ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

NEUMANN JÁNOS
INFORMATIKAI KAR

SZAKDOLGOZAT

OE-NIK	Hallgató neve:	**Csoór Péter Frank**
2023	Hallgató törzskönyvi száma:	**T/006724/FI12904/N**

Óbudai Egyetem
Neumann János Informatikai Kar
Kiberfizikai Rendszerek Intézet

SZAKDOLGOZAT FELADATLAP

Hallgató neve: **Csoór Péter Frank**
Törzskönyvi száma: T/006724/FI12904/N
Neptun kódja: [redacted]

A dolgozat címe:

Egységes rendszer kialakítása informatikai cégen belüli csoportok számára
Developing unified system for groups within an IT company

Intézményi konzulens: Dr. Holyinka Péter
Külső konzulens:

Beadási határidő: 2022. december 15.

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

A feladat

Ismerje meg egy cég projektek kezelésével kapcsolatos feladatait, és a jelenlegi kezelési módszerekkel kapcsolatos problémákat. Határozza meg azokat a célokat, melyek egy alkalmazás segítségével segítik a jelenlegi problémák megoldását, és készítse el a követelmény specifikációt. Készítse el a rendszertervet, és vizsgálja meg a rendszer fejlesztéséhez használható eszközöket. Válassza ki a megoldáshoz használandó eszközöket, és fejlessze ki a rendszert.

A dolgozatnak tartalmaznia kell:

- a jelenlegi projektkezelés menetének és eszközeinek tömör ismertetését,
- a fejlesztendő alkalmazás céljait,
- a követelmény specifikációt,
- a fejlesztéshez használandó eszközök tömör ismertetését és a választás indoklását,
- a rendszer terveit, és a tesztelés eredményeit,
- az elért eredményeket és a továbbfejlesztési lehetőségeket.

ÓBUDAI EGYETEM * Neumann János Informatikai Kar * 1. Budapest

P.H.

.....
Dr. Fleiner Rita
intézetigazgató

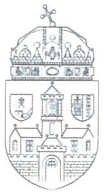
A szakdolgozat elévülésének határideje: **2024. december 15.**
(OE TVSz 55.§ szerint)

A dolgozatot beadásra alkalmasnak tartom:

.....
külső konzulens

.....
intézményi konzulens

Budapest,
2022.12.08.



ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Neumann János Informatikai Kar

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. 12. 14.

.....

hallgató aláírása

9. melléklet

ÓBUDAI EGYETEM

Neumann János Informatikai Kar

KONZULTÁCIÓS NAPLÓ

Hallgató neve: CSÓR PÉTER FRANK

Neptun Kód: [redacted]

Tagozat: LEVELEZŐ

Telefon: [redacted]

Levelezési cím (pl.: lakcím): [redacted]

Szakdolgozat / Diplomamunka[7] címe magyarul:
EGYSÉGES RENDSZER KIALAKÍTÁSA INFORMATIKAI CÉGEN BELÜLI CSOPORTOK SZÁMÁRA

Szakdolgozat / Diplomamunka[8] címe angolul:
DEVELOPING UNIFIED SYSTEM FOR GROUPS WITHIN AN IT COMPANY

Intézményi konzulens: DR. HOLYINKA PÉTER

Külső konzulens: -

Kérjük, hogy az adatokat nyomtatott nagy betűkkel írja!

Alk.	Dátum	Tartalom	Aláírás
1.	2022.02.16.	A DOLGOZAT CÉLJAINAK KIJELÖLÉSE, ÜTEMTERV KÉSZÍTÉSE	
2.	2022.03.07.	VÁZLAT KÉSZÍTÉSE, BEMUTATÁSA	
3.	2022.04.24.	A SZAKDOLGOZAT FORMAI ÉS TARTALMI KÖVETELMÉNYEINEK TANULMÁNYOZÁSA	
4.	2022.05.09	ELKÉSZÍTETT SZAKDOLGOZAT ÁTTEKINTÉSE	

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[9] tantárgy követelményét teljesítette, beszámolóra / védésre[10] bocsátható.

.....

Intézményi konzulens

Budapest, 2022. 05. 12.

---

[7] Megfelelő aláhúzandó!

[8] Megfelelő aláhúzandó!

[9] Megfelelő aláhúzandó!

[10] Megfelelő aláhúzandó!

9. melléklet

ÓBUDAI EGYETEM

Neumann János Informatikai Kar

KONZULTÁCIÓS NAPLÓ

Hallgató neve: CSÓR PÉTER FRANK | Neptun Kód: [redacted] | Tagozat: LEVELEZŐ

Telefon: | Levelezési cím (pl.: lakcím): [redacted]

Szakdolgozat / Diplomamunka[7] címe magyarul:
EGYSÉGES RENDSZER KIALAKÍTÁSA INFORMATIKAI CÉGEN BELÜLI CSOPORTOK SZÁMÁRA

Szakdolgozat / Diplomamunka[8] címe angolul:
DEVELOPING UNIFIED SYSTEM FOR GROUPS WITHIN AN IT COMPANY

Intézményi konzulens: DR. HOLYINKA PÉTER | Külső konzulens: –

Kérjük, hogy az adatokat nyomtatott nagy betűkkel írja!

Alk.	Dátum	Tartalom	Aláírás
1.	2022.10.08.	AZ ELKÉSZÜLT TERV BEMUTATÁSA, JÓVÁHAGYÁSA	
2.	2022.11.16.	A SZAKIRODALOM FELDOLGOZÁSÁRÓL VALÓ EGYEZTETÉS	
3.	2022.11.25.	AZ ELKÉSZÜLT FEJEZETEK FŐBB TARTALMI ELEMEINEK A BEMUTATÁSA	
4.	2022.12.02.	ELKÉSZÍTETT SZAKDOLGOZAT ATTEKINTÉSE	

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[9] tantárgy követelményét teljesítette, beszámolóra / védésre[10] bocsátható.

.....

Intézményi konzulens

Budapest, 2022.12.08.

---

[7] Megfelelő aláhúzandó!

[8] Megfelelő aláhúzandó!

[9] Megfelelő aláhúzandó!

[10] Megfelelő aláhúzandó!

TARTALOMJEGYZÉK

1 Bevezetés 1

2 Megoldás áttekintése 2

2.1 Jelenleg használatban lévő eszközök 2

2.1.1 Kommunikáció 2

2.1.2 Fejlesztés 2

2.1.3 Összegzés 2

2.2 Lehetséges megoldások 2

2.3 Választott megoldás 3

3 Feladatspecifikáció 4

3.1.1 Akceptorok 5

3.1.2 Use Case-ek 5

3.1.3 Adminisztrációs felület 6

3.1.4 Gem-monitorozó eszköz 6

3.1.5 Edge case-ek kezelése 6

4 Felhasznált technológiák 7

4.1 Ruby 7

4.2 Ruby on Rails 7

4.2.1 Működése a gyakorlatban 7

4.3 PostgreSQL 8

5 Tervezés 9

5.1 Adatbázis terv 9

5.1.1 Users tábla 9

5.1.2 Messages tábla 9

5.1.3 Rooms tábla 10

5.1.4 Participants tábla 10

5.1.5 Projects tábla 10

5.1.6 Companies tábla 11
5.1.7 Tasks tábla 11
5.1.8 Statuses tábla 11
5.1.9 ActiveStorage táblái 12
5.1.10 ActiveAdmin táblái 12
5.2 Szoftver terv 13
5.2.1 Felhasználókezelés 13
5.2.2 Jogosultságkezelés 17
5.2.3 Adminisztrációs felület 17
5.2.4 Kommunikáció 18
5.2.5 Projektmenedzsment 19
5.2.6 Feladatmenedzsment 21
5.2.7 Sebezhető gem-ek feltárása 23
6 Megvalósítás 25
6.1 Felhasználókezelés 25
6.2 Jogosultságkezelés 25
6.3 Kommunikáció 26
6.4 Projekt- és feladatmenedzsment 29
6.5 Sebezhető gem-ek feltárása 30
6.6 Adminisztrációs felület 31
6.7 Élesítés 32
7 Tesztelés 33
8 Eredmények 34
9 Fejlesztési lehetőségek 41
10 Összegzés 42
11 Summary 43
12 Irodalomjegyzék 44

13 Mellékletek 46

1 BEVEZETÉS

Szakdolgozatom témájául egy webalkalmazás elkészítését választottam IT cégek számára. A rendszer legfőbb célja, hogy mindazon eszközök, melyek egy SaaS szoftverek fejlesztésével foglalkozó cég használhat a mindennapokban, megtalálhatóak legyenek egy alkalmazáson belül, költségmentesen. Ilyen például egy üzenetküldő alkalmazás, vagy a cég fenségkörébe tartozó projektek és az azokhoz kapcsolódó feladatok teljeskörű menedzselésére szolgáló rendszer. A cég, aki számára az alkalmazást készítem egy olyan eszköz beépítését is kezdeményezte, mely a projektekhez tartozó külső könyvtárak (ún. gem-ek) sebezhetőségeit tárja fel és közvetíti a végfelhasználó felé.

