ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

NEUMANN JÁNOS
INFORMATIKAI KAR

SZAKDOLGOZAT

OE-NIK Hallgató neve: **Szalai Gergő**
2022 Hallgató törzskönyvi száma: **T/005822/FI12904/N**

Óbudai Egyetem
Neumann János Informatikai Kar
Kiberfizikai Rendszerek Intézet

SZAKDOLGOZAT
FELADATLAP

Hallgató neve: **Szalai Gergő**
Törzskönyvi száma: T/005822/FI12904/N
Neptun kódja: [redacted]

A dolgozat címe:

Állampapírok nyilvántartására kialakított rendszer fejlesztése
System development for registration of government securities

Intézményi konzulens: Dr. Holyinka Péter
Külső konzulens:

Beadási határidő: 2022. május 15.

A záróvizsga tárgyai: Számítógép architektúrák
Big Data és üzleti intelligencia specializáció

A feladat

Ismerje meg a vásárolt állampapírok historikus nyilvántartásával kapcsolatos igényeket. Határozza meg az igények kielégítését lehetővé tevő követelményeket, különös tekintettel az adatok anonimizálására. Határozza meg a rendszer fejlesztéséhez szükséges eszközöket. Tervezze meg és fejlessze ki a rendszert.

A dolgozatnak tartalmaznia kell:

- a rendszerrel szemben támasztott igényeket és követelményeket,
- a rendszer teveit,
- a megvalósításhoz választott eszközöket és a választás indoklását,
- a tesztelési terveket és a tesztelés eredményeit,
- az elért eredményeket és a továbbfejlesztési lehetőségeket.

Ph.

.....
Dr. Fleiner Rita
intézetigazgató

A szakdolgozat elévülésének határideje: **2024. május 15.**
(OE TVSz 55.§ szerint)

A dolgozatot beadásra alkalmasnak tartom:

.....
külső konzulens

.....
intézményi konzulens

HALLGATÓI NYILATKOZAT

Alulírott hallgató kijelentem, hogy a szakdolgozat / diplomamunka saját munkám eredménye, a felhasznált szakirodalmat és eszközöket azonosíthatóan közöltem. Az elkészült szakdolgozatomban / diplomamunkámban található eredményeket az egyetem és a feladatot kiíró intézmény saját céljára térítés nélkül felhasználhatja.

Budapest, 2022.05.05.

.....
hallgató aláírása

KONZULTÁCIÓS NAPLÓ

Hallgató neve: SZALAI GERGŐ

Neptun kód: [redacted]

Tagozat: NAPPALI

Telefon: [redacted]

Levelezési cím (pl: lakcím): [redacted]

Szakdolgozat / Diplomamunka[1] címe magyarul:

ÁLLAMPAPÍROK NYILVÁNTARTÁSÁRA KIALAKÍTOTT RENDSZER FEJLESZTÉSE

Szakdolgozat / Diplomamunka[2] címe angolul:

SYSTEM DEVELOPMENT FOR REGISTRATION OF GOVERNMENT SECURITIES

Intézményi konzulens: DR. HOLVIAKA PÉTER

Külső konzulens: .....

Kérjük, hogy az adatokat nyomtatott nagybetűkkel írja!

Alk.	Dátum	Tartalom	Aláírás
1.	2021.09.21.	Téma kiválasztása, feladat leírása	
2.	2021.10.15.	Felépítés megbeszélése	
3.	2021.11.19.	Üzleti folyamatok áttekintése	
4.	2021.12.05.	Specifikáció leírása	

A Konzultációs naplót összesen 4 alkalommal, az egyes konzultációk alkalmával kell láttamoztatni bármelyik konzulenssel.

A hallgató a Szakdolgozat I. / Szakdolgozat II. (BSc) vagy Diplomamunka 1 / Diplomamunka 2 / Diplomamunka 3 / Diplomamunka 4[3] tantárgy követelményét teljesítette, beszámolóra / védésre[4] bocsátható.

Budapest, 20 21.12.12.

.....
intézményi konzulens

[1] Megfelelő aláhúzandó!
[2] Megfelelő aláhúzandó!
[3] Megfelelő aláhúzandó!
[4] Megfelelő aláhúzandó!

KONZULTÁCIÓS NAPLÓ

Hallgató neve: SZALAI GERGŐ

Neptun kód: [redacted]

Tagozat: NAPPALI

Telefon: [redacted]

Levelezési cím (pl: lakcím): [redacted]

Szakdolgozat / Diplomamunka[1] címe magyarul:

ÁLLAMPAPÍROK NYILVÁNTARTÁSÁRA KIALAKÍTOTT RENDSZER FEJLESZTÉSE

Szakdolgozat / Diplomamunka[2] címe angolul:

SYSTEM DEVELOPMENT FOR REGISTRATION OF GOVERNMENT SECURITIES

Intézményi konzulens: DR. HOLVINKA PÉTER

Külső konzulens:

Kérjük, hogy az adatokat nyomtatott nagybetűkkel írja!

Alk.	Dátum	Tartalom	Aláírás
1.	2022.02.21.	Adatbázis előkészítése	
2.	2022.03.14.	Rendszerfolyamatokkal kapcsolatos kérdések megvitatása	
3.	2022.04.11.	Felhasználói interfész, elemzések	
4.	2022.05.02.	Áttekintés, javítások megbeszélése	

A Konzultációs naplót összesen 4 alkalommal, az egyes konzultációk alkalmával kell láttamoztatni bármelyik konzulenssel.

A hallgató a Szakdolgozat I. / Szakdolgozat II. (BSc) vagy Diplomamunka 1 / Diplo-mamunka 2 / Diplomamunka 3 / Diplomamunka 4[3] tantárgy követelményét teljesítette, beszámolóra / védésre[4] bocsátható.

Budapest, 2022. 05. 09.

Holvinka Péter

intézményi konzulens

[1] Megfelelő aláhúzandó!
[2] Megfelelő aláhúzandó!
[3] Megfelelő aláhúzandó!
[4] Megfelelő aláhúzandó!

TARTALOMJEGYZÉK

ÁBRAJEGYZÉK

TÁBLÁZATJEGYZÉK

1. BEVEZETÉS

Napjainkban megannyi lehetőséggel és különféle megoldásokkal találkozunk, ha szoftverfejlesztésben gondolkodunk. Egy megbízható, korszerű rendszer kiépítése igensok tervezési időt igényel, melyhez a megfelelő információ összegyűjtése talán az egyik legidőigényesebb feladat. Fontos, hogy az egyes komponenseket úgy válasszuk meg, hogy azok a lehető legjobban illeszkedjenek a mai trendekhez, valamint az ügyfél elvárásainak eleget tegyenek.

Mivel az adat játssza a legfontosabb szerepet egy információs rendszer fejlesztésében, fontos, hogy azok a lehető legbiztonságosabb módon, esetleg elosztott módon legyenek tárolva. Az adatok visszakövethetősége szintén lényeges szempont, hiszen ezek alapján leszünk képesek szolgáltatni olyan információt, amely adott cég esetében szükséges a megfelelő stratégiai döntés meghozatalához. A fejlesztendő szoftver CRM-nek tekintendő. A szoftvert több irányból közelíthetjük meg, ugyanis nem csak a vállalat részére, hanem az ügyfél és akár a forgalmazó részére is használható. Így tehát egy olyan rugalmasan használható, bármelyik fél részére kialakított rendszert terveztem meg és fejlesztettem le, melynek potenciálja az alkalmazás flexibilitásán, gyorsaságán és elérhetőségében rejlik.

A szakdolgozat első részében tisztázom az alapvető fogalmakat és ismertetem a fontosabb keretrendszereket, amelyek szükségesek a későbbiekben bemutatott fejlesztés megértéséhez. Ezt követően bemutatom az általam fejlesztett platformfüggetlen (web és mobil) alkalmazást a backendtől a frontendig, mellyel célom, hogy az olvasó egy teljes és átfogó képet kapjon egy piacképes pénzügyi informatikai rendszer kifejlesztéséről. Munka- és tanulmányi tapasztalataimat felhasználva fejlesztettem egy web- és mobilalkalmazást, amely magában foglalja az adattároláshoz szükséges adatbázisokat, az üzleti folyamatok lebonyolítására szolgáló API réteget, valamint az információ megjelenítéséhez szükséges felhasználói interfészt. Az összesített adatokból végezetül elemzéseket készítettem.

2. IRODALOMKUTATÁS

Ebben a fejezetben ismertetem a munkámhoz szükséges alapvető kifejezéseket, fogalmakat és alapvető módszertanokat. Az egyes alfejezetekben leírtak építőelemként szolgálnak az informatikai rendszer létrehozásához.

2.1 Állampapírok és fogalmi meghatározások

Kezdetnek bemutatom az általam fejlesztett rendszer egyik alapvető elemét, az állampapírt. Az alfejezetben tisztázom a fogalmakat és ismertetem a tudnivalókat, amelyek a rendszer megértéséhez szükségesek.

„Az állampapír az állam által kibocsátott hitelviszonyt megtestesítő értékpapír. Az állampapír vásárlásával tulajdonképpen az államnak adunk kölcsönt, előre meghatározott kamatra és időre. Ez utóbbi – azaz az állampapír futamideje - alapján megkülönböztetünk kincstárjegyet (1 éves vagy annál rövidebb lejáratú) és államkötvényt (1 évnél hosszabb lejáratú).” [1]

2.1.1 Névérték

„Az adott hitelviszonyt megtestesítő értékpapír tekintetében az a pénzben kifejezett összeg, amely a főkövetelést takarja. Az adott értékpapír ekkora összegű, az értékpapír jogosultját megillető tőkekövetelést testesít meg. A kamatozó értékpapíroknál a kamatok kiszámítása a névérték alapján történik.” [1]

2.1.2 Kamat, felhalmozott kamat

„A hitelviszonyt megtestesítő értékpapír névértéke vetített, a kamatfizetés időpont(ok)ban kifizetésre kerülő összeg százalékos formában kifejezett értéke.”

„Az adott állampapír tekintetében a kibocsátás napja, vagy az utolsó kamatfizetés napja óta felgyülemlett, de esedékessé még nem vált kamatot felhalmozott kamatnak hívjuk.” [1]

2.1.3 Futamidő

Az az időtartam, amely alatt a hitel visszafizetése teljesül. [1]

2.1.4 Nettó és bruttó árfolyam

„Az adott állampapír felhalmozott kamat nélküli, aktuális árfolyama. A bruttó árfolyam a nettó árfolyam, valamint a felhalmozott kamat összege.” [1]

2.1.5 Portfólió

Azon értékpapírok összessége, amelyeket egy befektető adott időpontban birtokol.

2.2 Ügyfélkapcsolat-menedzsment (CRM)

Manapság a Customer Relationship Management rendszerek meglehetősen magas potenciállal bírnak ahhoz, hogy egy vállalat sikeréhez, illetve növekedéséhez hozzájáruljanak.

2.2.1 Meghatározás

A CRM definiálása meglehetősen homályos, nincs konkrét meghatározás, sokkal inkább egy divatos szakkifejezés. Több komponensből áll, ezek lehetnek: technológia, folyamatok és emberek. Az ügyfélkapcsolat-kezelés lehetővé teszi, hogy a vállalat jobban megismerhesse ügyfeleit, továbbá egy tartós kapcsolatot tudjon kiépíteni velük. Az efféle kötelék mindkét fél érdeke.

A fogalmat többféleképpen is értelmezhetjük. Egyfajta üzleti filozófiaként állíthatjuk, hogy a CRM egy olyan kapcsolat, amely értéket teremt mind a vállalat, mind pedig az ügyfél részére. Tekinthetjük egy ügyfél központú stratégiának, amely elsősorban az ügyfél elégedettségére fókuszál, melynek során a vállalat elnyerheti az adott személy hűségét, ezzel elősegítve a személyre szabott szolgáltatások megjelenését. Egy másik értelmezésben pedig a CRM nem más, mint egy technológia, amely lehetővé teszi ügyfél és vállalat számára, hogy szorosabb kapcsolatot ápolhassanak. [2]

2.2.2 Formák

Az ügyfélkapcsolat-menedzsmentnek 3 formája létezik, melyek a gyakorlatban aligha különülnek el egymástól, inkább a rendszer céljaiban lelhetőek fel különbségek.

Az operatív rendszer a legelterjedtebb forma. Ezen rendszerek fő célja az értékesítés, marketing és szolgáltatások automatizálása.

Az analitikus architektúra az operatív társára épít, mivel az abból kinyert adatokkal dolgozik. Az ügyfelek kiszolgálásának maximalizálására törekszik a különböző érintkezési pontból jövő ügyféladatok értelmezésével, elemzésével, mellyel célja, hogy átláthatóbb képet kapjon a vállalat tevékenységeinek sikerességéről.

A kollaboratív, vagy más néven stratégiai CRM célja, hogy támogassa az egyes vállalaton belüli, részlegek közti kommunikációt. Normál esetben a részlegek egymástól külön tevékenykednek, azonban akadnak olyan helyzetek, amikor azok közösen együttműködve képesek megoldani egy üzleti problémát. Erre kiváló példa, hogy az

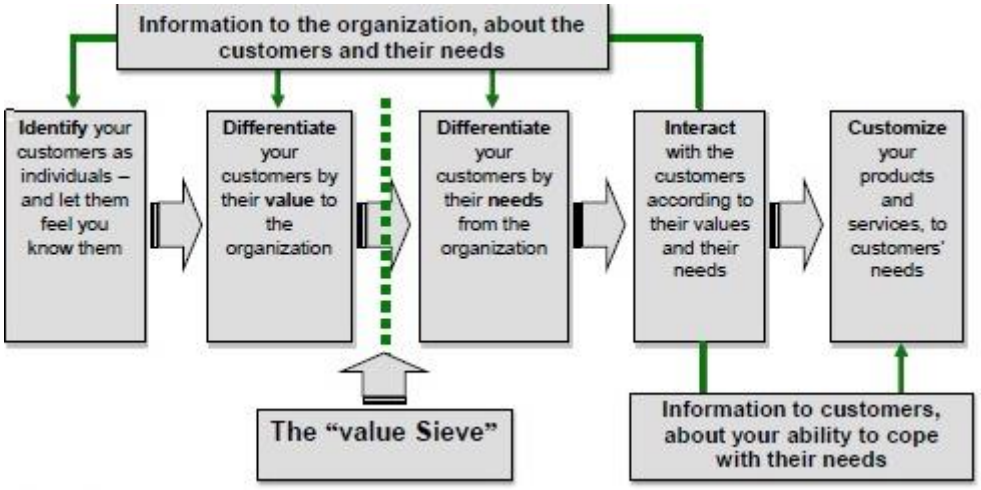
ügyfélszolgálat, vagy értékesítés birtokában lévő információ szolgálhatja a marketing érdekeit is, amellyel elősegíthetik a minőségibb szolgáltatásokat. [2][3]

2.2.3 CRM modellek

Többféle ügyfélkapcsolat-menedzsment modell is létezik. Ebben az alfejezetben szeretném a három talán legjelentősebbet megemlíteni.

