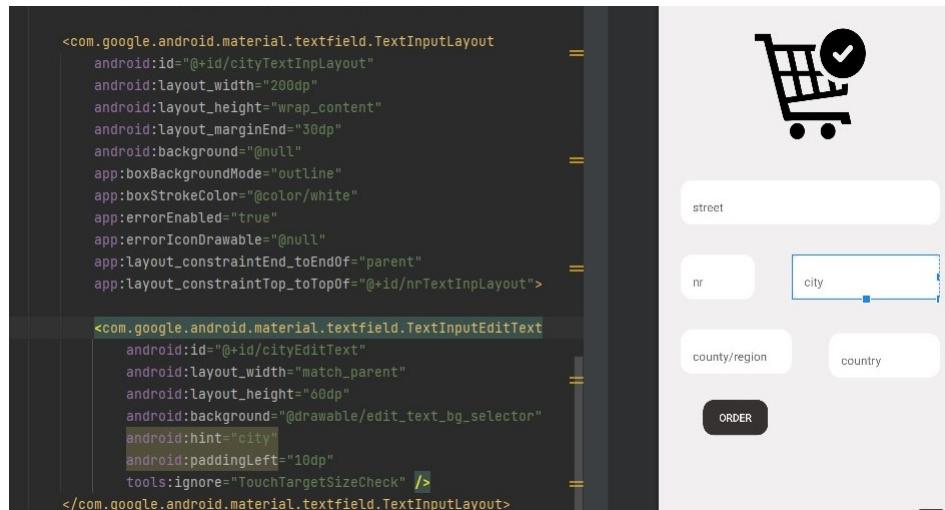
A kosár képernyőn recycleviewban szerepelnek a kiválasztott elemek. Változtathatjuk a kiválasztott bútorok számár, majd rendelést helyezhetünk el. Ha változtatjuk a darabszámot, változik az összár is, ezt LiveData segítségével valósítottam meg. Az admin ugyanezen a képernyőn API hívás segítségével megkapja az összes vásárlás listáját. Ezeket rendezheti különböző kritériumok alapján. Egy elemre kattintva megjelenik az adott vásárláskor kosárba rakott elemek listája és darabszáma, ez az adminhoz rendelt viewmodelben lesz tárolva, s ezen a képernyőn már nem lehet darabszámot változtani. A profil fragmenten a felhasználó megnézheti a saját adatait, az admin az összes felhasználót. Mindkét felhasználó innen jelentkezik ki.

10. fejezet

Felhasználói felület

10.1. Tervezés

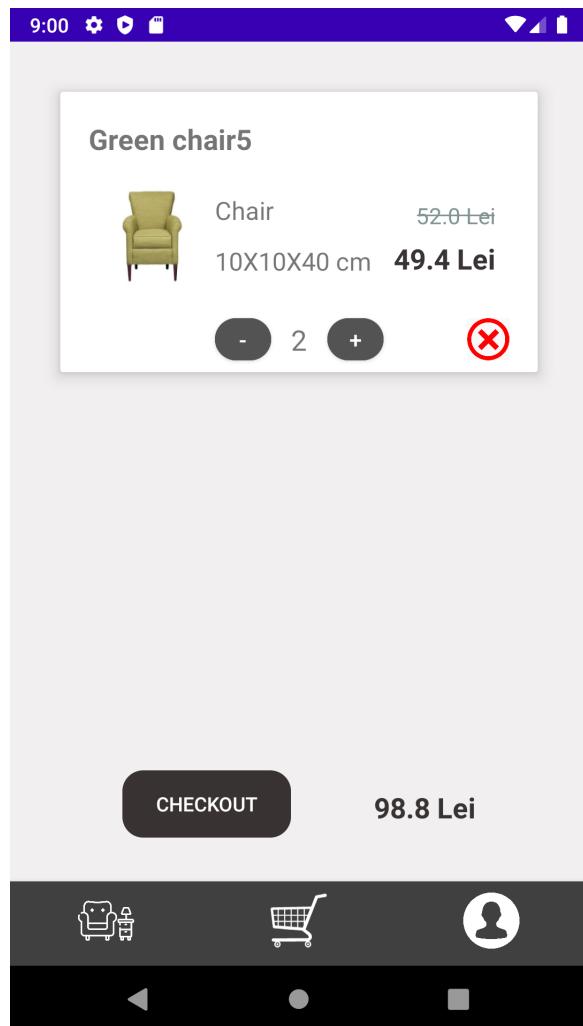
A felhasználói felület megtervezésében a hasonló típusú alkalmazásokból inspirálódtam, hiszen a bejelentkezés illetve az áruház vizuális megjelenése az ilyen alkalmazások többségében általában hasonló sémát követ. Az egyszerűségre való törekvés segített előre abban, hogy az alkalmazás minden felhasználó számára gördülékenyen használható legyen, függetlenül a felhasználó életkorától, esetleges digitális kompetenciáinak hiányosságától. Az alkalmazás megjelenése könnyen átlátható illetve a navigációt a szöveges illetve ábrák általi jelzések segítik elő. minden esetben a képernyőn megjelenő szövegek és ikonok egyértelműen jelzik a felhasználó számára, hogy milyen lehetőségei vannak azon folyamat folytatására, amelyet elkezdett.