Az alkalmazás elkészítését az indokolta, hogy az előbb felsorolt funkcionalitások eléréséhez több alkalmazást is igénybe kell venni, melyek közül némelyik extra költséggel is jár. Mindemellett a sérült gem-eket csak azok a emberek tudják felderíteni, akik hozzáférnek a kódbázishoz. Ebből kifolyólag egy saját fejlesztésű alkalmazás bizonyult a legjobb megoldásnak.

Szakdolgozatom első részében ismertetni fogom az alkalmazás követelményeinek feltárását, majd ezt követően specifikálom az általam választott technológiákat, majd részletezem a rendszer szoftver- és adatbázisterveit. Végezetül kitérek a megvalósítási folyamatokra és bemutatom az elkészült webalkalmazást felhasználói oldalról

2 MEGOLDÁS ÁTTEKINTÉSE

2.1 Jelenleg használatban lévő eszközök

2.1.1 Kommunikáció

A dolgozók közti kommunikációra jelenleg több eszköz is használatban van. A videóhívásokat internetes felületen folytatják (pl. Google Meet), az egymást között történő üzenetváltásokra pedig a Slack nevű alkalmazást használják. Ezen az alkalmazáson belül lehetőség van privát-, illetve csoportos üzenetváltásra is. Az ilyen csoportokban történik például a projektekhez kapcsolódó feladatok megvitatása.

2.1.2 Fejlesztés

A cég Ruby on Rails technológián alapuló webalkalmazások fejlesztésével foglalkozik, melyek kezeléséhez különböző verziókezelő eszközöket használnak, mint például GitHub, vagy GitLab. Ezen eszközök használatával követni és menedzselni tudják az alkalmazások forráskódjain és a fájlrendszerben végzett módosításokat. Tudomást szerezni a projektek sebezhetőségeiről csak azoknak a felhasználóknak van lehetősége, akik hozzáférnek a kódbázishoz.

2.1.3 Összegzés

A cég nem használ olyan szoftvert, amiben a leírtak mindegyike egy alkalmazáson belül, kompakt módon szerepelne. Kutatómunkám során csak olyan megoldásokat találtam, melyek fizetősek lettek volna. A cég, akinek az alkalmazást fejlesztem egy ingyenesen megvalósítható megoldást keres. Éppen ezért, az alkalmazás elkészítésének fő indoka az volt, hogy megvalósítsak egy olyan rendszert, melyben egyszerre tudnak hatékonyan kommunikálni a fejlesztők, továbbá áttekinthetően kezelni tudják saját alkalmazásaikat, ezzel is gördülékenyebbé téve a fejlesztést.

2.2 Lehetséges megoldások

Az egyik legkézenfekvőbb megoldás az lenne, ha minden eddig használt eszköz és funkcionalitás egy alkalmazáson belül, csokorba szedve megtalálható lenne, és a cég azt

használná a későbbiekben. Rengeteg kész termék van a piacon, ami arra szolgál, hogy projekteket és feladatokat tudjunk menedzselni és kommunikálni tudjunk a dolgozókkal. Viszont ezek többnyire fizetős megoldások és az ügyfélnek vannak olyan extra igényei, amit egy alkalmazás sem biztosít és csak egy külső eszközzel érhetőek el.

2.3 Választott megoldás

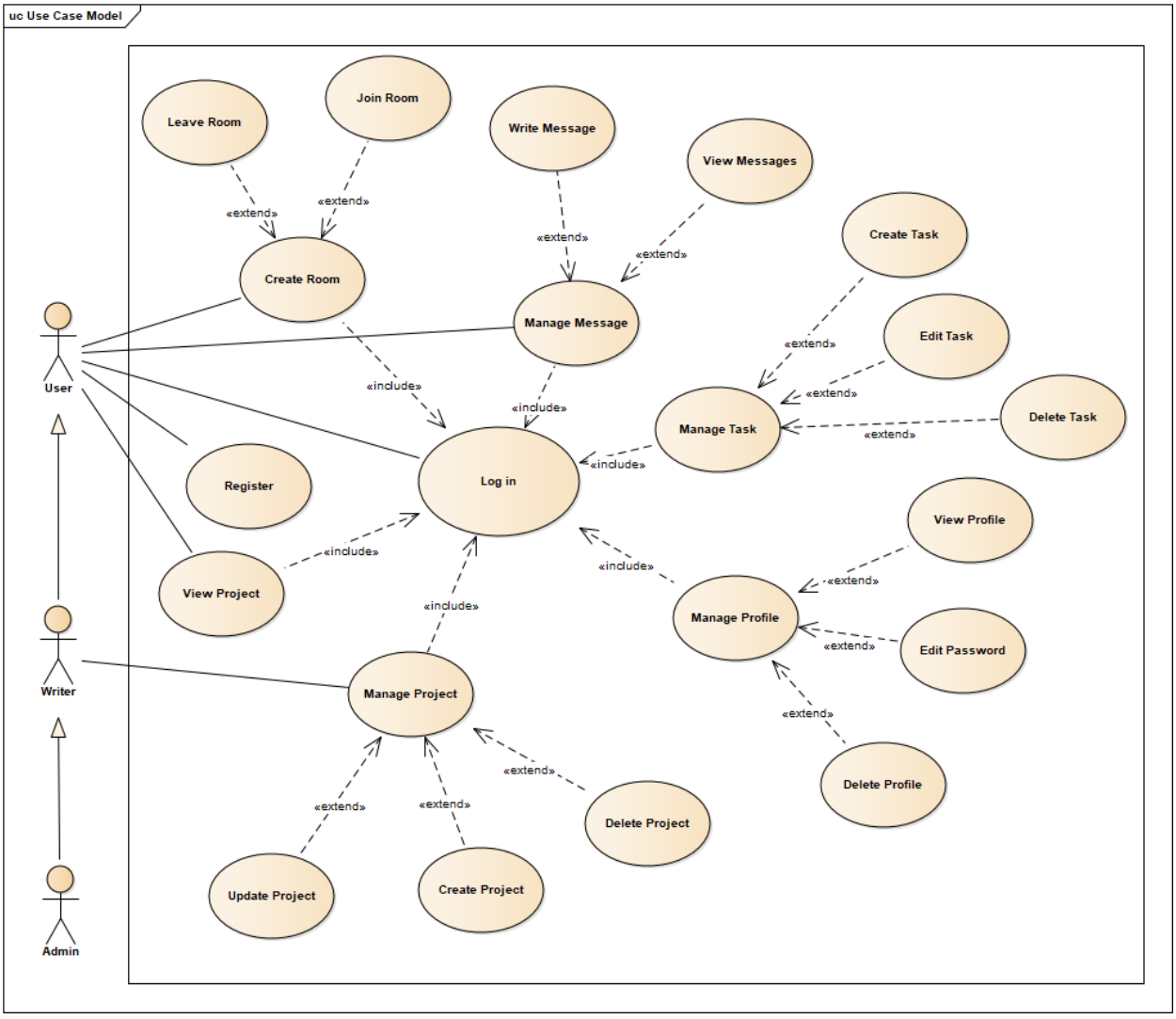
A választott megoldás egy új webalkalmazás fejlesztése lesz az általam választott technológiákkal, ami az igényekre szabva tartalmaz minden olyan eszközt és funkcionalitást, amit korábban a cég használt a mindennapokban. Továbbá, a rugalmasságának köszönhetően egyszerre több cég is használhatja az alkalmazást egymástól teljesen függetlenül.

3 FELADATSPECIFIKÁCIÓ

Cél egy olyan webes felületű alkalmazás elkészítése, mely egy cég fenségkörébe tartozó projektek kezelését és a dolgozók közti kommunikációt kiszolgálja. Továbbá, az ügyfél azt várja el a rendszertől, hogy a projektekhez feladatokat lehessen rendelni. Ezáltal egy tiszta és jól nyomon követhető képet kapnak a dolgozók, hogy kinek milyen elvégzendő feladata van milyen határidővel. Ez nagyban segíteni fogja a fejlesztést.

Miután egyeztettem az ügyféllel a követelményekről és az elvárt funkciókról, elkészítettem számára egy use-case diagramot, mely az alábbi ábrán [1. ábra] látható:

1. ábra
Az alkalmazás használati-eset diagramja



3.1.1 Akceptorok

1. User (alapértelmezett)
    a. Üzenőszobák létrehozása
    b. Saját profil szerkesztése
    c. Task-ok szerkesztése, létrehozása
2. Writer
    a. Minden jogosultság, ami a User-hez tartozik
    b. A saját cégén belüli projektek teljeskörű menedzselése
3. Admin
    a. Minden jogosultság, ami a Writer-hez tartozik
    b. Minden felhasználó teljes körű menedzselése
    c. Felhasználók létrehozása
    d. Projektek menedzselése (minden cég projektjére rálát és tudja is őket menedzselni)
    e. Felhasználók szerepköreinek megszabása

3.1.2 Use Case-ek

1. táblázat
Use-case-ek meghatározása

Register	Regisztrálás az alkalmazásra.
Log in	Bejelentkezés az alkalmazásba.
Create room	Üzenőszobák létrehozása, mely kiegészül szoba elhagyás- és csatlakozás funkciókkal.
Write message	Üzenet írása.
View messages	Üzenetek megtekintése.
Manage projects	Projektek menedzselése (megtekintése, létrehozása, módosítása, törlése).