Létezik egy ún. IDIC módszer. Ennek lényege, hogy ahhoz, hogy egy hatékony CRM rendszert tudjunk létrehozni, négy kulcsfontosságú tényező mentén kell elindulnunk:

- Identify: ügyfelek azonosítása
- Differentiating: az ügyfeleket aszerint kell beazonosítani, hogy mekkora értéket képviselnek az adott cél elérése érdekében
- Interacting: kapcsolat létesítése az ügyféllel, hogy a vállalat megértse az igényeket
- Customizing: személyre szabott ajánlatok és kommunikáció az ügyfél részére



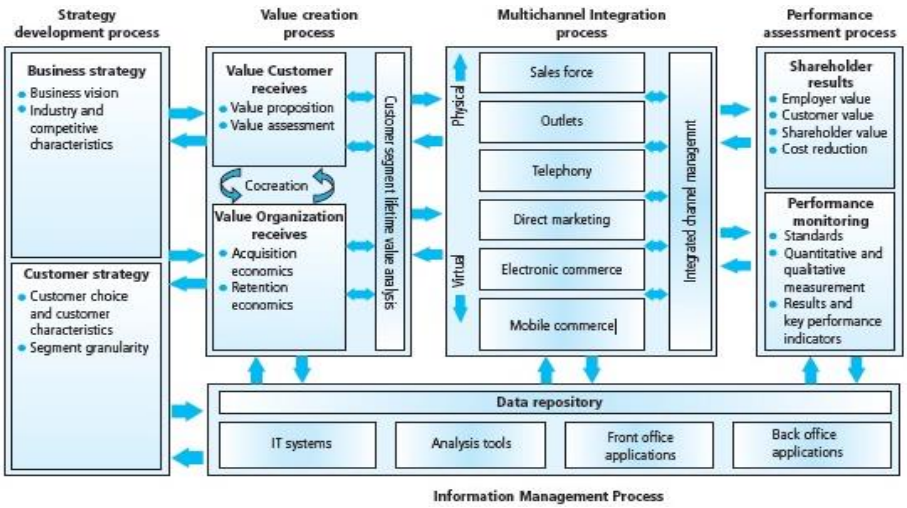
1. ábra: IDIC modell [4]

A QCI (Quality Competitiveness Index) modellt független szakértők alkották, akik a következő stratégiát alkalmazzák. Az ábra középpontjában elhelyezkedő tevékenységek megléte szükséges ahhoz, hogy ügyfeleket szerezzenek, illetve megőrizzék őket. Az erről szóló modellt a 2. ábra szemlélteti: [4]

2. ábra: QCI modell [4]

A Payne féle öt erő modell a következő folyamatok szerint dolgozik:

- Stratégiai fejlesztés: üzleti szempont szerint az ügyfelek megválasztása, kapcsolattartás
- Értékteremtés: olyan érték létrehozása, amely az ügyfelet elégedettséggel tölti el, ugyanakkor a cégnek is hasznot hoz
- Többcsatornás integráció: fizikai és virtuális csatornák kapcsolatától függetlenül egy egységes, megosztott tapasztalat kialakítása
- Teljesítményértékelés megadott szempontok szerint
- Információkezelés: különféle repository-kat, adattárolókat tartalmaz [4]



3. ábra: Stratégiai CRM modell [4]

2.3 Adatbázisok

Az adatbázisok definiálására különféle megoldások léteznek, ezért igyekeztem olyat találni, amely tömör és precíz választ ad. Az alábbi megfogalmazás egyszerűen, de mégis pontosan írja le, mit is jelent valójában.

„Adatbázis: egy olyan integrált adatszerkezet, mely több különböző objektum előfordulási adatait adatmodell szerint szervezetten perzisztens módon tárolja olyan segédinformációkkal, ún. metaadatokkal együtt, melyek a hatékonyság, integritásőrzés, adatvédelem biztosítását szolgálják. Az adatbázis szó rövidítésére gyakran használják az angol rövidítését, a DB-t." [5]

Maga az adatbázis képes az adatok tárolására, ám egymagában csak egy passzív szerkezetet jelent. Mivel az adatokat a későbbiekben szeretnénk manipulálni, többek közt lekérdezni, új elemeket felvinni, módosítani vagy törölni, szükségünk lesz egy adatbáziskezelő rendszerre. Ezen szoftver feladata, hogy biztosítsa az adatbázishoz való hozzáférést, valamint annak karbantartását. [5]

2.3.1 RDBMS vs. NoSQL

Gyakori kérdésként merül fel, hogy miért van szükség relációs és NoSQL adatbázis-kezelő rendszerekre. Miért ne lenne elég, ha csak az egyik létezne? A választ a kérdésre az alábbi bekezdésekben fejtem ki.

Kezdetben az RDBMS modell dominálta a világot, azonban később változtak az igények és a NoSQL is kezdett elterjedni az iparban. A két különböző modellnek nem szándéka kiszorítani egymást, csupán arról van szó, hogy bizonyos alkalmazások más alternatívát kívánnak, míg megint mások tökéletesen megvannak a relációs adatbázis-kezeléssel.

A nem-relációs adatbázisok egyszerű felépítéssel és magas skálázhatósággal bírnak, azonban az adatok integritása és biztonsága adatbázis szinten aligha támogatott. Egyik legfőbb tulajdonságuk, hogy strukturálatlan adatok kezelésére is alkalmas, mint pl.: dokumentumok, hangfájlok, videók tárolása. A rendszer feltelepítése és karbantartása könnyebb feladat az egyszerű felépítés miatt, ellentétben a relációs adatbázissal, ahol a tárolóeszközök beszerzése és a szerverek bővítése drága. Ezekből látható, hogy a nem-relációs adatbázisok ereje a tömeges adattárolásban, a skálázhatóságban, valamint a költséghatékonyságban rejlik.

Egy kicsit vizsgáljuk meg a másik oldalt is, hisz nem kizárólag pozitívumok akadnak a NoSQL oldalán, ráadásul az RDBMS-nek is vannak számtalan előnyei. Azontúl, hogy a relációs technika sokkal kiforrottabb és időben régebbre nyúlik vissza, ezen rendszereknek a támogatottsága is sokkal kedvezőbb, mint a nem-relációs adatbázisoké. Üzleti intelligencia terén megfontolandó, hogy melyik modellt

alkalmazzuk. Az elemzések és riportok készítésekor fontos szempont lehet, hogy az adatok minél inkább részletezhetőek legyenek, az ACID (Atomicity, Consistency, Isolation, Durability) elveknek megfeleljenek. A nem-relációs adatbázis-kezelők létrejöttének egyik fő oka a web és webalkalmazások, ebből kifolyólag a hasonló rendszerek funkcionalitása is az interneten küldött adatokra fókuszál, ami miatt az adatbázist igénybe vevő alkalmazás kevesebb analitikus funkcióval rendelkezik majd.

A fentiek alapján egy valami biztos: ha bármelyik modellt üzleti céllal implementálni szeretnénk, ahhoz előtte figyelembe kell vennünk az elhangzott előnyöket és hátrányokat. [6]

2.3.2 OLTP

Az Online Transaction Processing rendszerek fő feladata a tranzakciók kezelése. Alapvető műveletek tartoznak ide, mint például adott rekord beillesztése, módosítása vagy törlése. A lekérdezések többnyire rövidek és egyszerűek, emiatt kevésbé időigényesek, ami által az adat kevesebb helyet foglal. Ebben a rendszerben az adatbázis szinte folyamatosan frissül, melynek következménye lehet, hogy olykor hiba keletkezhet a tranzakciókezeléskor.

Különös figyelmet kell fordítani az adatok integritására, hiszen az előbb említett hiba befolyásolhatja azt. Ennek érdekében az adatbázis táblái normalizált formában (3NF) kell legyenek. OLTP rendszerre példa egy bankautomata, amely rövid tranzakciókat használ, hogy módosítsa bankszámlánk állapotát. A következő bekezdésben bemutatni kívánt OLAP rendszer forrásául szolgálhat egy ilyen tranzakciókezelő rendszer. [7]

2.3.3 OLAP

Az Online Analytical Processing adatbázis historikus, visszakövethető adatokat tárol. Az efféle rendszereket adatelemzésre használják, amely egy cég esetében a stratégiai döntések meghozatalában juttathat minket előnyös helyzetbe. A felhasználónak lehetősége van komplex lekérdezések generálására multidimenzionális adatokon. Ezek a tranzakciók jelentősen hosszabbak és ritkábban sülnek el, mint az OLTP esetében, továbbá több időt vesznek igénybe. Bonyolultságuk és terjedelmük miatt nagyobb helyre van szükségük. Ezesetben nem szempont, hogy az adatbázis táblái normalizált formában legyenek, ugyanis azok denormalizált alakban hatékonyabbak lehetnek.

Az OLAP rendszereket riportok elkészítéséhez és elemzésekre használják, mint például egy pénzügyi, vagy értékesítési jelentés létrehozására, amelyekhez akár több évre visszamenő adatokra van szükség. [7]

A két alkalmazás különbségeinek áttekintésére szolgál az alábbi ábra:

	OLTP	OLAP
Function	Day to day operation	Decision support
Database Design	Application oriented	Subject oriented
Data	Current, up-to-date detailed, flat relational, isolated	Historical, summarized multi-dimensional, consolidated
Usage	Repetitive	Ad-hoc
Access	Read/Write	Lots of scans
Unit of Work	Short, simple transaction	Complex query
Database Size	Gigabytes	Terabytes
Metric	Transaction throughput	Query throughput, response

4. ábra: OLTP és OLAP rendszerek összehasonlítása [8]

2.4 Üzleti igények megismerése

Ebben az alfejezetben ismertetem a hazai piac elsőrangú szereplőjét, a Magyar Államkincstár szoftverének főbb jellemzőit, majd a probléma felvetése után levonom a következtetéseket.

2.4.1 Hasonló rendszer

A Magyar Államkincstár egy országos hatáskörű, önállóan működő és gazdálkodó közhatalmi, központi költségvetési szerv. Az államkincstár fő tevékenységei közé tartozik az állami költségvetés végrehajtása, annak pénzforgalmának felügyelete, valamint felelős az állam által vállalt garanciákért, és az általa nyújtott hitelek nyilvántartásáért és kezeléséért. Ez utóbbira kínál megoldást a vállalat WebKincstár nevű informatikai rendszere, amelynek fő feladata az állampapírok nyilvántartása. [9]

Az ügyfélnek bejelentkezés után lehetősége van a számlanyitásra és az értékpapírok böngészésére, vásárlására, eladására. A weboldalon áttekintheti a befektetéseit, részletezheti portfólióját és tájékozódhat a jövőbeli esedékességekről. Az alkalmazás állampapírok nyilvántartása mellett további hasznos funkciókkal is rendelkezik, mint például az üzenetküldés, e-számlakivonat létrehozása, valamint ügyfélszolgálati elérhetőségek. [10]

2.4.2 Probléma felvetése és következtetések

A WebKincstár és mobilos applikációja egyeduralkodónak számít a hazai piacon, hiszen lényegében egy állami finanszírozású rendszerről van szó, emiatt csekély számú versenytárssal rendelkezik. Hasonló rendszereket böngészve általánosságban

elmondható, hogy azok kötött struktúrával rendelkeznek, egyesek még csak nem is felhasználóbarátak. Az ügyfelekkel való kapcsolattartás nehézkes. Alapvető információkat az egyes értékpapírokról mobil alkalmazáson keresztül is lehet olvasni, de további ismeretekhez nem árthat magától a forgalmazótól érdeklődni gyorsan és egyszerűen. Kellene egy alkalmazás, ami összehozza az ügyfelet az állampapír forgalmazójával a gördülékenyebb kommunikáció és a felhasználói igények kielégítése érdekében.

Arra a következtetésre jutottam, hogy az államkincstár rendszerei, mint pl.: a WebKincstár és a MobilKincstár csak az állampapírok nyilvántartására szolgál, viszont az általam fejlesztett rendszerben más értékpapírfajták nyilvántartására is lehetőség van a rendszer kibővítése esetén. A kiterjesztés után az ügyfelek szélesebb körű elérésére nyílik lehetőség mind a belföldi, mind a nemzetközi piacon.

3. SPECIFIKÁCIÓ

Az alábbi fejezetben kifejtem a rendszer követelményeit, valamint ismertetem az egyes feladatokat, amelyeket az alkalmazás ellát.

3.1 Felhasználói követelmények

A rendszerbe bejelentkezést követően a felhasználó rögtön értesülhet a változásokról, az állampapírok értékéről. A saját preferenciái szerint listázhatja az értékpapírokat, melyek alapján eldöntheti, melyik a számára kedvezőbb. Az ügyfél szintén információt kaphat arról, hogy hányan vettek meg egy adott értékpapírt az adott időpontig, ezáltal segítve őt a választásban.

Az alkalmazás tartalmaz egy üzenetküldés funkciót, amellyel az ügyfél könnyedén felveheti a kapcsolatot a cég alkalmazottjával, ezáltal további információt tudhat meg. Böngészheti az állampapírokat többféle szűrő alkalmazásával, akár a forgalmazók, vagy más értékpapír attribútum szerint keresve. Cross-platform révén az alkalmazás könnyedén használható akár mobilról is, tehát a felhasználók kényelmesen, akár utazás közben is használhatják az applikációt.

A vállalat számára fontos, hogy időben értesüljön az állampapírok változásairól, ügyfelek tevékenységeiről, vásárlásairól. A vezetőség a kialakított portfóliók alapján később képes lesz a cég számára kedvező döntéseket hozni a jövőben, segítve ezzel a saját és az ügyfelek helyzetét.