10.1. ábra. TextView és a hozzá társuló "avatar"

10.2. Megvalósítás

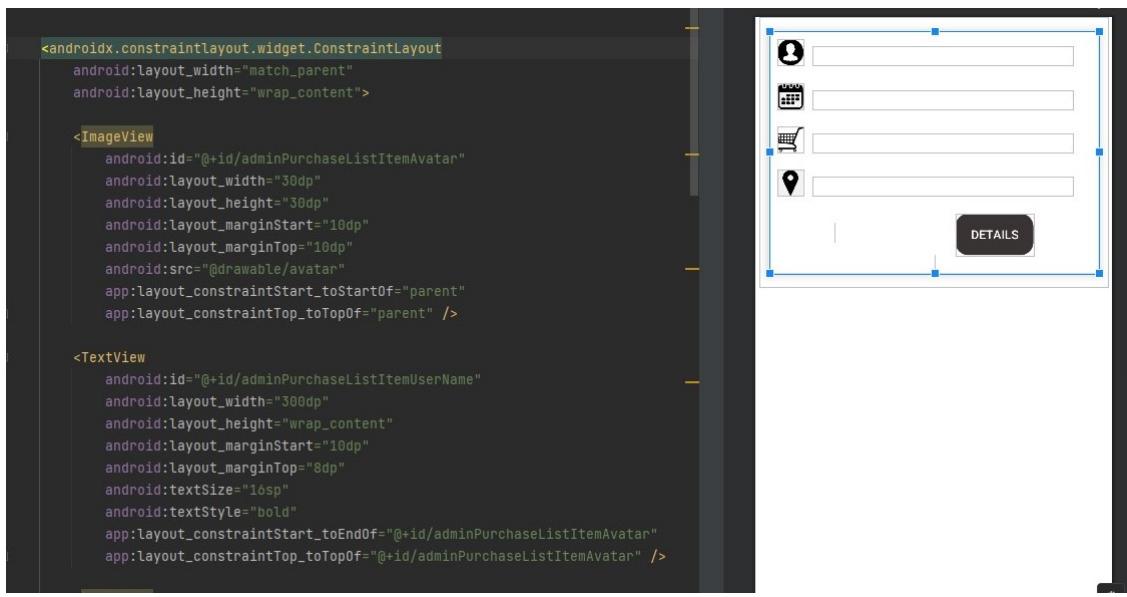
Az alkalmazás beszédes megjelenését elsősorban a szöveges segítségek biztosítják, amelyek visszatérő elemek úgy a képernyőn megjelenő magyarázó szándékú szövegek,



10.2. ábra. Beszédes névvel vagy ikonokkal ellátott gombok

mint a hibaüzenetek pontos megfogalmazása illetve a beszédes nevekkel ellátott mezők és gombok által. Az olyan mezők, amelyek a felhasználó általi kitöltésre várnak, beszédes súgó szövegekkel (hintek) vannak ellátva, annak érdekében, hogy a bevitt adatok biztosan helyesek, vagy akár megfelelő formátumúak, mértékegységűek legyenek.