Manage tasks	Egy projekthez tartozó feladat menedzselése (megtekintése, létrehozása, módosítása, törlése).
Manage profile	A felhasználó adatainak menedzselése (megtekintése, törlése, módosítása), kiegészülve a jelszó visszaállítása funkcióval.

3.1.3 Adminisztrációs felület

Az ügyfél szeretné, ha csak ő rendelkezne admin jogosultsággal. Annak érdekében, hogy mindent átláthatóan és egy helyen tudjon kezelni, egy adminisztrációs felület megvalósítását kezdeményezte, amit egy védett URL-en keresztül tud majd elérni.

Ezen az adminisztrációs felületen elérhetőek lesznek a felhasználók, a projektek, az ezekhez tartozó feladatok listája.

3.1.4 Gem-monitorozó eszköz

Az ügyfél további kérése volt az is, hogy a rendszeren keresztül a projektekhez tartozó gem-ek sebezhetőségét fel lehessen tárni.

Egy alkalmazáshoz tartozhatnak külső könyvtárak, más néven gem-ek. Egy gem lehet hibás, lehetnek sebezhetőségei. Ilyen volt a RubyGems.org esete is, ami a ruby ökoszisztémán alapuló rendszerekhez tartja nyilván a gem-eket. Egy bug-nak köszönhetően lehetősége volt bármelyik felhasználónak eltávolítani vagy módosítani bármelyik gem-et még akkor is, ha a felhasználó nem volt bejelentkezve. [1] Az ilyen és ehhez hasonló hibákat szűrni kell.

3.1.5 Edge case-ek kezelése

Abban az esetben, ha a felhasználó egy létrehozás során hibás értéket ad meg, vagy egy olyan útvonalat próbál elérni, amihez nem szabadna hozzáférnie (pl. egy másik cég projektjét akarja módosítani), akkor egy figyelmeztető üzenet formájában értesíteni kell a felhasználót.

4 FELHASZNÁLT TECHNOLÓGIÁK

A webalkalmazásom fejlesztése során igyekeztem olyan technológiákat választani, amiben sok tapasztalatom van és jártas vagyok, illetve egy stabil és jól működő alkalmazást létre tudok hozni sok hasznos beépített funkcionalitással, amiket később alaposan le tudok tesztelni.

4.1 Ruby

A megvalósításhoz Ruby on Rails keretrendszert választottam, ami Ruby programozási nyelvre épül. Ez a nyelv egy objektum-orientált interpretált szkript nyelv, amit az 1990-es évek közepén fejlesztettek ki. Elsősorban webes alkalmazások és Common Gateway Interface (CGI) szkriptek létrehozásához használják. [2]

4.2 Ruby on Rails

A Ruby on Rails egy ingyenes és teljesen nyílt forráskódú, MVC (Model-View-Controller), architektúrán alapuló szerver-oldali full-stack keretrendszer. Annak ellenére, hogy számos más megoldás van egy webalkalmazás megvalósítására, a Rails mindezektől külön áll az ereje és az eleganciája miatt, hiszen roppant gyorsan és könnyen lehet benne fejleszteni egy robosztus és nagy teljesítményű webes alkalmazásokat. Ez egy folyamatosan fejlődő stabil keretrendszer, ami gyorsan népszerűvé tette a dinamikus webalkalmazások csoportjában.

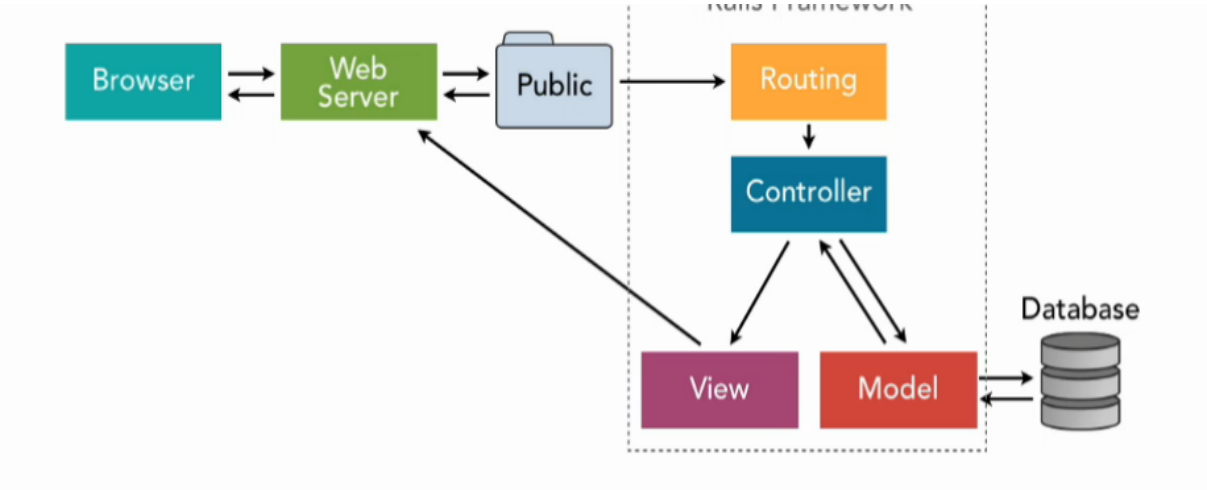
Egy Rails alkalmazás könnyedén bővíthető számos funkcionalitással gem-ek (külső könyvtárak) használatával, mint például felhasználókezeléssel, az alkalmazás tesztelésére szolgáló eszközökkel, CSS keretrendszerrel és még sok mással. Ez számomra rengeteg lehetőséget biztosít egy jó webalkalmazás fejlesztésére. [3]

4.2.1 Működése a gyakorlatban

Amikor valamilyen interakciót csinálunk a felületen, például megnyomunk egy gombot, akkor a böngésző elküld egy kérést, amit megkap egy webszerver és továbbküldi azt a Rails útvonalválasztó motorjának (routing engine). Az útválasztó megkapja a kérést és továbbküldi azt a Controller megfelelő metódusának az URL mintájától függően. A Controller majd kapcsolatba lép a Model réteggel, ami lekérdezi a szükséges adatokat az adatbázisból. Miután

a Controller megkapta a szükséges adatokat, előállítja a nézetet és visszatér azzal a felhasználó felé a böngészőbe. Ez a Rails-ben az alábbi módon néz ki [2. ábra] [4]:

2. ábra
Az MVC szerkezete Ruby on Rails keretrendszerben



4.3 PostgreSQL

A PostgreSQL egy relációsadatbázis-kezelő rendszer, ami hatalmas népszerűségnek örvend a robusztussága, megbízhatósága, adatintegritásának és bővíthetőségének köszönhetően. Alapesetben, amikor létrehozunk egy teljesen új Rails alkalmazást, akkor egy SQLite adatbázist kapunk a kezünkbe, ami egészen jól működik, viszont egy nagyobb alkalmazáshoz kevésnek bizonyul, hiszen csak egyetlen fájlból áll, tekintve, hogy fájlalapú rendszer. Ilyen helyzet például az, amikor az alkalmazásunkat egyszerre több száz felhasználó szeretné használni egyidőben, mivel hiányzik belőle a hozzáférés ellenőrzés. Mindemellett a másik hátránya az író-olvasó műveletek, melyek szerializálva vannak. Ez a gyakorlatban jelentős teljesítményromlást jelent. Éppen ezért jobb megoldás egy kliens-szerver oldali adatbázis-kezelő rendszer használata, mint például a PostgreSQL. [5]

5 TERVEZÉS

5.1 Adatbázis terv

5.1.1 Users tábla

A users tábla tárolja az alkalmazás minden felhasználóját. Hivatkozik a „Companies” táblára, hiszen minden felhasználó tartozhat egy bizonyos céghez. A felhasználókhoz lehet szerepköröket rendelni, amit az enum típusú „role” mező tárol.

- FIRST_NAME: a felhasználó elsődleges neve
- LAST_NAME: a felhasználó családneve
- ROLE: a felhasználó szerepköre (alapértelmezetten ‘nil’)
- COMPANY_ID: a cég, ahova tartozik a felhasználó
- EMAIL: A felhasználó email címe
- ENCRYPTED_PASSWORD: Titkosított jelszó
- RESET_PASSWORD_TOKEN: A jelszó visszaállításához szükséges kód
- RESET_PASSWORD_SENT_AT: A dátum amikor a jelszó visszaállításához tartozó utasítás el lett küldve
- REMEMBER_CREATED_AT: A dátum, amikor megjegyzés funkció be lett állítva
- CREATED_AT: A dátum amikor a felhasználó létrejött (regisztrálás dátuma)
- UPDATED_AT: A dátuma amikor a felhasználó frissítve lett

5.1.2 Messages tábla

Az üzenetküldést megvalósításához három darab tábla létrehozása volt szükséges, mely közül az egyik a Messages tábla. Ez tárolja magát az üzeneteket, hogy melyik felhasználó hozta őket létre, illetve, hogy melyik szobához tartozik.