3.1.1 Ügyfél

- Állampapírok lekérdezése és vásárlása
- Megvásárolt értékpapírok lekérdezése és követése
- Forgalmazók és elérhetőségeik megtekintése
- Üzenet küldése a dolgozók és forgalmazók részére
- Portfolió részletezése

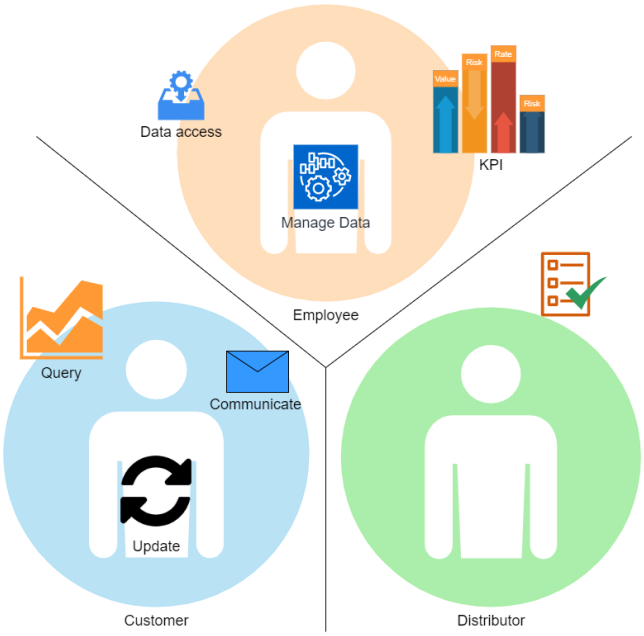
3.1.2 Dolgozó

- Értékpapírok listázása, törlése, módosítása és hozzáadása
- Forgalmazók listázása, törlése, módosítása és hozzáadása
- Tranzakciók és azokkal kapcsolatos információk lekérdezése
- Ügyfelek és azok portfóliójának lekérdezése
- Üzenet küldése az ügyfelek és forgalmazók részére
- Elemzések a tranzakciókról és ügyfelekről

3.1.3 Forgalmazó

- Saját értékpapírok listázása, törlése, módosítása és hozzáadása
- Forgalmazók listázása
- Tranzakciók és azokkal kapcsolatos információk lekérdezése
- Ügyfelek és azok portfóliójának lekérdezése
- Üzenet küldése az ügyfelek és dolgozók részére
- Elemzések a saját tranzakciókról és ügyfelekről

Az 5. ábra bemutatja a három szerepkör feladatait:



5. ábra: Use-case

3.2 Funkcionális követelmények

3.2.1 Állampapírok felvitele, kezelése

Az alkalmazás legfőbb eleme az állampapír, ami köré az egész üzleti logika épül, ezért gondosan megterveztem, hogy milyen adatokra lesz szükségem. Az állampapírok alapján később létrehozható egy adott személy portfóliója. Fontos, hogy az állampapírok és egyéb adatok is nyomon követhetőek legyenek a rendszerben, mivel adott esetben a cég úgy dönthet, hogy visszamenőleg elemezni kívánja azokat.

3.2.2 Ügyfelek felvitele, kezelése

Az ügyfeleket, befektetőket is nyilvántartja a rendszer. Egy ügyfél azonosítása több módon is történhet, ezért a hozzájuk kapcsolódó adatokat, mint például a lakóhely, szintén eltárolásra kerültek, melyek ugyancsak historikus formában vannak nyilvántartva.

3.2.3 Egyéb entitások felvitele, kezelése

Az ügyfelek és az állampapírok közötti kapcsolatokat a tranzakciók adattábla biztosítja. Egy személy több állampapírt is birtokolhat, valamint egy bizonyos kategóriájú állampapírt többen is igénybe vehetnek. A lakóhely vagy bármilyen más információ az ügyféllel kapcsolatban változhat az időben, ezért a táblák rekordjait szintén visszakövethető alakban tárolja az alkalmazás.

3.2.4 Portfólióelemzés

A portfólió készítése az összesített adatokból következik. Az állampapírok vásárlásával létrejött tranzakciókat személyekre lebontva tárolja, amely a későbbi elemzések tárgyaként fog szolgálni.

3.2.5 Riportok készítése

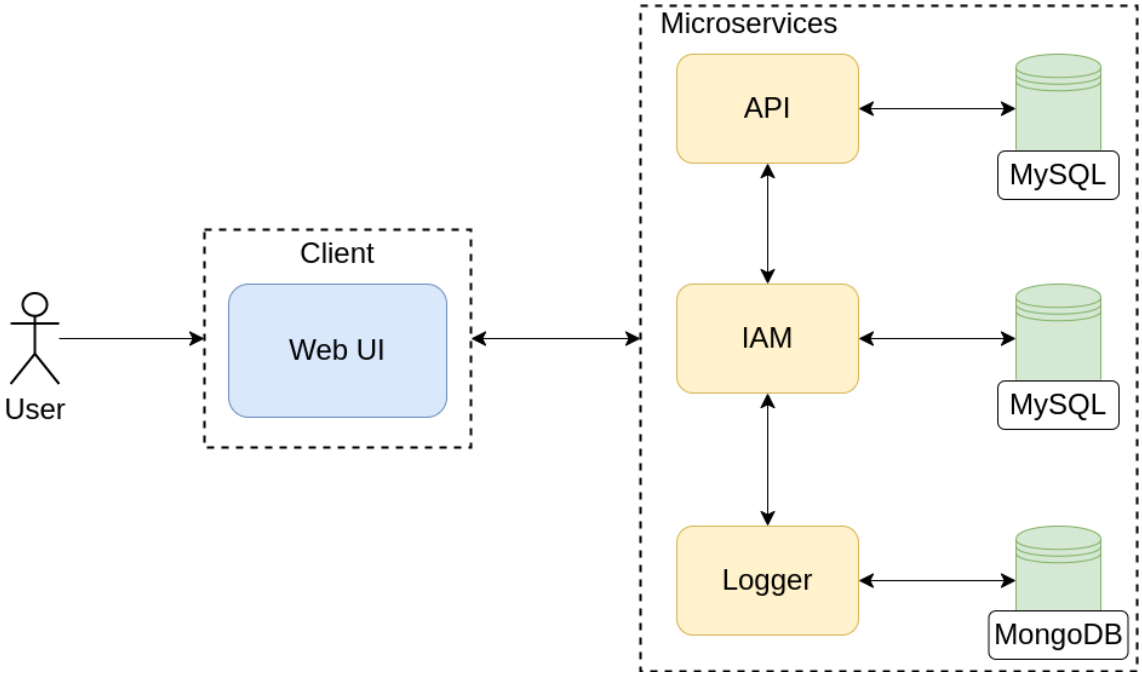
Az eltárolt adatokról jelentéseket lehet készíteni a vállalat számára nyilvánvaló üzleti okokból: például látni szeretné, hogy az egyes forgalmazóknak mekkora a teljesítménye, hol veszik a legtöbb állampapírt. Az egyes portfóliók is analizálási célokat szolgálnak. Granularitás szempontjából vizsgálva az adatokat, azok a legmélyebben részletezhetők. Az állampapírok árfolyama, egyéb tulajdonságaik változása gyakori eset, ezért fontos lehet a vállalat szemszögéből, hogy ezekről akár napi szintű riportokat is készítsen.

3.2.6 Felhasználó- és jogosultságkezelés

Az alkalmazás felhasználóinak, valamint jogosultságainak kezelése kritikus pont a rendszer életében, hiszen egy hasonló rendszernél alapszintű elvárás, hogy illetéktelenek ne férhessenek hozzá bizonyos funkciókhoz.

4. RENDSZERTERV

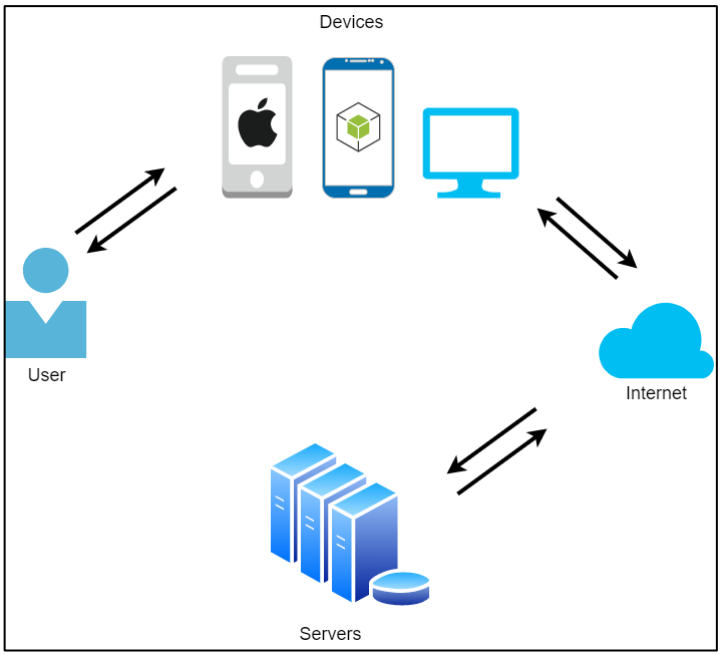
A fejlesztendő rendszer alapvetően kliensből és szerverből áll. A fejezetben ezen komponensek kapcsolatát, valamint az adatbázis táblák szerkezetét írom le. Az összefüggéseket a 6. ábra tartalmazza:



6. ábra: Rendszerterv

4.1 Kliens

A felhasználói felületet egy Angular kliens szolgáltatja, amely kommunikál a backenddel. Ez a webes platform biztosítja a kapcsolatot a felhasználó és az adatok között HTML, CSS és TypeScript segítségével. A cross-platform létrehozásához egy másik keretrendszerre, az Ionic Framework-re is szükségem volt. A kettő kombinálásával ám egy kódot írva, mégis több platform (Android, iOS) is futtatható lesz az alkalmazás. Ezt a kapcsolatot reprezentálja a 7. ábra:



7. ábra: Cross-platform

4.2 Szerver

A rendszer backendjét több service alkotja, valamint az ezekhez kapcsolódó adatbázisok. A szolgáltatásokat Java Spring Framework-kel valósítottam meg.

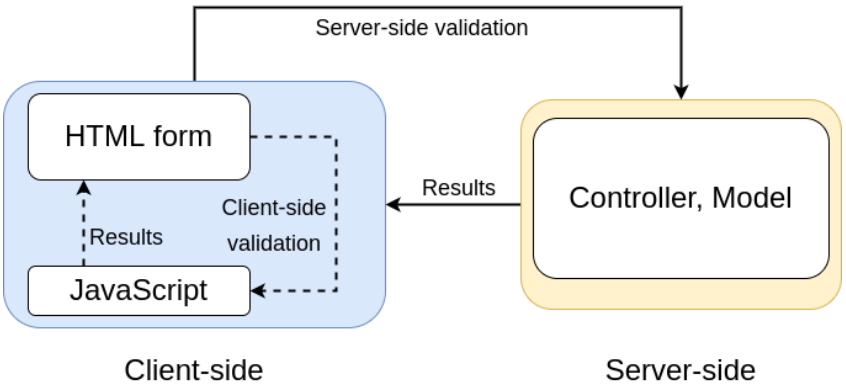
Az információkat szolgáltató REST API magában foglalja a különféle modelleket, az üzleti logikához szükséges szolgáltatásokat, valamint kontrollereket, amelyek az adatokat fogadják a klienstől, vagy éppen továbbítják felé. Az adatbázis számára is kapcsolatot kell biztosítani, ezért az azt leíró interfész is itt található. A modellek vagy entitások ezen service-hez kapcsolódó adatbázisba kerültek be.

Az Identity and Access Management (IAM) keretrendszer lehetővé teszi az alkalmazásba való bejelentkezést, valamint a jogosultságkezelést. Fontos, hogy a rendszer bizonyos részéhez csak a megfelelő jogosultsággal rendelkezők férhessenek hozzá. A szolgáltatás elemeinek eltárolására egy külön adatbázist készítettem.

Bármiféle hiba esetén szükség lehet a hibaüzenetek, vagy csak egyszerűen a hálózaton kimenő, bejövő adatok megjelenítésére. Erre a célra egy ún. Logger service-t hoztam létre.

4.3 Validáció

Az klienstől a szerver felé beérkező adatok olykor hiányosak lehetnek, esetleg nem felelnek meg néhány, általunk támasztott követelménynek. Az efféle paraméterekre validációs szabályokat alkalmaztam a szerver oldalon és a kliens oldalon egyaránt.



8. ábra: Validáció folyamata

4.4 Adatbázisok

Az entitások, valamint a felhasználó- és jogosultságkezeléshez kapcsolódó táblák tárolásához relációs adatbázist választottam, mivel ezen a területen több számításra van szükség. Ez esetben a választásom a MySQL-re esett, mivel ebben több a tapasztalatom, mint más relációs adatbázisok terén. Az üzleti modell központjában az ügyfelek és az állampapírok állnak. Az elemzések, riportok létrehozásához szükséges, hogy az adatok normalizált formában legyenek, így a két legfontosabb elem kapcsolatát különböző perspektívákból nézve tudtam elemezni.