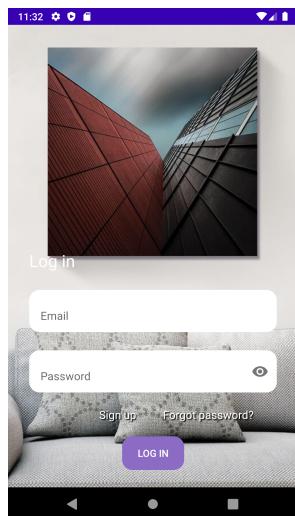
A képernyő azon részein, ahol nem jelennek meg szöveges elemek vagy utasítások, képes jelzések segítik a felhasználót. Ez leginkább a navigációs sávra jellemző illetve esetenként a szöveges adatok megjelenítése mellett olyan képek jelennek meg, amelyek jelzik, hogy az adat mire vonatkozik. Mindezekre az alábbiakban láthatunk konkrét példákat.



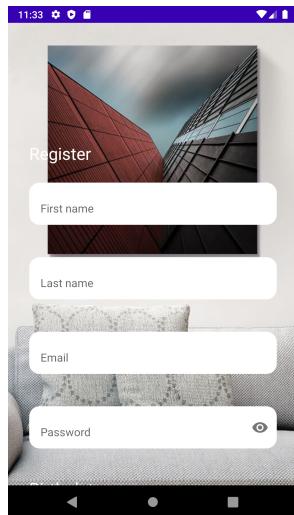
10.3. ábra. TextView és a hozzá társuló "avatar"

11. fejezet

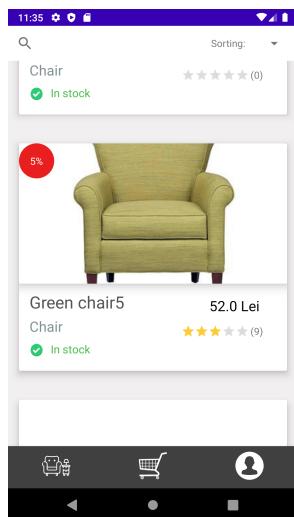
Az alkalmazás működésének bemutatása képekkel



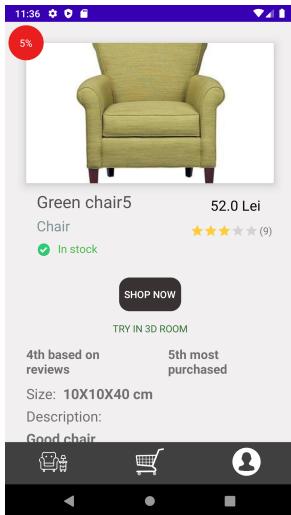
11.1. ábra. Login képernyő



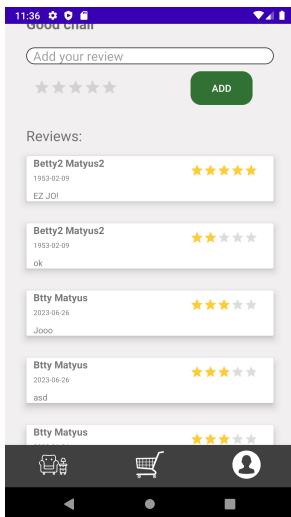
11.2. ábra. Regisztráció képernyő"