- CONTENT: Az üzenet szövege
- READ: Logikai érték – az üzenetet elolvasta-e a felhasználó vagy sem.
- USER_ID: A felhasználó, aki az adott üzenetet készítette
- ROOM_ID: Hivatkozás a szobára – melyik szobához tartozik az üzenet
- CREATED_AT: Dátum amikor az üzenet létrejött
- UPDATED_AT: Dátum amikor az üzenet frissült

5.1.3 Rooms tábla

Ez a tábla az üzenőszobákat tartalmazza. Ebbe beletartoznak mind a privát szobák, ahol egy-egy fél tartózkodik, illetve a nyilvános, több felhasználót is tartalmazó szobák. Kikötés, hogy minden szoba nevének egyedinek kell lennie.

- NAME: A szoba egyedi neve
- IS_PRIVATE: Jelöli, hogy a szoba privát-e vagy sem
- CREATED_AT: Dátum amikor a szoba létrejött
- UPDATED_AT: Dátum amikor a szoba objektum frissült
- COMPANY_ID: Hivatkozás a Companies táblára. Egy szoba minden esetben hozzá kell legyen rendelve egy céghez.

5.1.4 Participants tábla

Ez a tábla a privát szobák előállításához szükséges. Látható, hogy hivatkozik a Users és a Rooms táblákhoz. Ha két felhasználó között szeretnénk privát beszélgetést létrehozni, akkor két új rekordot kell beillesztenünk a táblába, hivatkozva a két user-re és egy szobára, ahol a két felhasználó fog beszélni. Ha már létezik egy privát beszélgetés két felhasználó között, akkor nem fog új privát szobát létrehozni számukra. A szobanév egyediségét a két felhasználó azonosítójával értem el.

- USER_ID: Hivatkozás users táblára.
- ROOM_ID: Hivatkozás a rooms táblára
- CREATED_AT: Dátum, amikor az objektum létrejött
- UPDATED_AT: Dátum, amikor az objektum frissült

5.1.5 Projects tábla

A projects tábla a cégekhez tartozó egyes projekteket tartalmazza. Ez a tábla hivatkozik a companies, illetve a statuses táblákhoz, illetve N:M kapcsolatban van a Users táblával. Minden projekthez tartozik egy cím, leírás, egy szín, hogy könnyebb legyen őket megkülönböztetni, illetve egy lockfile mező is. Ez a mező a Rails alkalmazásokhoz tartozó Gemfile.lock elérési útvonalát tartalmazza. Ennek segítségével lehet a sebezhetőségeket feltárni.

- TITLE: A projekt címe/neve
- CONTEXT: A projekt leírása
- STATUS_ID: Mutatás a statuses táblára. Jelzi, hogy az adott projekt milyen státuszban van

- ACQUIRED_AT: Dátum, hogy mikor indult a projekt, mikor került a céghez.
- COMPANY_ID: Hivatkozás a companies táblára.
- LOCKFILE: Ez a mező tárolja a projekthez tartozó Gemfile.lock fájl elérési útvonalát
- COLOR: A projekt színe.
- CREATED_AT: Dátum, amikor az objektum létrejött
- UPDATED_AT: Dátum, amikor az objektum frissült

5.1.6 Companies tábla

Ez a tábla tartalmazza a cégeket.

- NAME: A cég neve
- CREATED_AT: Dátum, amikor az objektum létrejött
- UPDATED_AT: Dátum, amikor az objektum frissült

5.1.7 Tasks tábla

A tasks tábla a feladatokat tartalmazza. Hivatkozik a Projects táblára, ugyanis minden task kell, hogy tartozzon egy projekthez. Emellett hivatkozik még a Users táblához is, hiszen az egyes taskok-hoz több user-t is hozzárendelhetünk. Az egyes feladatokhoz hozzárendelhetünk alfeladatokat, éppen ezért ez a tábla önmagára is hivatkozik. Ez az öröklődés legfeljebb kétszintes lehet.

- PROJECT_ID: Hivatkozás a projekt táblára.
- DUE_DATE: Dátum, a feladat határideje, ameddig el kell végezni azt.
- DESCRIPTION: A feladat leírása
- TITLE: A feladat neve
- PARENT_ID: Ezzel a mezővel hivatkozik önmagára
- COMPLETED: Jelöli, hogy a feladat kész van-e vagy sem (boolean érték)
- CREATED_AT: Dátum, amikor az objektum létrejött
- UPDATED_AT: Dátum, amikor az objektum frissült

5.1.8 Statuses tábla

Ez a tábla a projektek státuszát tartalmazza. Mindegyikhez tartozik egy megnevezés és egy szín.

- TITLE: A státusz megnevezése
- BACKGROUND: A státusz színe

5.1.9 ActiveStorage táblái

Azok a táblák, melyek neve tartalmazza az ‘active’ szót az a Rails ActiveStorage működéséhez szükségesek. Használata a felhasználókhoz tartozó avatárokhoz szükséges, ami egy kép.

5.1.10 ActiveAdmin táblái

Az admin felület létrehozásához használ ActiveAdmin gem által generált táblák.

Az adatbázis séma a következőképpen néz ki [3. ábra]:

3. ábra
Adatbázis sémája

5.2 Szoftver terv

5.2.1 Felhasználókezelés

Az alkalmazásra bárki felregisztrálhat, viszont a dolgozókat a cégekhez csak arra jogosult felhasználók rendelhetnek hozzá. A felhasználókezelés és authorizáció megvalósításához fontos egy olyan megoldást találnom, mely titkosítani tudja a jelszavakat és képes a jelszóvisszaállításra is, tehát ne kelljen újra feltalálnom a kereket. Szintén fontos átgondolni azt is, hogy egy saját alkalmazásba integrált, vagy egy külső hitelesítési szolgáltatóval (ún. third-party authentication provider) megvalósított megoldást használjak.

Szempontok:

- Külső eszköz használata kevesebb kódolást igényel, egyszerűbb bekötni, ugyanakkor fizetős megoldás.
- Egy third-party auth használata jobb választás akkor, amikor esetleg Facebook [6], GitHub, vagy Google fiókkal is engedélyezni szeretnénk a hitelesítést és a felhasználók száma a százezres nagyságot is meghaladja.
- Egy saját és kisebb méretű alkalmazásba integrált megvalósítás ingyenesen megvalósítható és kis felhasználó mennyiség esetén jobb választás.

Éppen ezért szükségem lesz egy bejelentkezés-, egy regisztrációs-, illetve jelszóvisszaállítás felületekre. A felhasználókezelést saját magam fogom bekötni az alkalmazásba gem segítségével.

Az új felhasználók az alábbi oldalon [4. ábra] tudnak bejelentkezni. A megadott e-mail cím segítségével azonosítom a user-t, majd a belépést követően és átirányítom a kezdőoldalra. Abban az esetben, ha rossz adatokat ad meg, hibaüzenet formájában tájékoztatom a felhasználót arról. A „remember me” checkbox lehetőséget ad a hitelesítő adatok megjegyzésére a későbbi belépések gördülékenységének fokozása érdekében. A „Forgot your password?” átirányítja a felhasználót egy új oldalra, ahol az e-mail címet megadva a jelszóvisszaállítási procedúra veszi kezdetét. A jelszót visszaállítani az e-mail cím megadásával van lehetőség, mégpedig úgy, hogy a megadott e-mail címre egy üzenetet küldünk, ami tartalmazni fog egy linket, ami átirányítva a felhasználót egy oldalra új jelszót adhat meg. Egészen addig, amíg a felhasználó nem jelentkezett be, nem láthat belsős céges adatokat.