4.4.1 Táblák tartalma

Securities (Állampapíok)

Megnevezés	Leírás	Típus
Security ID (elsődleges kulcs)	azonosító kód	bigint
Name	az állampapír megnevezése	varchar (128)
Description	az állampapír részletes leírása	text
Exchange rate	állampapír aktuális árfolyama	decimal (8,4)

Currency	pénznem	varchar (3)
Denomination	alapcímlet	decimal (16,2)
Accrued interest	felhalmozott kamat	decimal (4,4)
Interest	kamat mértéke	decimal (4,4)
Fixed interest	fix kamatozású-e	bit
Term	időszak	decimal (16,2)
Expiration	lejárat időpontja	datetime
Frequency	kamatfizetés gyakorisága	decimal (2,0)
Inserted	beszúrás ideje	datetime
Last modified	utolsó módosítás ideje	datetime
Visible	láthatóság	bit

1. táblázat: Állampapírok tábla

Transactions (Tranzakciók)

Megnevezés	Leírás	Típus
Transaction ID (elsődleges kulcs)	a tranzakciót azonosító kód	bigint
Security ID (idegen kulcs a Securities táblára vonatkozóan)	az állampapír azonosítója	bigint
Issuer ID (idegen kulcs az Issuers táblára vonatkozóan)	az forgalmazó azonosítója	bigint
Portfolio ID (idegen kulcs a Portfolios táblára vonatkozóan)	az portfólió azonosítója	bigint
Face value	névérték	decimal (16,2)
Net value	nettó érték	decimal (16,2)
Yield	kalkulált hozam	decimal (16,2)
Reference Yield	referencia hozam	decimal (16,2)
Inserted	tranzakció időpontja	datetime

Last modified	utolsó módosítás ideje	datetime
Visible	láthatóság	bit

2. táblázat: Tranzakciók tábla

Opening hours (Nyitvatartások)

Megnevezés	Leírás	Típus
Opening hours ID (elsődleges kulcs)	a nyitvatartást azonosító kód	bigint
Weekdays	hétköznapi nyitvatartás	varchar (16)
Saturday	szombati nyitvatartás	varchar (16)
Sunday	vasárnapi nyitvatartás	varchar (16)
Inserted	beszúrás ideje	datetime
Last modified	utolsó módosítás ideje	datetime
Visible	láthatóság	bit

3. táblázat: Nyitvatartások tábla

Issuers (Forgalmazók)

Megnevezés	Leírás	Típus
Issuer ID (elsődleges kulcs)	a forgalmazót azonosító kód	bigint
Address ID (idegen kulcs az Addresses táblára vonatkozóan)	az forgalmazó címének azonosítója	bigint
Opening hours ID (idegen kulcs az Opening hours táblára vonatkozóan)	nyitvatartás azonosítója	bigint
Name	a forgalmazó neve	varchar (128)
Email	a forgalmazó elektronikus levelezési címe	varchar (64)
Phone	a forgalmazó telefonszáma	varchar (16)

Inserted	beszúrás ideje	datetime
Last modified	utolsó módosítás ideje	datetime
Visible	láthatóság	bit

4. táblázat: Forgalmazók tábla

Customers (Ügyfelek)

Megnevezés	Leírás	Típus
Customer ID (elsődleges kulcs)	az ügyfelet azonosító kód	bigint
Address ID (idegen kulcs az Addresses táblára vonatkozóan)	az ügyfél címének azonosítója	bigint
First Name	az ügyfél keresztneve	varchar (128)
Last Name	az ügyfél vezetékneve	varchar (128)
Email	az ügyfél elektronikus levelezési címe	varchar (64)
Phone	az ügyfél telefonszáma	varchar (16)
Date Of Birth	az ügyfél születési dátuma	date
ID Card	az ügyfél személyét igazoló dokumentum azonosítója	varchar (16)
Gender	ügyfél neme	bit
Inserted	beszúrás ideje	datetime
Last modified	utolsó módosítás ideje	datetime
Visible	láthatóság	bit

5. táblázat: Ügyfelek tábla

Employees (Dolgozók)

Megnevezés	Leírás	Típus
Employee ID (elsődleges kulcs)	az dolgozót azonosító kód	bigint

First Name	az dolgozó keresztneve	varchar (128)
Last Name	az dolgozó vezetékneve	varchar (128)
Email	az dolgozó elektronikus levelezőcíme	varchar (64)
Phone	az dolgozó telefonszáma	varchar (16)
Inserted	beszúrás ideje	datetime
Last modified	utolsó módosítás ideje	datetime
Visible	láthatóság	bit

6. táblázat: Dolgozók tábla

Messages (Üzenetek)

Megnevezés	Leírás	Típus
Message ID (elsődleges kulcs)	az üzenetet azonosító kód	bigint
Employee ID (idegen kulcs az Employees táblára vonatkozóan)	a dolgozót azonosító kód	bigint
Customer ID (idegen kulcs a Customers táblára vonatkozóan)	az ügyfelet azonosító kód	bigint
Content	az üzenet szövege	text
To Customer	a küldő azonosítása ez alapján történik	bit
Inserted	az üzenet időpontja	datetime
Last modified	utolsó módosítás ideje	datetime
Visible	láthatóság	bit

7. táblázat: Üzenetek tábla

Countries (Országok)

Megnevezés	Leírás	Típus
Country ID (elsődleges kulcs)	az országot azonosító kód	bigint
Country code	országkód	varchar (2)
Name	az ország neve	varchar (128)
Inserted	beszúrás ideje	datetime
Last modified	utolsó módosítás ideje	datetime
Visible	láthatóság	bit

8. táblázat: Országok tábla

Cities (Városok)

Megnevezés	Leírás	Típus
City ID (elsődleges kulcs)	a várost azonosító kód	bigint
Country ID (idegen kulcs a Countries táblára vonatkozóan)	az országot azonosító kód	bigint
Postal code	a város irányítószáma	varchar (16)
Name	a város neve	varchar (128)
State	az állam/megye neve, ahol a város található	varchar (128)
Inserted	beszúrás ideje	datetime
Last modified	utolsó módosítás ideje	datetime
Visible	láthatóság	bit

9. táblázat: Városok tábla

Addresses (Címek)

Megnevezés	Leírás	Típus
Address ID (elsődleges kulcs)	a címet azonosító kód	bigint
City ID (idegen kulcs a Cities táblára vonatkozóan)	a várost azonosító kód	bigint
Address line 1	cím 1	varchar (128)
Address line 2	cím 2	varchar (128)
Inserted	beszúrás ideje	datetime
Last modified	utolsó módosítás ideje	datetime
Visible	láthatóság	bit

10. táblázat: Címek tábla

Feedbacks (Visszajelzések)

Megnevezés	Leírás	Típus
Feedback ID (elsődleges kulcs)	a visszajelzést azonosító kód	bigint
Customer ID (idegen kulcs a Customers táblára vonatkozóan)	az ügyfelet azonosító kód	bigint
Rate	a visszajelzés értékelése	int
Text	a visszajelzés szövege	text
Inserted	beszúrás ideje	datetime
Last modified	utolsó módosítás ideje	datetime
Visible	láthatóság	bit

11. táblázat: Visszajelzések tábla

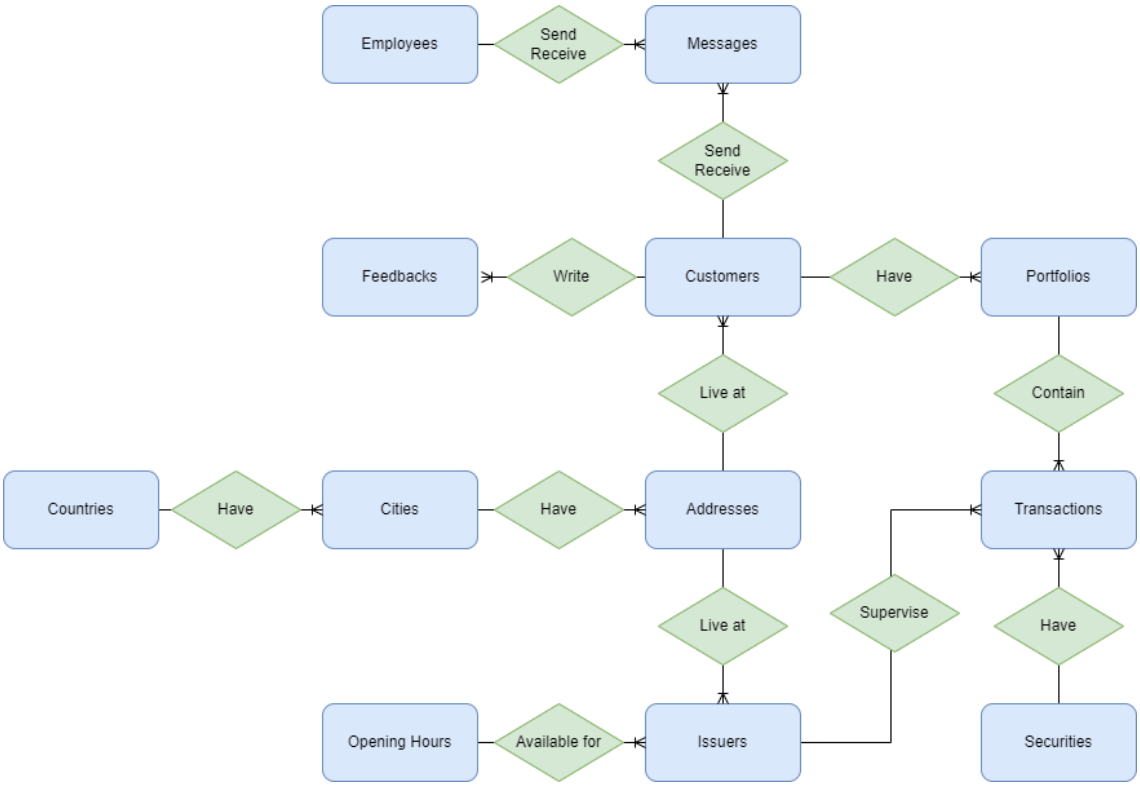
Portfolios (Portfóliók)

Megnevezés	Leírás	Típus
Portfolio ID (elsődleges kulcs)	a portfóliót azonosító kód	bigint
Customer ID (idegen kulcs a Customers táblára vonatkozóan)	az ügyfelet azonosító kód	bigint
Balance	portfólió egyenleg	decimal(16,2)
Inserted	beszúrás ideje	datetime
Last modified	utolsó módosítás ideje	datetime
Visible	láthatóság	bit

12. táblázat: Portfóliók tábla

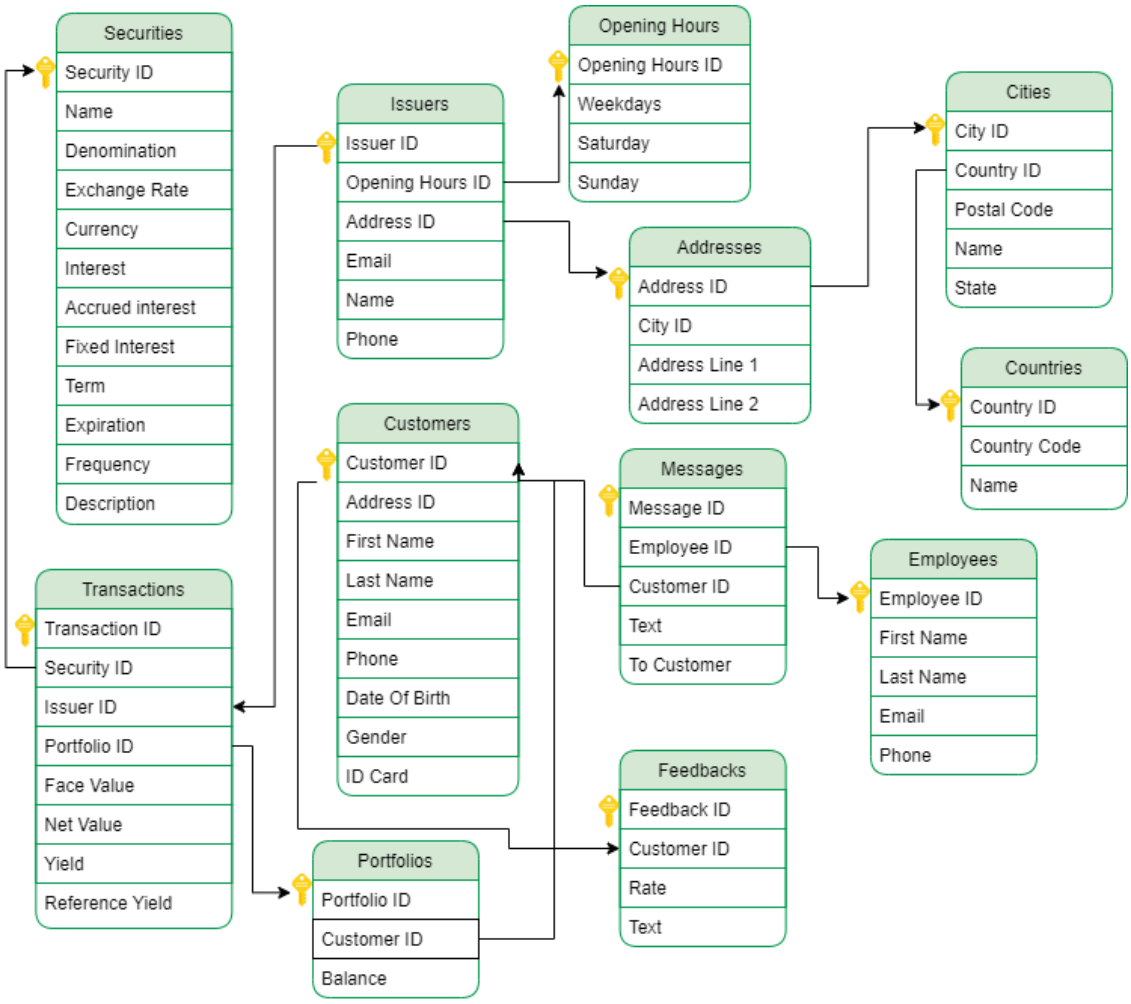
4.4.2 ER és kapcsolati diagram

Az egyed-kapcsolat diagramon jól láthatók az egyes táblák közti összefüggések. A táblák közti kapcsolatok többnyire egy-a-többhöz típusúak.



9. ábra: ER diagram

Az adatbázis táblákat, mezőiket és kapcsolataikat az alábbi ábra szemlélteti:



10. ábra: Táblák és kapcsolataik

5. MEGVALÓSÍTÁS

5.1 Fejlesztői környezet

5.1.1 Docker

Ezen technológia lehetővé teszi a fejlesztők számára, hogy konténereket hozzanak létre, vagy már meglévők használatával szoftvereket futtassanak. A projekt maga nyílt forráskódú, amelyet a közösség folyamatosan fejleszt, így szélesebb felhasználói körökhöz jut el.