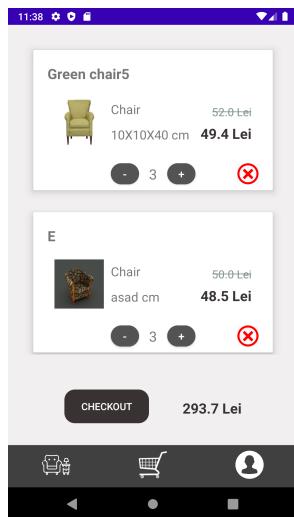
11.3. ábra. Főoldal



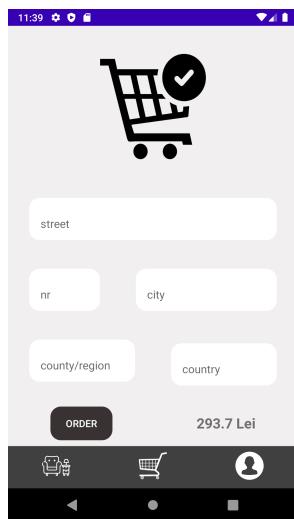
11.4. ábra. Termék adatai



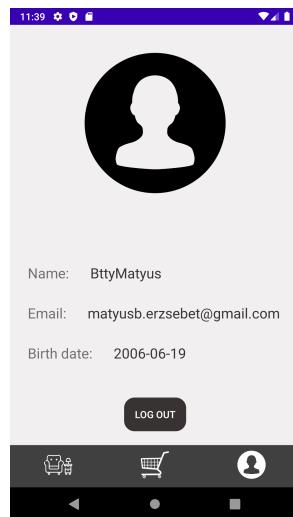
11.5. ábra. Termék Értékelései



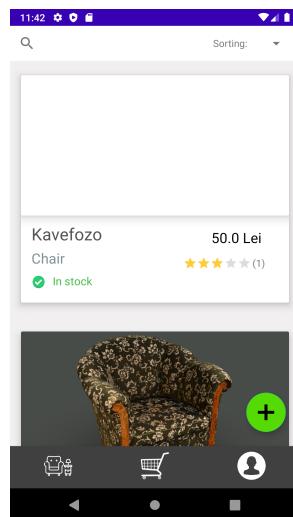
11.6. ábra. Kosár



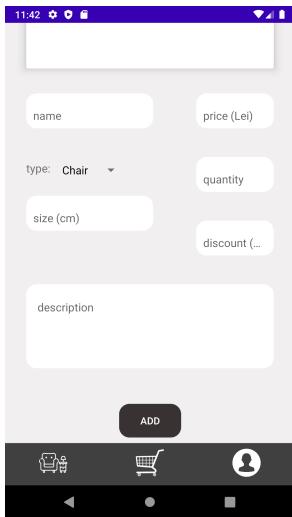
11.7. ábra. Rendelés Véglegesítése



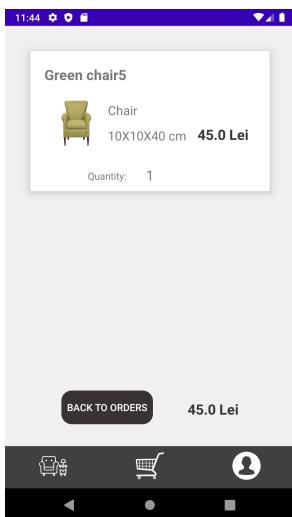
11.8. ábra. Profil



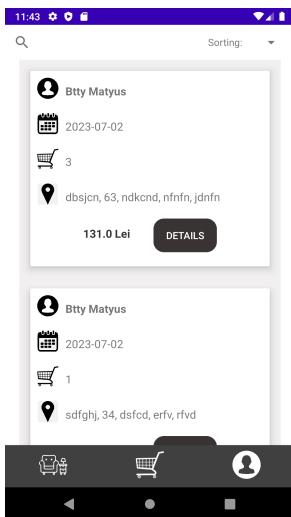
11.9. ábra. Főoldal hozzáadás gombbal



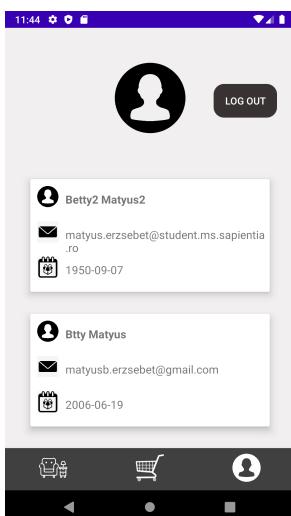
11.10. ábra. Bútor hozzáadása



11.11. ábra. Rendelés részletei



11.12. ábra. Rendelések



11.13. ábra. Felhasználók listája

12. fejezet

Továbbfejlesztési lehetőségek

Az applikáció végeredményben pontosnak, és megbízhatónak nevezhető, a felhasználó könnyen tud boldogulni ezzel a projekttel. Ennek ellenére, az applikáció egyelőre kezdetleges, ami rengeteg ötletnek szolgálhat biztos alapot majd a jövőben. Az applikáció jelen állapotában, hagy helyet a továbbfejlesztésnek. Annak érdekében, hogy még érdekesebb legyen az applikáció több funkcióval kéne rendelkezzen: Kártyás fizetés: A kártyás fizetési rendszer fejlesztése során kiemelkedő fontosságú lenne az adatbiztonság és a felhasználói bizalom növelése. Ennek érdekében javasolható lenne a kártyaadatok további védelme, például a tokenizáció vagy a vásárlások többszintű hitelesítése. Emellett az alkalmazásban történő kifizetések gyorsaságának és megbízhatóságának javítása is fontos szempont lenne. Token frissítése: A tokenek automatikus frissítése, amikor azok lejárnak, lehetővé tenné a felhasználók számára, hogy zavartalanul folytathassák az alkalmazás