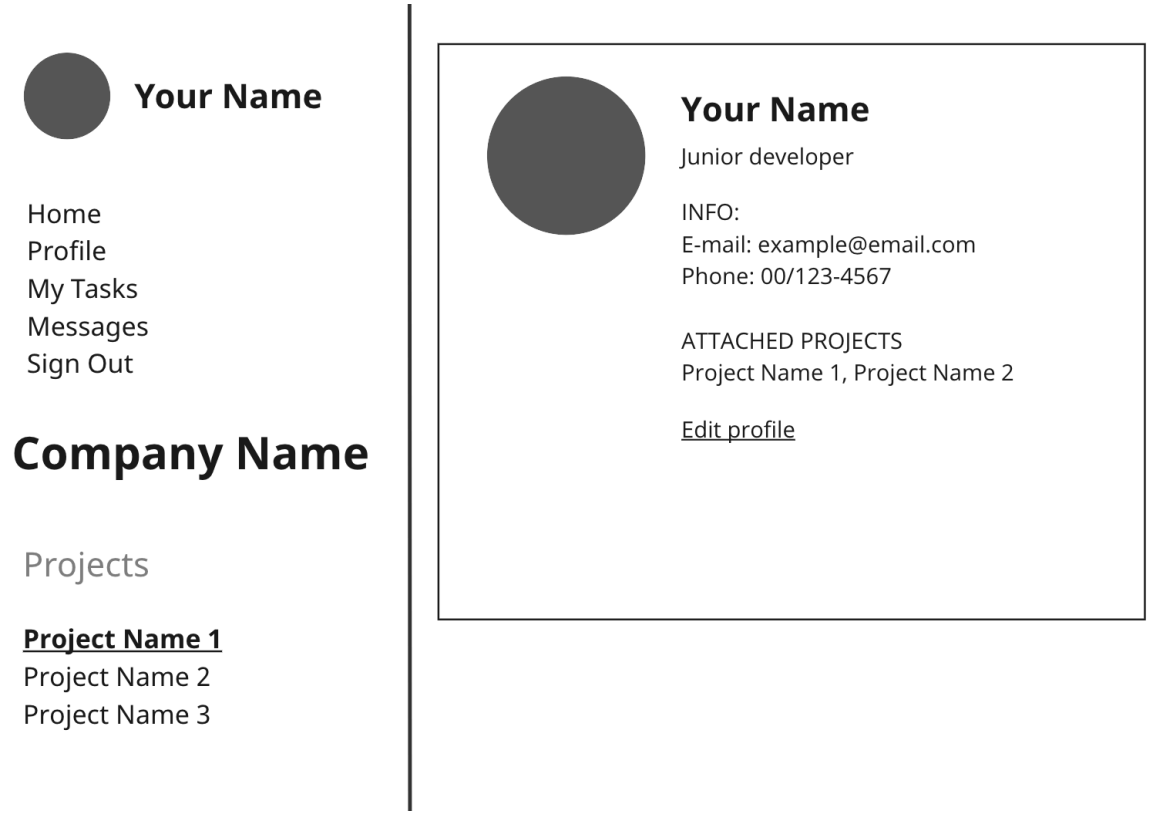
4. ábra
Bejelentkezési felület felhasználói interfész terve

Regisztrálni a bejelentkezési felületen megtalálható „Sign up” gombon keresztül van lehetőség. A regisztrációs oldalon [5. ábra] bekérjük a szükséges adatokat a felhasználótól, ami magába foglalja az e-mail címet, a vezeték- és keresztnevet, illetve a jelszót. A regisztráció során opcionálisan megadható egy avatar kép is. Abban az esetben, ha nem adunk meg semmit, egy alapértelmezett avatar-t állítok be a felhasználónak. Annak érdekében, hogy biztosak legyünk abban, hogy a felhasználó helyesen írta be a jelszavát, kétszer is meg kell adnia. Hibák esetén, ami fakadhat a két jelszó eltéréséből, vagy az egyes adatok helytelen megadásából a felhasználót hibaüzenet formájában értesítem.

5. ábra
Regisztrációs oldal felhasználó felület terve

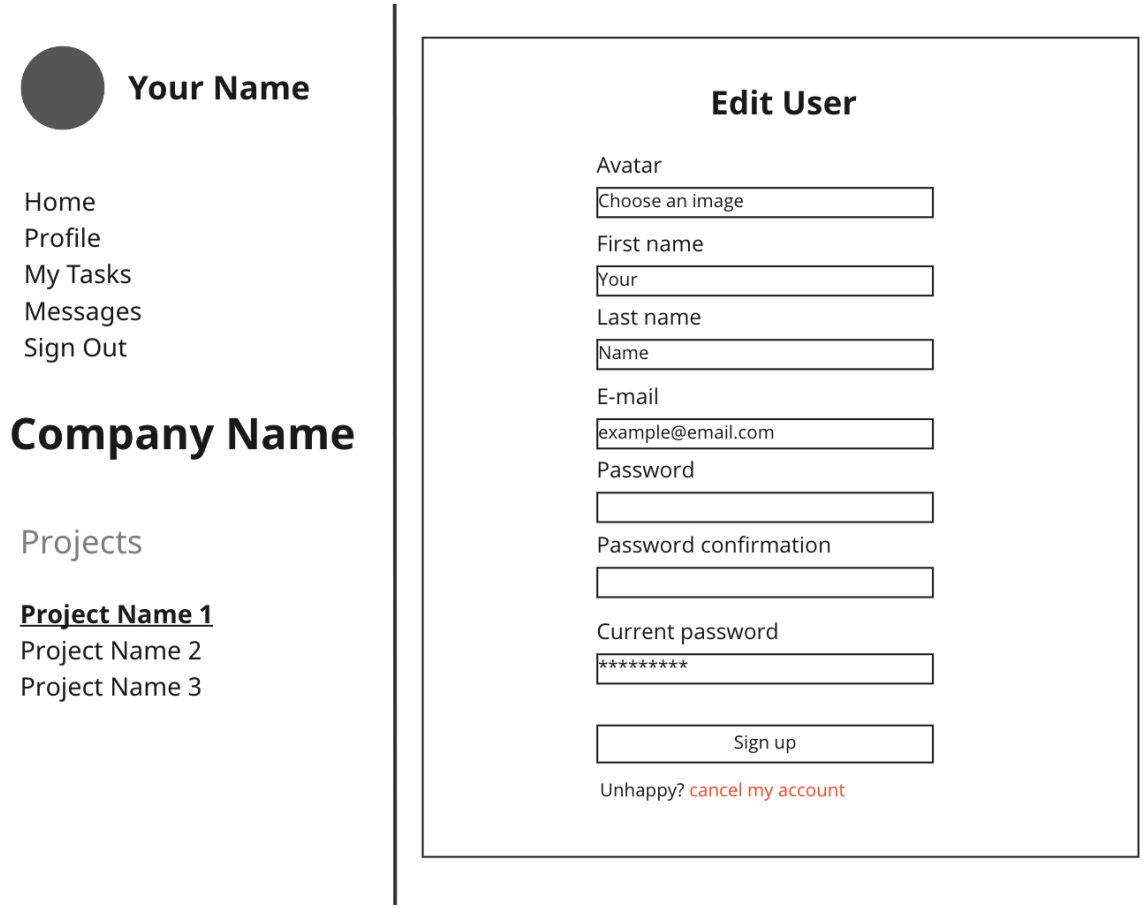
A bejelentkezés után, ha a felhasználó meg szeretné tekinteni a saját profilját, azt az oldalmenüben megtalálható „Profile” link rákattintásával teheti meg. Az oldalon [6. ábra] megjelenítek minden információt, ami a User objektumhoz tartozik (5.1.1 fejezet). Link formájában kilistázom azokat a projekteket, amihez a felhasználó hozzá van rendelve. Az „Edit profile” gomb elnavigálja a felhasználót a szerkesztő oldalra [7. ábra].

6. ábra
Saját profil oldala



A profil szerkesztésének oldalán [7. ábra] lehetőség van a felhasználó minden adatának módosítására. Abban az esetben, ha jelszót is szeretnénk módosítani, meg kell adnunk a jelenleg használatban lévő jelszót is biztonsági okokból. Ugyanezen az oldalon alulra elhelyeztem egy „Uhappy? cancel my account” felirattal ellátott gombot, amire rákattintva törölhetjük a profilunkat.

7. ábra
Profil szerkesztésének oldala



5.2.2 Jogosultságkezelés

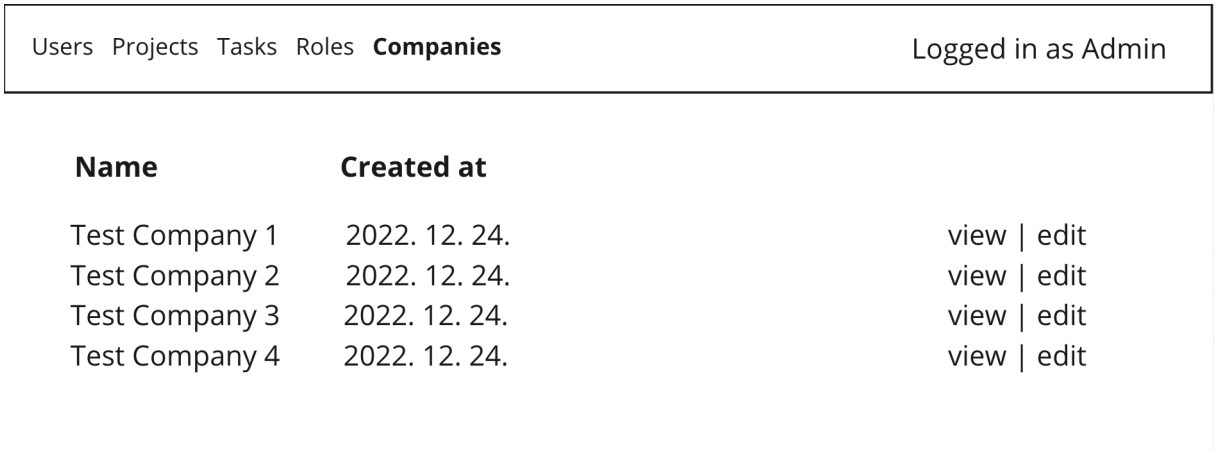
A webalkalmazásban nélkülözhetetlen a szerepkörök használata, hiszen nem minden felhasználó rendelkezhet azzal a jogosultsággal, hogy projekteket szerkeszthessen, felhasználókat hozzon létre, módosítson, vagy éppen töröljön. Továbbá, egyik cég sem szeretné, ha más cégek dolgozói belelátnának a csoportos beszélgetéseikbe, vagy a projektjeik tartalmaiba és azok feladataiba és módosításokat végezzenek el rajtuk. Az ilyen és ehhez hasonló szcenáriók kezelése roppant fontos egy alkalmazásban. Az admin felhasználót terveim szerint kézzel fogom létrehozni, és onnantól kezdve az ő kezében van a lehetőség, hogy megszabja a felhasználók szerepköreit. Ezt az adminisztrációs felületen (5.2.7. fejezet) teheti majd meg.