A szoftvert azért választottam, mert rugalmas vele a fejlesztés, valamint automatizált az alkalmazások telepítése. Ahogyan egy projekt egyre csak nő, úgy annál fontosabb, hogy a különféle modulokat egy helyen, könnyedén tudjuk kezelni. A technológia a Linux kernelt használja, hogy az egyes folyamatokat egymástól függetlenül tudja futtatni, amely a biztonság szempontjából hatékony. Sokszor előfordulhat, hogy az alkalmazás egyes részeit javítani szeretnénk, vagy egy szolgáltatás eltávolítására kényszerülünk. Az efféle helyzeteket a Docker konténeres megoldásai könnyedén kezelik a modularitásból kifolyólag.

A szoftver rétegeket használ, melyekből egy képet (image) hoz létre. Rétegek akkor keletkeznek, ha a képben valamiféle módosítás (futtatás, másolás) történt. A Docker az említett rétegeket újra felhasználja később a konténerekhez, így képes megosztani az egyes változtatásokat, növelve ezzel a sebességet és a hatékonyságot. A rétegezés további előnye, hogy létezik egy beépített naplózó, amely rögzíti a változásokat, ezáltal verziókezelhetővé válik a rendszer. [11]

5.1.2 IntelliJ IDEA

Az alkalmazás backend-fejlesztésére, valamint az adatbázisok menedzseléséhez erre a fejlesztői környezetre esett a választásom, hiszen a program ezen része Java-ban íródott és meglehetősen sok tapasztalattal rendelkezem az eszközt illetőleg. Komplex volta miatt a fejlesztést könnyítendő, hogy számos nem Java-specifikus funkció is integrálva van a rendszerben.

Egy ilyen, általam is használt funkció az adatbázis kapcsolatok megteremtése volt. Ezt követően kényelmesen tudtam manipulálni az adatbázist az IDE teremtette eszközökkel. [12]

5.1.3 Visual Studio Code

A frontend fejlesztését egy ingyenes, nyílt forráskódú programmal valósítottam meg. A Microsoft által készített kódszerkesztővel az egyszerű szöveg, a HTML és

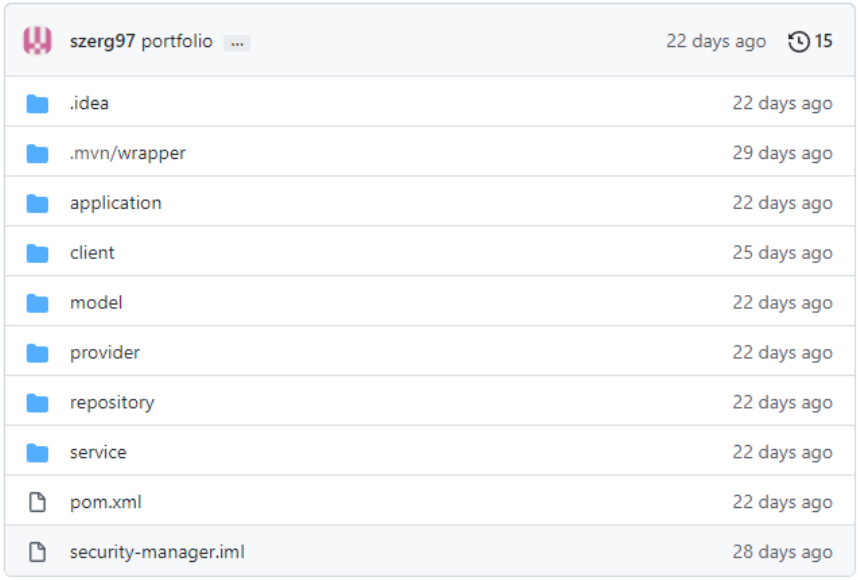
JavaScript kódok kényelmesen kezelhetőek, a különféle bővítmények pedig még jobb felhasználói élményt nyújtanak.

5.1.4 Google Chrome

A folyamatos fejlesztéshez elengedhetetlen, hogy tesztelni tudjuk, amit valójában alkotunk. Az API-t és az applikáció megjelenését egyaránt figyelemmel kísérhetjük, manipulálhatjuk a böngészőben. Webalkalmazás révén ez egyértelműnek tűnik, ugyanakkor egy könnyed lépéssel átválthatunk a mobilos nézetre.

5.1.5 Verziókezelés (GIT)

A verziók követése fontos egy szoftver életében, hisz ezáltal nyomon követhetőek a modulok korábbi fázisai, és egy komolyabb hiba esetén pedig könnyebb a rendszer visszaállítása. A fejlesztés alatt a backend és frontend ugyanazon repository alá tartoznak.



11. ábra: GIT repository

5.2 Backend

Az alkalmazás REST API moduljának alapját a Spring Boot képezi. A megvalósításához szükségesek további könyvtárak, függőségek, melyek egy részét a projekt létrehozásakor köttöttem be.

Egy program sok helyütt tartalmaz olyan kódot, amely elengedhetetlen az egyes osztályok felépítésében (pl.: getterek, setterek, konstruktor). Az efféle kódrészletek elrejtésében segít a Lombok nevezetű könyvtár.

A Spring Web egy MVC komponenst kínál, amely Apache Tomcat beépített konténert használ a webszerverhez.

Az alkalmazás és az adatbázis közti kapcsolat fenntartásához, valamint az adatok perzisztenciájához a Java Persistence API és a Hibernate (objektum-relációs leképezést megvalósító könyvtár) együttes működésével ad megoldást a Spring Data JPA. Továbbá, szükség van egy MySQL driverre az adatbázis kapcsolatokhoz.

5.2.1 Konfiguráció

Az alkalmazás a *http://localhost:8080/security-manager* címen érhető el, a backend alapvető konfigurációját az application.properties fájl írja le. Itt találhatjuk továbbá az adatbázisokra vonatkozó beállításokat, hozzáféréshez szükséges adatokat. A felhasználó- és jogosultságkezelést végző keretrendszer (Keycloak) konfigurációjának egy részét szintén tartalmazza a fájl. Maga az Application.java is tartalmaz biztonsági beállításokat az OpenID protokoll számára.

application.properties

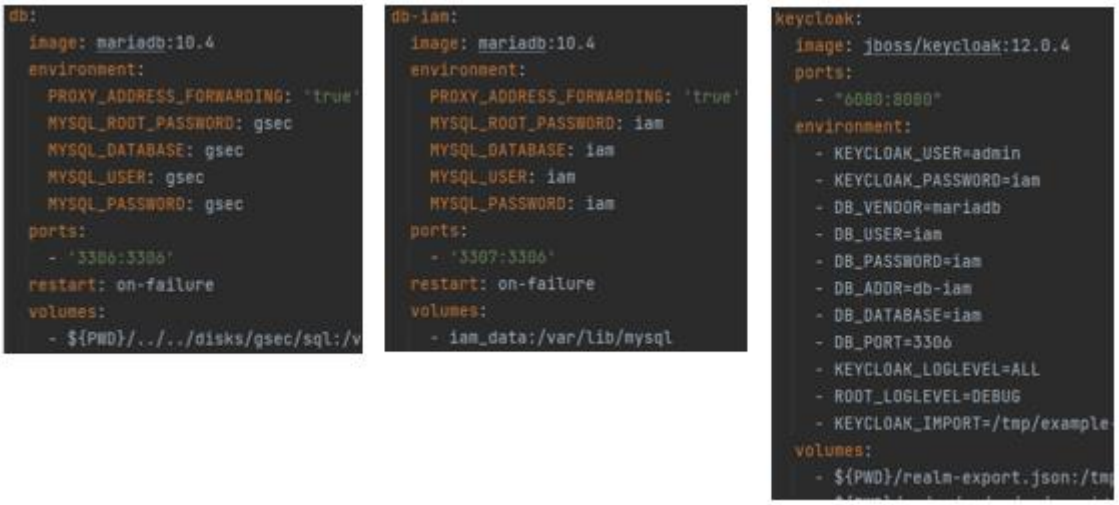
```
#server
server.port=8080
server.servlet.context-path=/security-manager

#swagger
spring.flyway.baseline-on-migrate=true
springdoc.api-docs.path=/api-docs
springdoc.swagger-ui.path=/swagger-ui-custom.html
spring.jackson.serialization.write_dates_as_timestamps=false

#db
spring.datasource.url=jdbc:mysql://localhost:3306/gsec
spring.datasource.username=gsec
spring.datasource.password=gsec
spring.jpa.properties.hibernate.format_sql=true
spring.datasource.driver-class-name=com.mysql.jdbc.Driver
spring.jpa.hibernate.ddl-auto=none
```

12. ábra: Konfiguráció 1.

Az alkalmazás több Docker image-t is használ, ezeknek szintén szükségük van különféle beállításokra, amelyeket egy docker-compose.yaml nevű fájl tartalmaz. Az adatbázisok és a jogosultságkezelő felparaméterezése is itt történik:



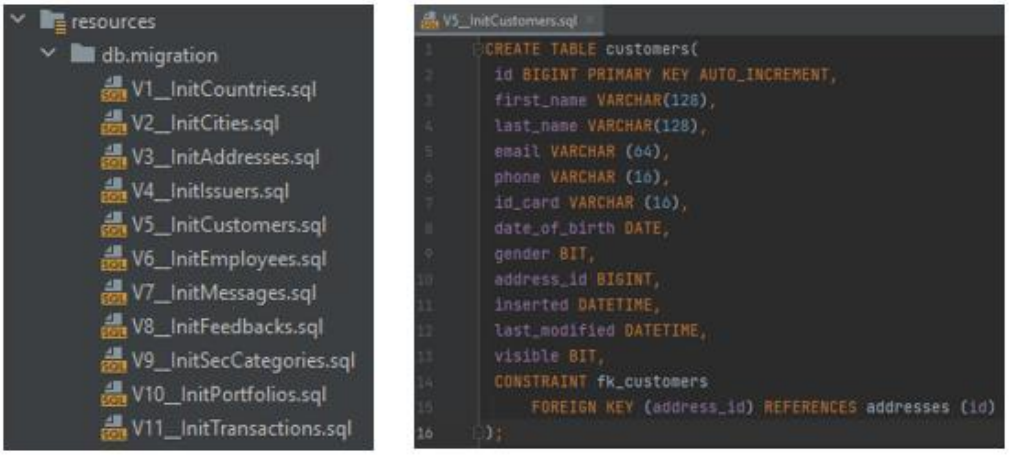
13. ábra: Konfiguráció 2.

A bal oldali képen az üzleti entitásokat tároló adatbázis (gsec) konfigurációja látható azonos felhasználónévvel és jelszóval. Hasonlóképpen néznek ki a felhasználókat kezelő DB beállításai a középső képen. Jobb oldalon az utóbbi adattárolóra támaszkodó szolgáltatás, a Keycloak egyes paraméterei figyelhetők meg.

5.2.2 Adatbázisok

Az alkalmazás számára az adatokat két külön adatbázis szolgáltatja, mindkettő SQL alapú és Docker segítségével operálnak, amelyek konfigurációját az előző fejezetben említett YAML fájl tárolja. A virtualizációs program ennek segítségével tölti le, majd futtatja a megfelelő programot vagy keretrendszert a webről.

Az üzleti adatokért, entitások tárolásáért felelős MariaDB létrehozói scriptjei külön-külön fájlokban találhatók. Az alkalmazás kódjának verziókövetését számos modern eszközzel elvégezhetjük, folyamatos integrációval (CI) és megfelelő deployment folyamatokkal pedig naprakészen tarthatjuk azt. Az adatbázisok verziókövetésére egy Flyway nevezetű adatbázis-migrációs eszközt használtam, amely figyelemmel kíséri azok állapotát és lehetőséget ad arra, hogy alkalom adtán migrálni tudjam a DB jelenlegi verzióját egy következőre.



14. ábra. SQL létrehozói szkriptek

A másik, szintén MySQL alapú adattároló a felhasználó- és jogosultságkezeléshez szükséges adatokat tárolja. Ezen DB adatait egy Keycloak nevű szoftver szolgáltatja, amely lehetővé teszi az egyszeri bejelentkezést (Single Sign-On) az Identity and Access Management alkalmazással.

Hogy az imént említett két típusú adat külön legyen egymástól kezelve azért is fontos, hogy bármilyen alrendszer (pl.: entitásokat tartalmazó adatbázis) visszaállítása esetén ne kelljen hosszú időt eltölteni az adatok szétválasztásával, ez megkönnyíti a fejlesztést.

5.2.3 Model

A modul magában foglalja az üzleti logika entitásait. Az összes modell egy őstől származik, ezeknek több közös attribútumai is létezik, ilyen pl.:

- Entitás azonosítója
- Beszúrás ideje
- Utolsó módosítás ideje
- Láthatóság

Ezek az osztályok „mappelve” vannak a gsec adatbázis tábláira, azonos megszorításokkal szerepelnek, valamint a táblák közti kapcsolatok (Many-to-Many, One-to-Many) is megjelennek. Ahhoz, hogy a Hibernate keretrendszer megfelelően felismerje az objektumok közti kapcsolatokat, nem elég csak az egyik oldalról annotálni azt, fel kell tüntetni az összes, a kapcsolat szempontjából releváns osztályban.

15. ábra: JPA entitás kapcsolatok

Egy modell mezőin több annotáció is van, ezek közül több a validációért felelős. A specifikáció során leírt adatok típusát, valamint méreteit a kódban itt határoztam meg:

```java
@Entity
@Table(name = "issuers")
@Getter
@Setter
@NoArgsConstructor
public class Issuer extends GSecEntity {

    @Schema(description = "Name of issuer")
    @NotBlank(message = "error.issuer.name.not-blank")
    @Size(min = 2, max = 128, message = "error.issuer.name.size")
    private String name;
    @Schema(description = "Email of issuer")
    @NotBlank(message = "error.issuer.email.not-blank")
    @Size(min = 5, max = 64, message = "error.issuer.email.size")
    private String email;
    @Schema(description = "Phone number of issuer")
    @NotBlank(message = "error.issuer.phone.not-blank")
    @Size(min = 2, max = 16, message = "error.issuer.phone.size")
    private String phone;
```

16. ábra: Modell szintű validáció

5.2.4 Repository

Ezen komponens feladata a perzisztencia kezelés, amely egyfajta hidat képez az alkalmazás és az adatbázis között. Ezen modul interfészei és metódusainak kezelése szintén a JPA feladata. Egyfajta szolgáltatásként működnek az alkalmazásban. Mindegyik modellnek saját repository-ja van. Tipikus funkciói a lekérdezés, hozzáadás, módosítás és törlés, valamint ezek különféle variációi. Szerettem volna kihasználni az entitásokra releváns műveleteket is, ezért kiegészítettem a JPA nyújtotta interfészt.