használatát. Ez azt jelentné, hogy a felhasználók nem kellene manuálisan frissíteniük a tokeneket, hanem az alkalmazás automatikusan megújítaná azokat a háttérben. Jelszó titkosítása: A jelszavak megfelelő titkosítása kulcsfontosságú a felhasználói fiókok védelme szempontjából. A jelszavak egyszeri titkosítása (hashing) mellett kiegészítő védelmi rétegeket is be lehet vezetni, például sózás (salting) vagy kulcskezelési technikák használatával. Ez biztosítaná, hogy még a kiszivárgott adatok esetén is nehéz lenne visszafejteni a felhasználói jelszavakat. Értesítések küldése az alkalmazáson keresztül: Az alkalmazáson belüli értesítési rendszer kialakítása lehetővé tenné, hogy a felhasználók részletes és releváns értesítéseket kapjanak a lejáró szolgáltatásokról, fontos eseményekről vagy egyéb információkról. Ez a megközelítés megkönnyítené a kommunikációt a felhasználók és az alkalmazás között, és minimalizálná a szükségességet a külső e-mail értesítésekre.

Összefoglaló

Az online vásárlás az elmúlt időszakban kiemelkedően fontossá vált, és én, mint fejlesztő, büszke vagyok arra, hogy egy olyan alkalmazást hoztam létre, amely megkönnyíti a vásárlók életét. Az alkalmazás lehetővé teszi a felhasználók számára, hogy kényelmesen vásároljanak otthonról, és virtuális szobákban helyezzék el a kiválasztott bútorokat és dekorációkat.

Az adminisztrátori felületet személyesen terveztem úgy, hogy egyszerűen kezelhető legyen és lehetőséget adjon az összes adat feltöltésére és frissítésére. A céлом az volt, hogy az adminisztrátoroknak könnyű legyen a termék- és felhasználói információk kezelése, így időt és erőfeszítést takaríthatnak meg.