5.2.3 Adminisztrációs felület

Az adminisztrációs felületen lehetőség van az egyes objektumok teljeskörű szerkesztésére. Itt szeretném átláthatóan megjeleníteni a végfelhasználó számára az egyes objektumokat, és egy kezelhető felületet biztosítani arra, hogy könnyedén lehessen módosítani vagy törölni az

adatokat. Ennek a felhasználói terve az alábbi ábrán [8. ábra] látható:

8. ábra
Adminisztrációs felület

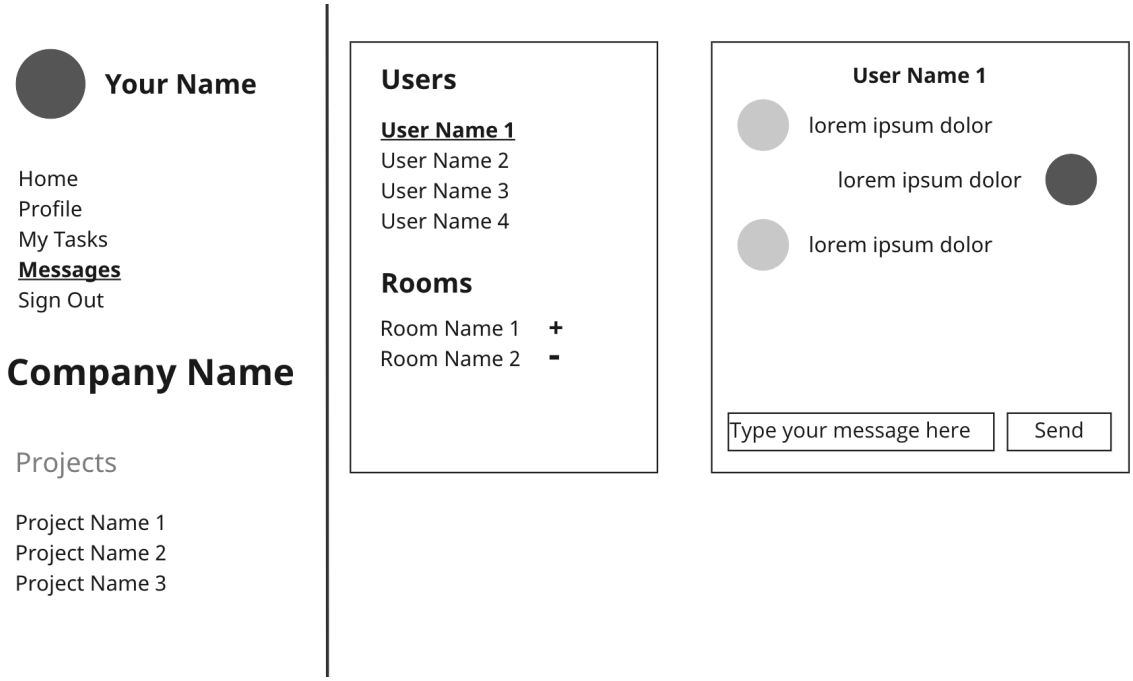


5.2.4 Kommunikáció

A kommunikáció folyhat egy publikus csoportos szobában vagy egy két fős privát szobában két felhasználó között. Ennek megfelelően egy oldalon szeretném megjeleníteni egymás alatt a csoportos és a privát szobák listáját (ami a felhasználók neveit fogja jelenteni). A listától jobbra, az egyes szobák nevére rákattintva megjelenítem a szoba tartalmát, attól függően, hogy melyikre kattint a felhasználó. Ezt a részt egy külön téglalappal jelöltem. Ugyanazon a részfelületen lehet az üzenetünket leírni és elküldeni a „Send” gombbal. A legfontosabb az az, hogy az elküldés után az üzenet azonnal megjelenjen, függetlenül attól, hogy ki küldte azt. Az aktív felhasználó saját üzeneteit, szeretném úgy megjeleníteni, hogy azt jobbra igazítom. Az üzenetek mellé a felhasználó avatar képét is megjelenítem az olvashatóság segítése érdekében. Ha az üzenetek listájának mennyisége és méretbeli nagysága elér egy bizonyos értéket, az üzenőszoba görgethetővé válik, így elkerülhető az, hogy a végtelenségig mennek az oldal aljára az elküldött üzenetek. Továbbá, felül kiírom a szoba nevét, ami a privát szobák esetében a felhasználó neve lesz.

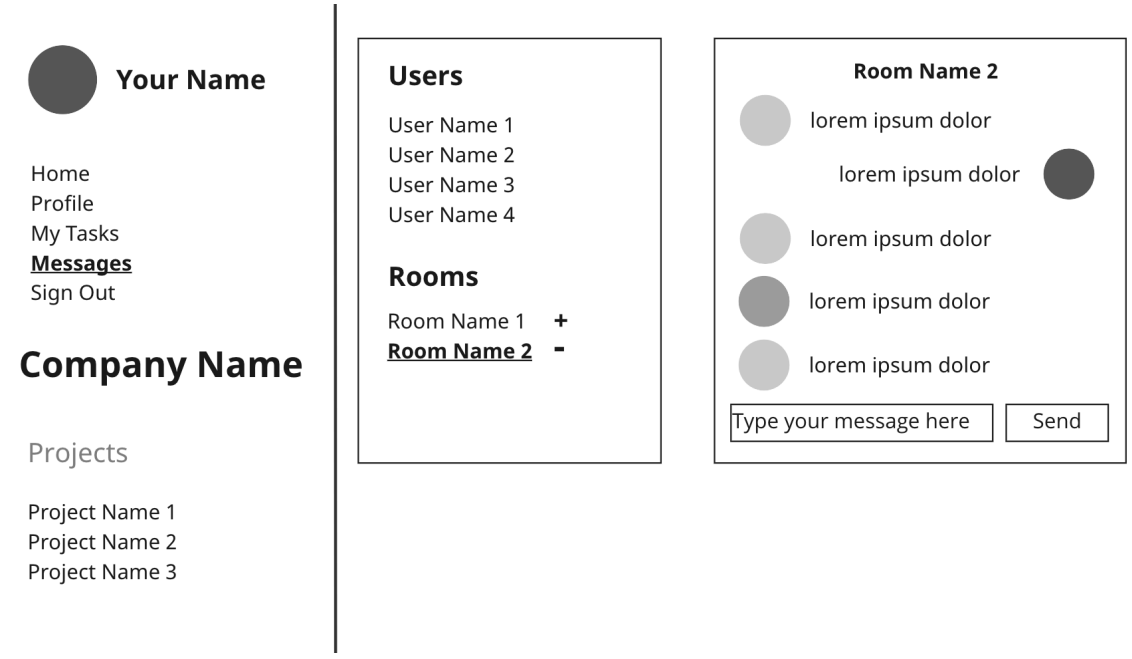
Az alábbi ábra [6. ábra] mutatja a privát üzenőszoba felhasználói interfészét.

9. ábra
Privát üzenőszoba felhasználói felületének a terve



A csoportos üzenőszobák ehhez hasonlóan fognak kinézni [7. ábra]. A szobákhoz bármelyik felhasználónak lehetősége van csatlakozni a plusz gombbal, a mínusszal pedig lecsatlakozni. A kattintást követően egy megerősítést váró üzenet jelenik meg a képernyőn.