Az értékpapír nagyon sok információt tartalmaz, de önmagában nem sok mindent lehet vele kezdeni, ezért szükségem van különféle szűrőkre, csoportosításokra, mint pl.:

- Portfoliónkként
- Kategóriánkként
- Forgalmazónkként

TransactionRepository.java

```java
package com.secman.repository;

import ...

@Repository
public interface TransactionRepository extends JpaRepository<Transaction, Long> {
    List<Transaction> findByPortfolio(Portfolio portfolio);
    List<Transaction> findByCategory(SecurityCategory category);
    List<Transaction> findByIssuer(Issuer issuer);
}
```

17. ábra: JPA Repository

5.2.5 Service

Az üzleti logikáért felelős modul a repository modulra támaszkodva hatja végre az egyes feladatokat, adattárolási műveleteket. Az alapvető cél az volt, hogy ezen a helyen közrefogjam az adatbázis műveleteket, valamint olyan addicionális funkciókat, amelyek további számításokhoz, valamint hibakezeléshez szükségesek.

```java
public Transaction getOne(Long id){
    Optional<Transaction> optional = this.transactionRepository.findById(id);
    if (optional.isPresent())
        return optional.get();
    throw new RuntimeException("Could not found");
}
```

18. ábra: Backend service függvény

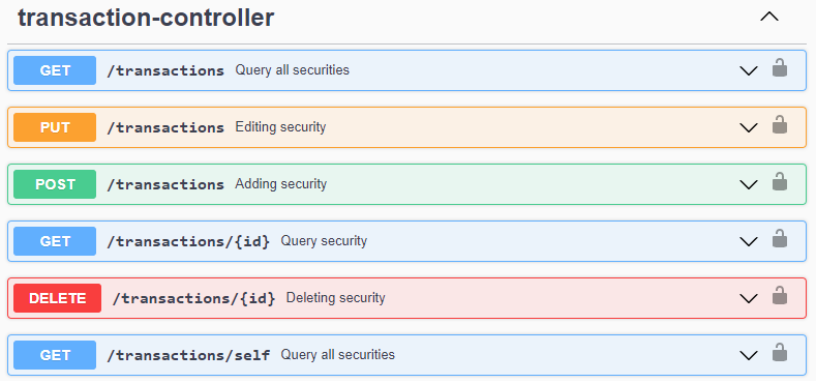
Az adattáblák feltöltéséért szintén ez a modul felelős. Mivel nem volt lehetőségem éles adatokkal dolgozni, így azokat magam generáltam le.

5.2.6 REST Controller

Az MVC pattern klienssel való kommunikációját a kontrollerek végzik, amelyek különféle végpontokon keresztül szolgáltatják az egyes service-ek által adott eredményeket. Ez esetben az eredmények a legtöbbször lekérdezéseket jelentenek,

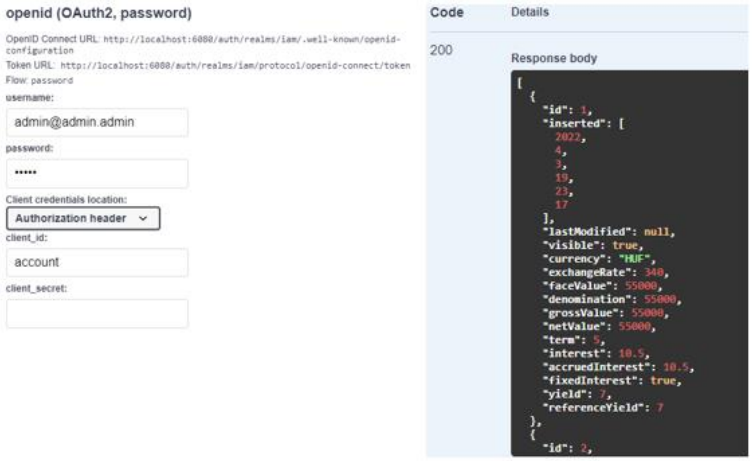
melyeket a webes felhasználói felületen, vagy mobilalkalmazáson keresztül intéz a felhasználó.

A kliens-szerver kapcsolatot felügyelő CORS mechanizmus az erőforrások megosztásában játszik fontos szerepet, ezért ezt engedélyezni szükséges. A fejlesztés erejéig ezt minden kontroller esetében beállítottam, továbbá jeleztem, hogy az adott végpontok milyen útvonalon keresztül érhetőek el. Az egyes végpontok esetében kiemelten fontos, hogy azokat kizárólag arra jogosult felhasználó érhesse el, és használhassa az erőforrást. Ennek érdekében annotáltam az egyes elérési pontokat a megfelelő szerepkörrel. A kliens felől beérkező kérések szintén validálásra kerültek a kontrollerben. Az API-t egy grafikus felületen keresztül is lehet manipulálni ellenőrzés céljából, valóban az elvárt eredményeket szolgáltatja-e az alkalmazás. Ezt a 19. ábra mutatja be:



19. ábra: Swagger UI

Mivel még nem hitelesítettem magam az alkalmazásban, bármilyen kérést intézve az API felé sikertelen választ kaptam. Autorizációt követően a válasz sikeresen megérkezik JSON formátumban, további részleteket is megtekinthetünk a felületen:

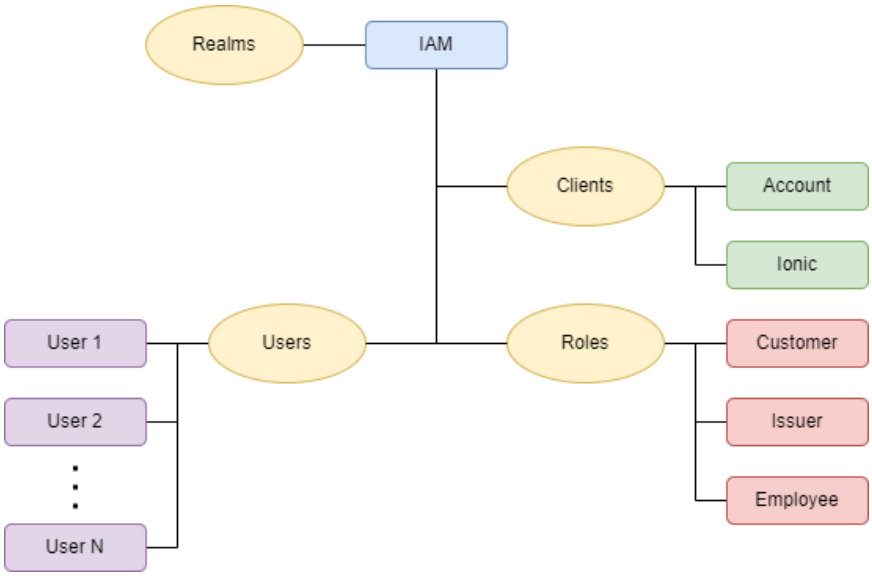


20. ábra: Autorizációt követő sikeres válasz

5.3 Felhasználó- és jogosultságkezelés (IAM)

A Keycloak legfelsőbb szervezeti egysége a Realm. Itt kezelhetjük az alá tartozó objektumokat. Egy ilyen objektum alá tartozhat több kliens, még több szerepkör, és annál is több felhasználó. Egy felhasználónak többféle szerepe is lehet. Az alkalmazásban csak egyetlen egy Realm létezik, iam névvel létrehozva, melyhez tartozik további két kliens: account és ionic.

Az applikáció felhasználói a vállalatnál dolgozó személy, az állampapírt kibocsátó szervezet felhasználója, valamint az állampapírt vásárló ügyfél. Ezen típusokból kiindulva három szerepkört hoztam létre: employee, issuer és customer. A szervezeti felépítést az alábbi ábra szemlélteti:



21. ábra: Keycloak struktúra

5.4 Frontend

Az alkalmazás felhasználói interfészét Ionic Framework-kel fejlesztettem, amely tulajdonképpen egy felokosított Angular.js, ami TypeScript, HTML és CSS kódok kombinációját képezi. Ez lehetővé tette számomra, hogy egyetlen keretrendszerrel cross-platform alkalmazást készítsek, nem kell külön Androidra, vagy iOS-re fejleszteni.

5.4.1 Model

A UI számára a modelleket leíró interfészeket a backend mintájára készítettem el, ezért a kliens és backend közt cserélődő adatok automatikusan konvertálódnak.

22. ábra: Frontend modellek

5.4.2 Service

A szolgáltatások feladata a szerverrel való kapcsolattartás, rajta keresztül hajt végre HTTP kéréseket a kliens. Egy példán keresztül bemutatom egy vevő által vásárolt tranzakciók lekérdezését:

```
fetchSecuritiesByCustomer(){
  return this.http.get<Transaction[]>(`${this.baseUrl}/self`)
  .pipe(
    map(response => {
      const securities: Transaction[] = [];
      for(const data in response){
        securities.push(response[data]);
      }
      console.log(securities);
      return securities;
    }),
    tap(securities => {
      this._securities.next(securities);
    })
  );
}
```

23. ábra: Frontend GET kérés

A kliens egy GET kéréssel fordul az API felé a megadott URL-en keresztül, majd várja a tőle érkező választ. A kapott választ, jellemzően egy JSON formátumban lévő objektumtömböt elraktározom egy listában, majd frissítem az adott listát. A frissítésre

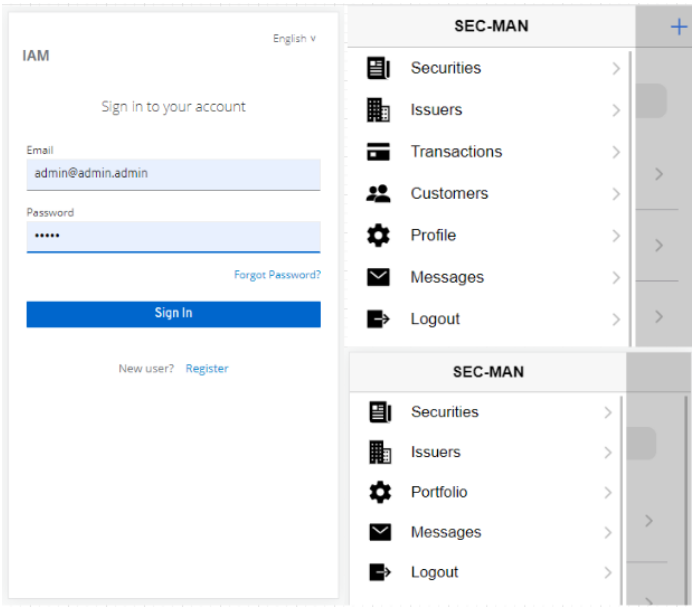
azért van szükség, mert alapesetben a statikus weboldalak nem képesek rá, ezért olyan adatszerkezetet (pl.: Observable) érdemes használni, amellyel a kérésre érkező válaszra fel lehet iratkozni. Később, detektálva a bekövetkező változásokat, értesíti a feliratkozókat, akik ezáltal frissítik az állapotukat.

5.4.3 További komponensek

A frontend alapvető részét képezik az oldalak és a közöttük lévő navigációt biztosító routing modulok. Ezek az alkotó elemek egymáshoz képest hierarchikusan rendezettek, szülő-gyermek kapcsolatokkal definiálva. Egy ilyen komponens tartalmazza a már említett modult, a megjelenítésért felelős HTML és CSS fájlokat, valamint az utóbbi számára az adatokat biztosító JavaScriptet.

5.5 Bemutatás és elemzések

Ebben a fejezetben bemutatom az elkészült szoftver működését az applikáció menüpontjai mentén. A bemutatás alatt összehasonlítom, melyek azok a funkciók, amelyeket a hétköznapi felhasználó, valamint amelyeket az általam megcélzott vállalat dolgozója tud használni. A projekt emellett lehetőséget nyújt a forgalmazó általi használatra is, amelyről a továbbfejlesztési lehetőségek részben írok bővebben.

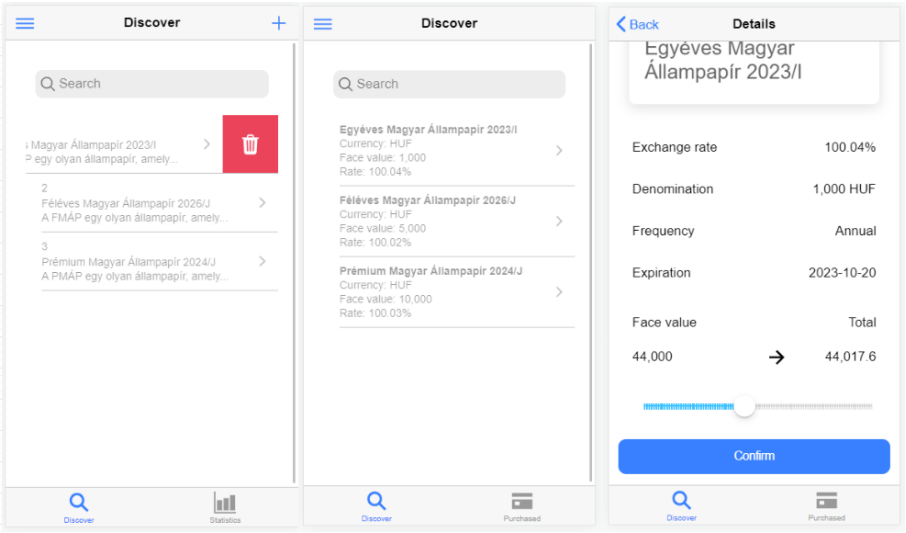


24. ábra: Bejelentkezés és menü

Az alkalmazás indításával a felhasználó a belépési oldalra jut, ahol bejelentkezhet a már meglévő fiókjába, vagy regisztrálhat egy újat. A bejelentkezést követően a menüt előhozva különböző lehetőségek jelennek meg, amik közül választhatunk.