Az alkalmazás felhasználói felülete az én tervezésem eredménye, és hangsúlyt fektettem a felhasználóbarát élményre. Célom az volt, hogy a felhasználók zökkenőmentesen tudjanak vásárolni, és a virtuális szobákban való tárgyak elrendezése közben is könnyen navigálhassanak. A visszaküldések elkerülésére is figyelem, lehetővé téve a termékek több szögből történő valósághű megtekintését és vizsgálatát.

Az általam kifejlesztett alkalmazás a kényelem és a rugalmasság előnyeit nyújtja mind a vásárlóknak, mind az eladóknak. A vásárlók élvezhetik a kényelmet és a biztonságot, miközben otthonról vásárolnak, és a termékek virtuális környezetben való megtekintése valósághű élményt nyújt számukra. Az adminisztrátorok pedig könnyedén kezelhetik és frissíthetik az adatokat az alkalmazásban.

Rendkívül elégedett vagyok a fejlesztésem eredményével, és büszke vagyok arra, hogy az alkalmazásom segítséget nyújt a vásárlóknak és elősegíti a kényelmes és hatékony online vásárlást.

Aa projekt fejlesztési folyamatát a GitHubomon lehet követni, az alábbi linken:
<https://github.com/MatyusErzsebet/Allamvizsga.git>

Ábrák jegyzéke

2.1. Git	13
2.2. GitHub	14
2.3. Postman	14
2.4. Visual Studio 2022	15
2.5. .NET Framework	16
2.6. SQL Server management Studio	17
2.7. AR Core	19
2.8. Android	20
3.1. Microsoft SQL Server	21
3.2. Adatbázis diagramja	23
4.1. Use Case Diagram	26
7.1. HttpGet példa szemléltetése	32
7.2. API végpontok	33
7.3. Példa a kérésre kapott status codera	34
7.4. Példa JSON válaszra	35
8.1. Connection String	37
8.2. Idegen kúlcs kapcsolat	37
8.3. Adatbázis létrehozásának szabályozása	37
8.4. Szerepkörhöz kötött bejelentkezés	38
8.5. Fejlécből kapott felhasználó	38
8.6. Bútorok vásárlása	40
10.1. TextView és a hozzá társuló "avatar"	43
10.2. Beszédes névvel vagy ikonokkal ellátott gombok	44
10.3. TextView és a hozzá társuló "avatar"	45
11.1. Login képernyő	46
11.2. Regisztráció képernyő"	47
11.3. Főoldal	47
11.4. Termék adatai	48
11.5. Termék Értékelései	48
11.6. Kosár	49
11.7. Rendelés Véglegesítése	49
11.8. Profil	50
11.9. Főoldal hozzáadás gombbal	50

11.10Bútor hozzáadása	51
11.11Rendelés részletei	51
11.12Rendelések	52
11.13Felhasználók listája	52

Irodalomjegyzék

<https://developer.android.com/studio/run/emulator>

<https://www.techtarget.com/whatis/definition/Kotlin>

<https://www.xenonstack.com/blog/kotlin-android-comparison>

<https://developers.google.com/ar/develop>

https://en.wikipedia.org/wiki/SQL_Server_Management_Studio

<https://learn.microsoft.com/en-us/aspnet/core/tutorials/web-api-help-pages-using-swagger?view=aspnetcore-7.0>

<https://docs.bmc.com/docs/BMCHelixPortal/role-endpoints-in-the-rest-api-982335803.html>

https://en.wikipedia.org/wiki/Android_software_development

<https://nulab.com/learn/software-development/how-a-uml-use-case-diagram-can-benefit-any-process/>

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

<https://auth0.com/blog/role-based-authorization-for-aspnet-webapi/>

https://en.wikipedia.org/wiki/Microsoft_SQL_Server#cite_note-27

<https://docs.github.com/en/organizations/managing-user-access-to-your-organizations-repositories/repository-roles-for-an-organization/>

<https://www.postman.com/api-documentation-tool/>

<https://learn.microsoft.com/en-us/visualstudio/windows/?view=vs-2022>

<https://learn.microsoft.com/en-us/dotnet/framework/>