10. ábra
Csoportos üzenőszoba felhasználói felületének a terve

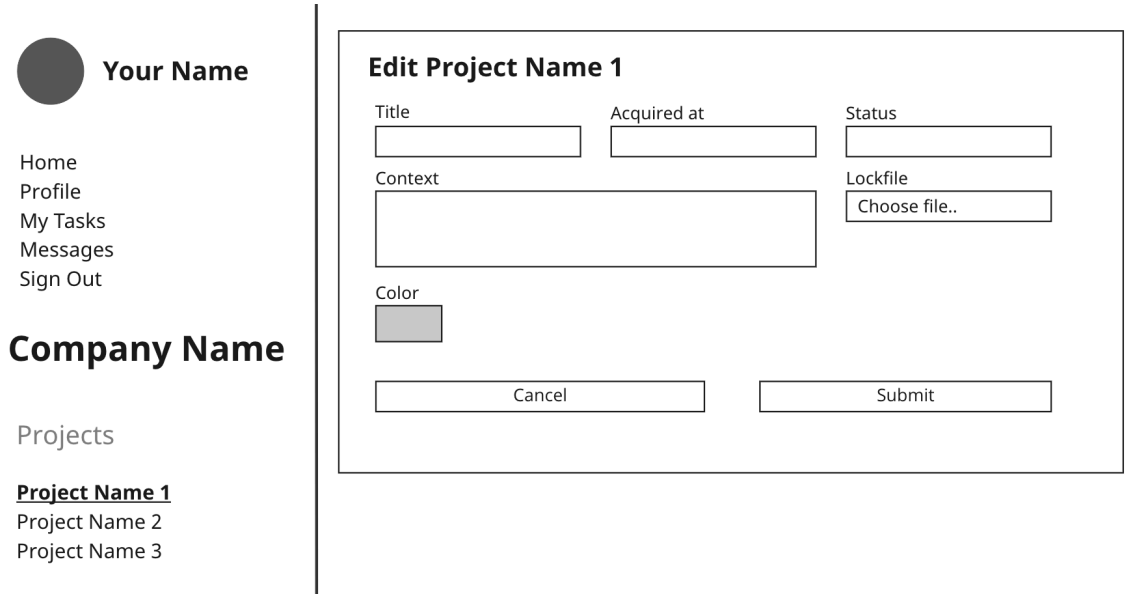


5.2.5 Projektmenedzsment

Az alkalmazásom fő funkcionalitása a projektek nyilvántartása és kezelése. Egy projekt

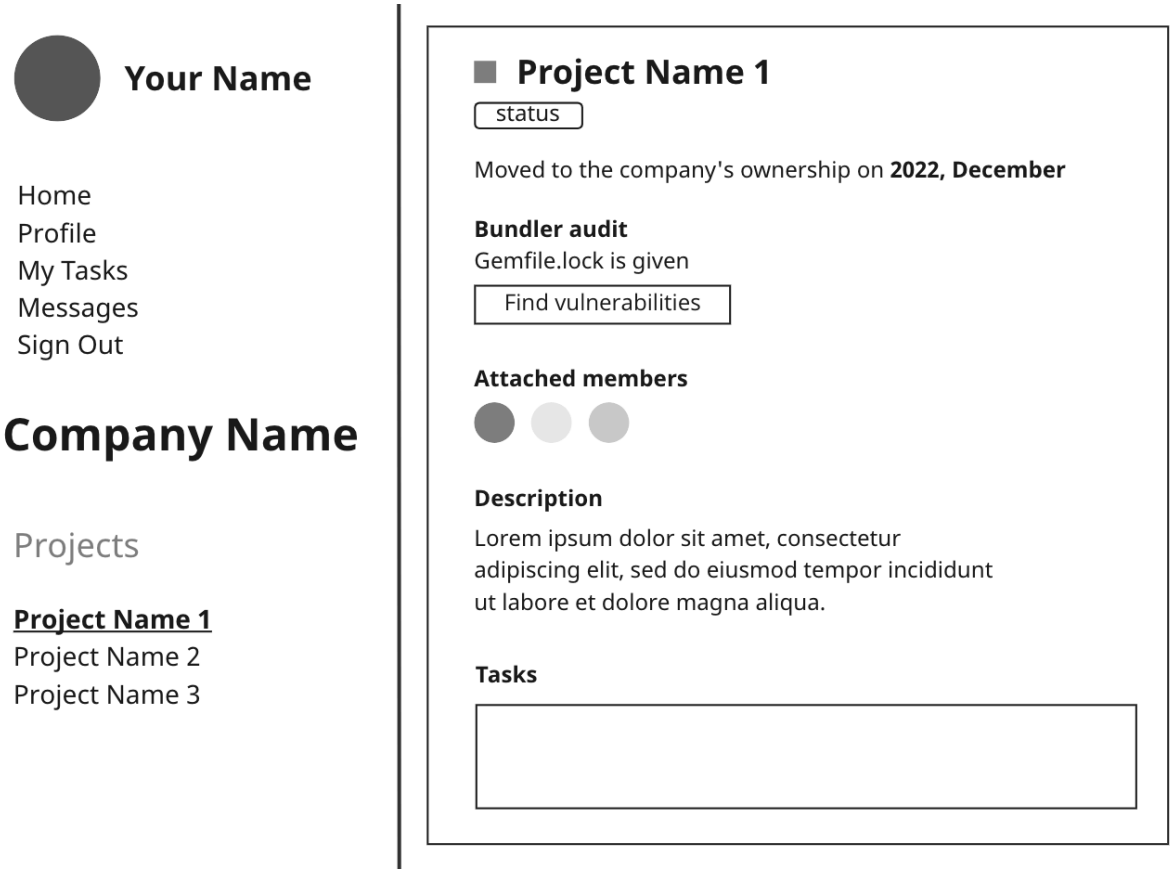
szerkesztésére szolgáló felület az alábbi ábrán [8. ábra] látható. Az oldalon a projekt minden adatát lehetőség van módosítani. Itt opcionálisan meg lehet adni a lockfile fájl helyét, ami a sebzett gem-ek feltárásához szükséges. A projektek könnyebb megkülönböztetése érdekében minden projektnek be lehet állítani egy tetszőleges színt, ami a projekt show oldalán jelenik meg.

11. ábra
Projekt szerkesztésének vagy létrehozásának a felhasználói felületterve



A projektek saját oldalára elnavigálva megtaláljuk a részleteket és információkat, mint például azt, hogy kik dolgoznak rajta, mi a státusza, továbbá megtalálható még hozzátartozó feladatok listája is. A „Bundler audit” rész alatt találjuk a gem-ek sebezhetőségeinek feltárására szolgáló gombot, mely elvezeti a felhasználót a sebezhető gem-ek oldalára. Abban az esetben, ha a projekthez nem tartozik lockfile fájl, akkor a segédüzenet módosul, továbbá a gombra kattintást követően egy figyelmeztető üzenet jelenik meg, ami értesíti a felhasználót erről. A projekten dolgozó felhasználókat a profilképük megjelenítésével jelenítem meg. A „Tasks” rész felülettervét a következő alfejezetben részletezem, így annak helyét egy üres téglalap jelzi. A felhasználói felület terve az alábbi ábrán [9. ábra] látható:

12. ábra
Egy projekt oldalának felhasználói felületterve



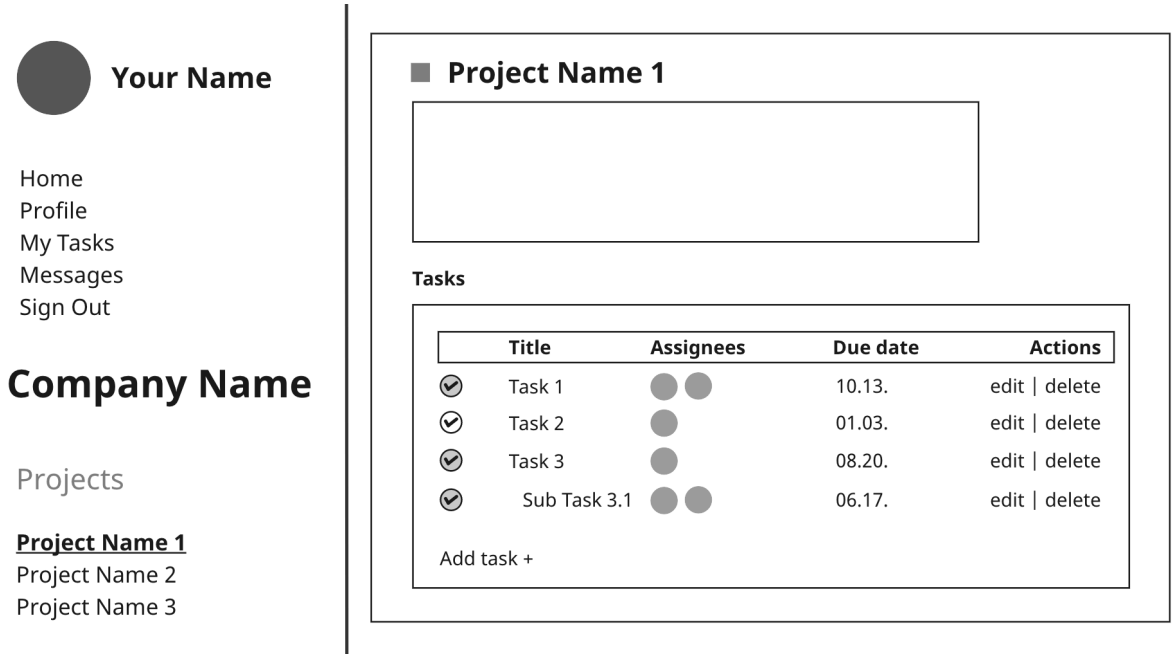
5.2.6 Feladatmenedzsment

A feladatokat több helyen is lehet menedzselni:

1. Az oldalmenüben megtalálható „My Tasks” link elvezet egy külön oldalra, ami egy helyen jeleníti meg azokat a feladatokat, amihez az aktuális felhasználó hozzá van rendelve. Az itt megjelenített feladatokat link formájában jelenítem meg, ami elvezeti a felhasználót a feladat saját szerkesztő oldalára [11. ábra].
2. Minden projekt alatt megtalálhatóak lesznek az elvégzendő, vagy már elvégzett feladatok lista formájában. A listában megjelentem azoknak a felhasználóknak a profilképét, akik hozzá van rendelve. Rákattintva a felhasználók profilképeire eljutunk a felhasználó saját oldalára. Mindemellett helyett kapott egy kattintható pipa körikon is, amivel a projekt státuszát lehet egyszerűen módosítani. Rákattintva az ikon színe zöldre és egy értesítő üzenet jelzi a felhasználónak, hogy a feladat státusza megváltozott. Továbbá, a feladatokhoz megjelenítem a dátumot, ami a feladat elvégzésének a határideje, illetve a szerkesztő és törlő ikonokat, amikkel az adott feladatot lehet vagy szerkeszteni, vagy törölni.