5.5.1 Értékpapírok

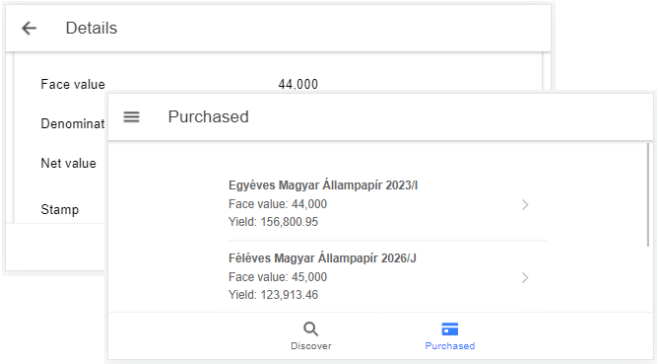
Az ügyfél az első, „Értékpapírok” nevű menüpontban az állampapírokat látja, ahol lehetősége van a keresésre vagy szűrésre egy kiválasztott attribútum alapján. Ezen a ponton a dolgozó képes törölni is már meglévő állampapírt, azonban az ügyfél csak lekérdezni tud.



25. ábra: Értékpapírok modul iOS eszközön

Egy elemet kiválasztva részletesebb képet kap, valamint itt van lehetőség a vásárlásra is. A csúszkát mozgatva a névérték és a kamat összegéből előáll a fizetendő összeg, amely ellenében birtokba veheti az értékpapírt. Mivel a pénzügyi modul nem tárgya a szakdolgozatomnak, a rendszer fizetési modult nem tartalmaz, így a véglegesítéskor nem történik semmilyen valódi, bankszámla terhelő tranzakció.

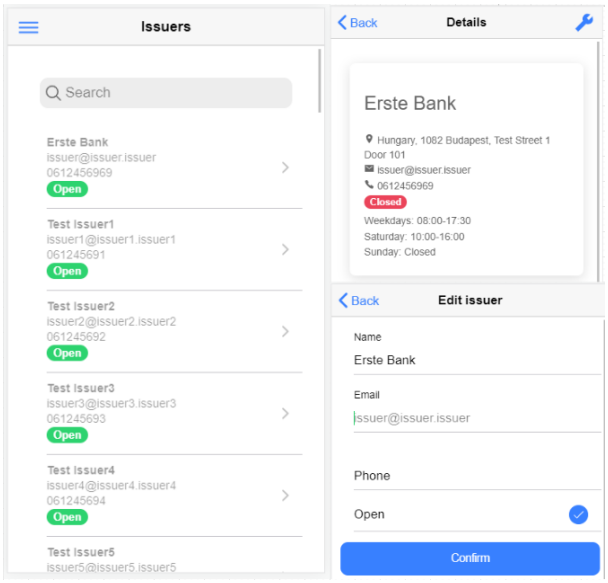
A vásárolt értékpapírok szintén elérhetőek a „Vásárolt” fülön, ahol egyre ráklikkelve láthatóak a vásárlással és a tranzakcióval kapcsolatos általános információk, a kamatok és a várható hozam. Ezt szemlélteti a 26. ábra:



26. ábra: Vásárolt értékpapírok Android eszközön

5.5.2 Forgalmazók

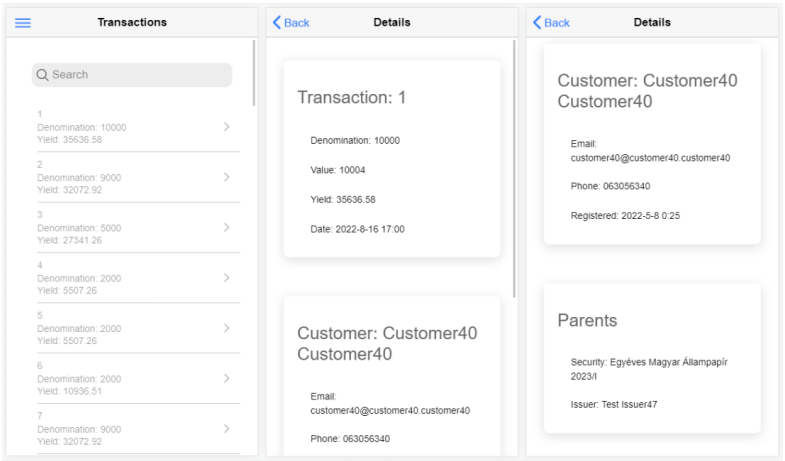
A listában megjelennek az értékpapírok forgalmazói, valamint azoknak alapvető információi: név, e-mail cím, telefonszám és a helyi idő szerint kalkulált nyitvatartás állapota. Az ügyfél, az előző alfejezetben olvasottakhoz hasonlóan csak listázni tud, azonban a cég alkalmazottja számára elérhető az új forgalmazó felvitele, módosítása és törlése is. További részleteket az egyik listaelemre kattintva előugró oldal jelenít meg.



27. ábra: Forgalmazók modul

5.5.3 Tranzakciók

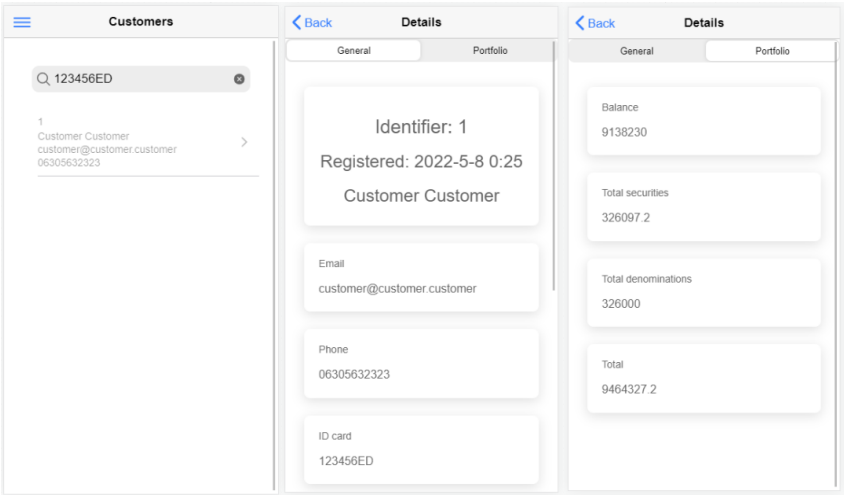
Ez a menüpont kizárólag csak dolgozók részére elérhető. Itt találjuk az összes végbement tranzakciót szűrési lehetőséggel, valamint egy konkrét elemet kiválasztva a hozzá tartozó információkat: ügyfél, értékpapír és kibocsátó.



28. ábra: Tranzakciók modul

5.5.4 Ügyfelek

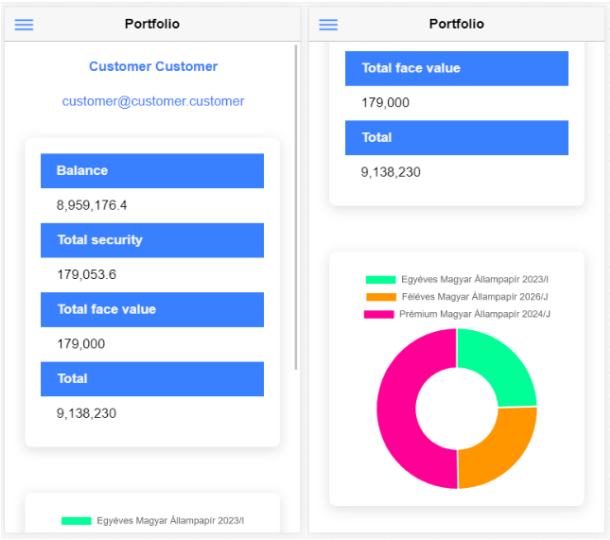
Az alábbi funkciókat szintén csak az arra jogosult személy, azaz a vállalat egy munkatársa vagy az állampapírt forgalmazó intézmény egy munkatársa használhatja. A tranzakciókhoz hasonlóan itt is lekérdezhetőek az egyes rekordok, valamint egy konkrét ügyfélhez tartozó portfólióadatok.



29. ábra: Ügyfelek modul

5.5.5 Portfólió

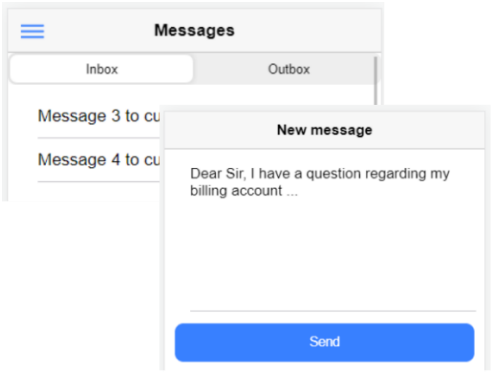
Az egyszerű felhasználónak lehetősége van megtekinteni, hogy az általa vásárolt értékpapírok, valamint az egyenlegén lévő pénzösszeg milyen arányban állnak egymással. A jobb áttekinthetőség érdekében egy fánk-diagrammot használok az adatok vizualizációjára, ezáltal az ügyfél is könnyebben tudja értelmezni azt.



30. ábra: Portfólió részletező

5.5.6 Üzenetek

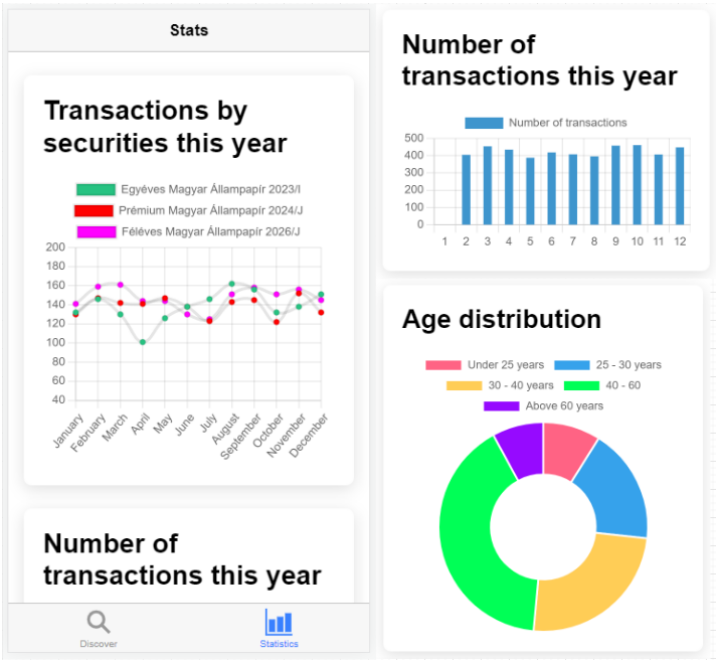
A kommunikáció lebonyolítására egy üzenetküldési funkciót alakítottam ki, amelyen keresztül az ügyfél, valamint a cég alkalmazottja tudnak információt cserélni egymással.



31. ábra: Üzenetek

5.5.7 Elemzések

A vállalat számára a szoftver az ügyfelek portfóliója és az értékpapírok alapján elemzéseket hoz létre. Az általam létrehozott egyik riport a tranzakciók számára adott választ állampapíronként, amely egy vonaldiagram mentén, havi bontásban mutatja az értékeket. Az adott évben sikeres tranzakciókat oszlopdiagrammal valósítottam meg. Az ügyfelek életkorának megoszlását pedig fánk diagram segítségével vizualizáltam.



32. ábra: Elemzések

6. TESZTELÉS

Alkalmazás fejlesztésekor szeretnénk meggyőződni arról, hogy a termékünk megfelelően működik, ehhez a különböző folyamatokat tesztelni kell. A szoftver backendjét unit tesztelésnek vetettem alá. A folyamat során elkülönítettem a valódi és tesztelt adatokat egymástól, ezért az utóbbiakat egy in-memory adatbázisban (H2) tároltam el. Az adattároló komponenst és az azt használó szolgáltatást az ún. Arrange-Act-Assert módszerrel teszteltem, melynek célja, hogy egy keretrendszer segítségével vizsgáljam az adatbázis és a backend közötti folyamatok sikerességét. Az efféle tesztelés kiválóan alkalmas arra, hogy a fejlesztő megbizonyosodjon arról, az általa írt szoftver egységei sikeresen végrehajtják az utasításokat és az elvárt eredményeket adják.

Az adatbázisok táblái a program indításakor automatikusan feltöltődnek az általam generált adatokkal. A tesztelés természetesen túlnyúlik a forráskód manipulálásán, ezért egy felhasználó segítségével, működés közben is teszteltem az alkalmazást ugyanazzal a számítógéppel, amelyen a fejlesztés történt. A gép adatait az alábbi táblázat szemlélteti:

Számítógép	HP EliteBook 8470p
Processzor	Intel(R) Core(TM) i5-3320M CPU @ 2.60GHz
RAM	16,0 GB

13. táblázat: Tesztelői számítógép jellemzői

A tesztelést az 14. táblázatban felsorolt környezetek biztosítják.

	Backend	**Frontend**
Operációs rendszer	Microsoft Windows 10 Education N 64bit	Microsoft Windows 10 Education N 64bit
Keretrendszer	Spring Framework 5.3.20	Ionic Framework 5
Fejlesztőkörnyezet	IntelliJ IDEA Ultimate 21.1	Visual Studio Code 1.67.0
Programnyelv	Java	TypeScript, HTML, CSS
Alkalmazás	Java Spring Boot Web Application	Cross-platform UI
Adatbázis 1 (IAM)	MariaDB 10.4	Nincs
Adatbázis 2 (Üzleti)	MariaDB 10.4	Nincs

14. táblázat: Tesztelői számítógép fejlesztői környezetei

A folyamat során a következő funkciók lettek tesztelve, beleértve a korábban tárgyalt use-case műveleteket.

Dolgozói regisztráció	sikeres
Ügyfél regisztráció	sikeres
Forgalmazói regisztráció	sikeres
Bejelentkezés dolgozóként	sikeres
Bejelentkezés ügyfélként	sikeres
Bejelentkezés forgalmazóként	sikeres
Kijelentkezés	sikeres
Állampapírok lekérdezése / felvitele / módosítása / törlése	sikeres
Állampapír vásárlása	sikeres
Ügyfelek lekérdezése / felvitele / módosítása / törlése	sikeres
Forgalmazók lekérdezése / felvitele / módosítása / törlése	sikeres
Tranzakciók lekérdezése	sikeres
Üzenetek listázása / küldése / fogadása	sikeres

15. táblázat: Tesztek eredményei

A tesztelés továbbá igazolja, hogy a portfoliók és egyéb számított adatok a valóságnak megfelelnek.

7. ELÉRT EREDMÉNYEK

A szakdolgozatom elkészítése alatt lehetőséget kaptam arra, hogy betekinthessek egy, a pénzügyi szektor részére megoldásokat nyújtó vállalat üzleti folyamataiba, ezáltal ihletet merítve a lehetséges üzleti igényekből. Az irodalomkutatás segített abban, hogy a piacon lévő CRM szoftverek kisebb csoportját megismerjem, ezáltal elsajátítva fejlesszek egy olyan rendszert, amely képes az állampapírok igény szerinti tárolására. A különféle GUI technológiák közül egy olyat ismerhettem meg, amellyel rugalmassá vált az alkalmazás fejlesztése. Az Ionic Framework-kel a TypeScript, HTML és CSS ismereteim elegendőnek bizonyult ahhoz, hogy egy cross-platform alkalmazást készíthessek egyetlen keretrendszert használva, így egyetlen forráskódból építettem web- és mobilalkalmazást. A szoftvernek ezen tulajdonsága igencsak kézenfekvő a fejlesztés szempontjából, hiszen a grafikus felhasználói interfészt csak egyszer kell implementálni és onnantól kezdve üzemelni tud sokféle platformon.

Az applikáció elkészítésekor törekedtem annak logikus felépítésére, ezáltal megcélozva azt, hogy a felhasználó kényelmesen tudjon navigálni a menüpontok között és megtalálja az éppen számára szükséges információt. A vállalat szempontjából különös odafigyelést igényelt az elemzendő adatok aggregálása és segítségükkel a pillanatnyi állapot ábrázolása mind az állampapírok alakulását illetőleg, mind pedig az ügyfelek tevékenységeit figyelemmel követve. Az adatok historikus tárolására azért fektettem hangsúlyt, hogy a cég és a forgalmazók számára a múltbéli adatok alapján a szoftver képes legyen különféle elemzéseket generálni, valamint előrejelzéseket adni egy esetleges továbbfejlesztéskor.

A backend tervezésekor először figyelembe kellett vennem, hogy a felhasználókat és szerepköröket tartalmazó adatok, valamint az üzleti folyamatok entitásai külön adatbázisban legyenek, mivel az előbbieket egy erre a feladatra specializálódott keretrendszer kezeli. Az autentikációval kapcsolatos információk kellőképpen el vannak szeparálva más folyamatoktól, így az adatok sértetlensége is biztosítva van.

8. TOVÁBBFEJLESZTÉSI LEHETŐSÉGEK

Mint minden alkalmazás tekintetében, úgy ez esetben is vannak még kihasználásra váró területek. Az általam megcélozott piac bővülése esetén az információáramlást is biztosítani kell egy vállalat részlegei között, valamint ezt a lehető leghatékonyabb eszközökkel kell végigvinni. Egy elemzés exportálása rendkívül hasznosnak bizonyulhat, ha a vállalat a többi modulja felé szeretné eljuttatni azokat. Jelenleg az applikáció állampapír vásárlására kínál lehetőséget, azonban a szolgáltatás-portfóliót később célszerű bővíteni. Hivatalos, a rendszer hitelességét bizonyító dokumentum kiállításának bevezetésében is érezhető a potenciál, így az általam vélt lehetőségek a következők:

- számla készítése
- értékpapírok eladása, jegyzése
- elemzések exportálása, továbbítása
- feladatok modul beépítése a rendszerbe
- az alkalmazás teljeskörű monitorozása
- MI bevezetése
- egyéb értékpapírok (pl.: részvények) nyilvántartása

9. ÖSSZEGZÉS

A téma kiválasztásánál fontos szerepet játszott, hogy egy olyan rendszert fejlesszek, amely az egyetemi évek alatt összegyűjtött tapasztalataimat reprezentálja. Big Data és Üzleti Intelligencia szakirányos hallgatóként egy olyan alkalmazást szerettem volna megvalósítani, amelyben fontos szerepet játszik az adatok sértetlenségének biztosítása, valamint, hogy a szoftver értéket teremtsen a felhasználója számára. Tanulmányaim során az adattárházak világa mellett a webprogramozás fogott meg, ezért is döntöttem úgy, hogy szeretném a két területet összehangolni, hiszen ezek nagyon jól működnek együtt. Webalkalmazás fejlesztésével már korábban is volt dolgom, viszont a mögötte álló adatoknak nem az volt kifejezetten a célja, hogy egy adattárház struktúrájába illeszkedjenek, vagy később elemezhetőek legyenek. Szakdolgozatom elkészítése során úgy éreztem, hogy sikerült egy olyan rendszert alkotnom, amelyben kellően jól strukturált adatok a megfelelő vizualizációval egyértelműbbé teszik a megértést a felhasználó számára.

Tanulmányaim alatt egy informatikai cégnél dolgoztam, ahol korábban a szakmai gyakorlatomat is töltöttem. Ez a Forum Digital Informatikai, Kereskedelmi és Szolgáltató Korlátolt Felelősségű Társaság, ahol részt vettem különböző projektekben, melynek köszönhetően lehetőségem nyílt arra, hogy megismerhessem, hogyan működnek éles helyzetben az üzleti adatok tárolására szánt adatbázisok és a folyamatokat kezelő backend. Az értékpapírok a vállalat tevékenységei kapcsán kerültek komolyabb előtérbe számomra, hiszen az alap ötlet is innen ered. Az általam készített alkalmazás, tulajdonképpen egy kezdeti alternatívának is megfeleltethető.

Az applikáció elkészítésekor törekedtem arra, hogy az általam már jól elsajátított programozási nyelveket (C#, Java, JavaScript) alkalmazni tudjam. A szakdolgozat és az alkalmazás elkészítése remek lehetőséget biztosított arra, hogy új számos olyan területet is megismerhessek, amelyet előtte még nem ismertem.

10. SUMMARY

An important role in selecting the topic of my thesis was to develop a system that would represent my experience gained during my university years. As a student majoring in Big Data and Business Intelligence, I wanted to implement an application in which ensuring the integrity of data and the value of the software to its user play the most important role. Regarding my studies, in addition to the world of data warehousing, web programming was a thing that I became soon familiar with, which is why I decided to coordinate the two areas, as they work really well together. I have been working on web application development before, but the data behind that was not specifically meant to fit into the structure of a data warehouse or to be analyzed later. During the preparation of my thesis, I felt that I had managed to create a system in which sufficiently well-structured data with proper visualization would make the understanding clearer to the user.

During my studies, I worked for a company where I had previously completed my internship. This is the Forum Digital IT, Commerce and Service Limited Liability Company where I have been involved in varoius projects that have given me the opportunity to learn how databases for storing business data operate together with the backend handling management processes in live situations. For me, government securities have come to the forefront connected to the company's activities, as the basic idea comes from here. The application I made can actually be treated as an initial alternative.

During the development of the application, I tried to be able to use the programming languages I have already got familiar with. The preparation of my thesis and the application provided a great opportunity to get to many new areas that I had not met before.

11. IRODALOMJEGYZÉK

[1] Állampapír fogalmak

https://www.allampapir.hu/fogalomtar, utoljára megtekintve: 2022.05.10.

[2] Customer Relationship Management (CRM) Processes from Theory to Practice: The Pre-implementation Plan of CRM System (Khalid Rababah, Haslina Mohd, and Huda Ibrahim)

[3] Vállalatirányítási rendszerek

https://vallalatiranyitasi-rendszer.hu/crm-rendszerek-tipusai/ , utoljára megtekintve: 2022.05.10.

[4] Customer Relationship Management

https://wikimemoires.net/2011/06/the-customer-relationship-management-frameworksmodels, utoljára megtekintve: 2022.05.10.

[5] Adatbázis rendszerek

Dr. Kovács, L.: Adatbázis rendszerek I. (https://people.inf.elte.hu/kiss/DB/ab1-kesz.pdf), utoljára megtekintve: 2022.05.10.

[6] NoSQL vs. RDMBS

https://core.ac.uk/download/pdf/301359204.pdf, utoljára megtekintve: 2022.05.10.

[7] OLTP vs. OLAP

https://techdifferences.com/difference-between-oltp-and-olap.html#KeyDifferences, utoljára megtekintve: 2022.05.10.

[8] OLTP és OLAP összehasonlítás

https://medium.com/t%C3%BCrk-telekom-bulut-teknolojileri/whats-the-difference-between-oltp-and-olap-bdcafdffb1c3, utoljára megtekintve: 2022.05.10.

[9] Magyar Államkincstár tevékenységek

https://www.allamkincstar.gov.hu/hu/a-kincstar/tevekenysegunk, utoljára megtekintve: 2022.05.10.

[10] WebKincstár használati útmutató

https://www.allamkincstar.gov.hu/files/%C3%81LLAMPAP%C3%8DR_FORGALMAZ%C3%81S_20170101-tol/webkincst%C3%A1r/Webkincst%C3%A1r%20Felhaszn%C3%A1l%C3%B3i%20K%C3%A9zik%C3%B6nyv_20210528.pdf, utoljára megtekintve: 2022.05.10.

[11] Docker ismertető

https://www.redhat.com/en/topics/containers/what-is-docker?sc_cid=7013a000002wLw4AAE&gclid=Cj0KCQjw_4-SBhCgARIsAAlegrV8wH_AFZznUFC8lvobdAoaokUIU87N5vAbQbTOn3Ti-QF0TNTU5HIaAmCWEALw_wcB&gclsrc=aw.ds, utoljára megtekintve: 2022.05.10.

[12] IntelliJ ismertető

https://www.jetbrains.com/idea/, utoljára megtekintve: 2022.05.10.

12. MELLÉKLETEK

Az üzleti entitások tárolásáért felelős MariaDB adatbázis (Docker)

```
db:
  image: mariadb:10.4
  environment:
    PROXY_ADDRESS_FORWARDING: 'true'
    MYSQL_ROOT_PASSWORD: gsec
    MYSQL_DATABASE: gsec
    MYSQL_USER: gsec
    MYSQL_PASSWORD: gsec
  ports:
    - '3306:3306'
  restart: on-failure
  volumes:
    - ${PWD}/../../disks/gsec/sql:/var/lib/mysql
```

Az IAM adatok tárolásáért felelős MariaDB adatbázis (Docker)

```
db-iam:
  image: mariadb:10.4
  environment:
    PROXY_ADDRESS_FORWARDING: 'true'
    MYSQL_ROOT_PASSWORD: iam
    MYSQL_DATABASE: iam
    MYSQL_USER: iam
    MYSQL_PASSWORD: iam
  ports:
    - '3307:3306'
  restart: on-failure
  volumes:
    - iam_data:/var/lib/mysql
```

Keycloak környezet futtatásáért felelős kód (Docker)

```
keycloak:
  image: jboss/keycloak:12.0.4
  ports:
    - "6080:8080"
  environment:
    - KEYCLOAK_USER=admin
    - KEYCLOAK_PASSWORD=iam
    - DB_VENDOR=mariadb
    - DB_USER=iam
    - DB_PASSWORD=iam
    - DB_ADDR=db-iam
    - DB_DATABASE=iam
    - DB_PORT=3306
```

```
    - KEYCLOAK_LOGLEVEL=ALL
    - ROOT_LOGLEVEL=DEBUG
    - KEYCLOAK_IMPORT=/tmp/example-realm.json
  volumes:
    - ${PWD}/realm-export.json:/tmp/example-realm.json
    - ${PWD}/../../../../../provider/target/provider-0.0.1-
SNAPSHOT.jar:/opt/jboss/keycloak/standalone/deployments/provider-0.0.1-SNAPSHOT.jar
  depends_on:
    - db-iam
```

Forgalmazók táblát inicializáló SQL script

```
CREATE TABLE issuers(
 id BIGINT PRIMARY KEY AUTO_INCREMENT,
 name VARCHAR (128),
 email VARCHAR (64),
 phone VARCHAR (16),
 status BIT,
 address_id BIGINT,
 opening_hours_id BIGINT,
 inserted DATETIME,
 last_modified DATETIME,
 visible BIT,
 CONSTRAINT fk_issuers_addresses
    FOREIGN KEY (address_id) REFERENCES addresses (id),
  CONSTRAINT fk_issuers_opening_hours
    FOREIGN KEY (opening_hours_id) REFERENCES opening_hours (id)
);
```

Forgalmazókért felelős repository kódja

```
@Repository
public interface IssuerRepository extends JpaRepository<Issuer, Long> {
  List<Issuer> findByAddress(Address address);
}
```

Forgalmazó módosításáért felelős service kódja

```
public Issuer updateOne(Issuer pIssuer){
      Issuer issuer = getOne(pIssuer.getId());
      if (issuer != null) {
         issuer.copyFrom(pIssuer);
         this.issuerRepository.save(issuer);
         return issuer;
      }
      throw new RuntimeException("Could not update");
  }
```

Forgalmazó módosításáért felelős kontroller kódja

```
@ApiResponses(value = {
        @ApiResponse(responseCode = "200", description = "Request successful",
```

```
            content = { @Content(mediaType = "application/json",
                array = @ArraySchema(schema = @Schema(implementation = Issuer.class))) }),
    @ApiResponse(responseCode = "400", description = "Validation error",
            content = { @Content(mediaType = "application/json",
                array = @ArraySchema(schema = @Schema(implementation = Issuer.class))) }),
    @ApiResponse(responseCode = "401", description = "Token expired",
            content = { @Content(mediaType = "application/json")}),
    @ApiResponse(responseCode = "403", description = "You do not have permission",
            content = { @Content(mediaType = "application/json")}),
    @ApiResponse(responseCode = "302", description = "You are not logged in, redirecting",
            content = { @Content(mediaType = "application/json")}),
    @ApiResponse(responseCode = "500", description = "Internal server error",
            content = { @Content(mediaType = "application/json")}),
})
@Operation(summary = "Editing issuer",
    security = {
            @SecurityRequirement(name = "apikey", scopes = {"gsec"}),
            @SecurityRequirement(name = "openid", scopes = {"gsec"}),
            @SecurityRequirement(name = "oauth2", scopes = {"gsec"})
    }
)
@PutMapping("")
public ResponseEntity<Issuer> updateOneIssuer(
    @Parameter (name = "issuer", required = true)
    @RequestBody (required = true) Issuer issuer){
  return ResponseEntity.ok(this.issuerService.updateOne(issuer));
}
```