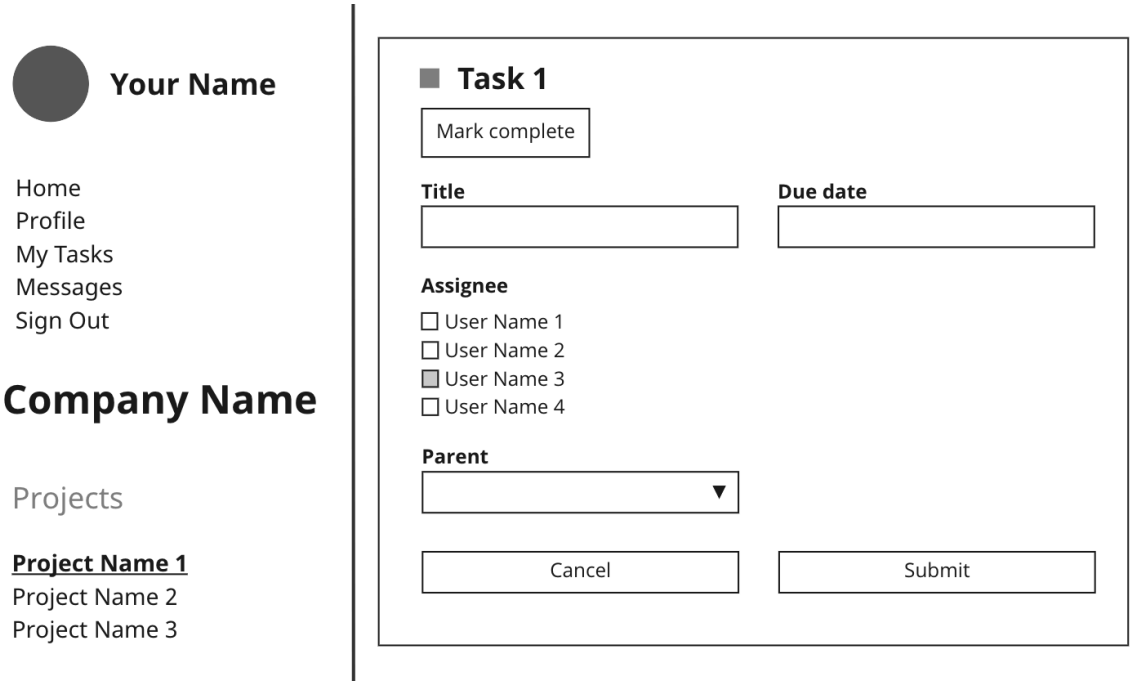
A lista alatt helyet kapó „Add task +” gombbal lehetőségünk van újabb feladatokat készíteni és hozzárendelni a projekthez. A felhasználói felület tervét az alábbi ábra [10. ábra] mutatja:

13. ábra
Projekthez tartozó feladatok listájának felhasználói felületterve



Az egyes feladatok szerkesztésére és létrehozására szolgáló oldal felhasználói interfésze pedig a következőképpen fog kinézni [11. ábra]:

14. ábra
Egyes feladatok elkészítésére és szerkesztésére szolgáló oldal felhasználói interfészének a terve

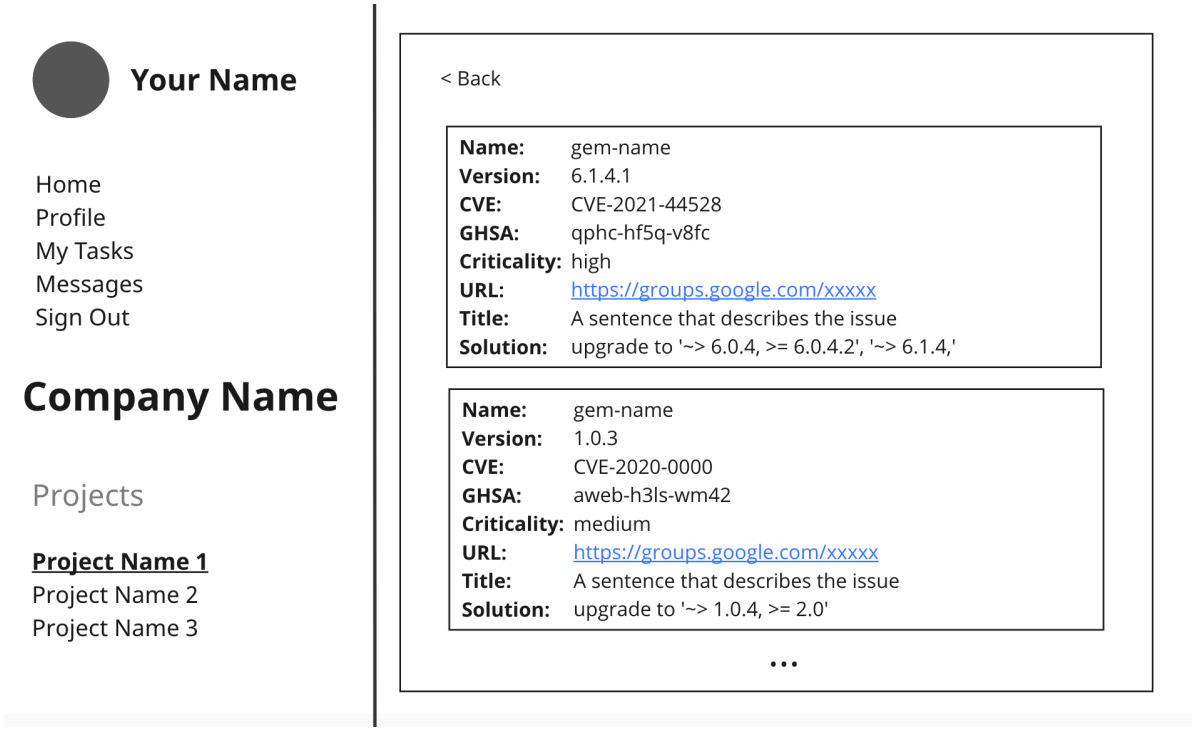


Itt megadhatunk minden szükséges adatot egy feladat módosításához vagy létrehozásához. A feladathoz hozzárendelni a felhasználókat az „Assignee” rész alatt található listán keresztül lehet a checkbox-ok segítségével. Ez egy egyszerű módja a hozzárendelésnek több felhasználó esetében.

5.2.7 Sebezhető gem-ek feltárása

A projekt oldalán fel lesz tüntetve egy „Find vulnerabilities” felirattal ellátott gomb, melyre rákattintva a projekthez tartozó gem-ek sebezhetőségei kilistázódnak ahogyan azt az alábbi ábra [12. ábra] is mutatja. A 5.2.4 fejezetben említett esetben, ha nincs megadva a projekthez lockfile fájl, akkor a felhasználót figyelmeztető üzenet formájában értesítem arról.

15. ábra
Gem-sérülékenység vizsgálat eredményének a felhasználói felületterve



Minden egyes sebezhetőséget külön téglalapban elhelyezve jelenítek meg az oldalon. Az oldal tetején tud a felhasználó visszanavigálni a projekt oldalára.

Egy gem-ről a következő információkat jelenítem meg:

1. **Name**: a gem neve
2. **Version**: a gem aktuális verziója, amit a project használ
3. **CVE**: Common Vulnerabilities and Exposures (CVE) ID (a sebezhetőség azonosítója)
4. **GHSA**: GitHub Security Advisory ID (a GitHub saját biztonsági sebezhetőségi adatbázisában lévő tanács egyedi azonosítója)
5. **Criticality**: meghatározza, hogy mennyire kritikus a sebezhetőség
6. **URL**: Google Groups oldalra mutató URL, ami a sérült gem feltárásáról szól
7. **Title**: A sebezhetőség rövid leírása
8. **Solution**: Azon verziók listája, ahol a hiba már javítva van

6 MEGVALÓSÍTÁS

6.1 Felhasználókezelés

A felhasználókezelést az alkalmazáson belül a devise nevű gem-mel fogom implementálni. A devise egy elképesztően rugalmas, teljesen MVC megoldás az autentikáció megvalósítására. Mindemellett teljesen moduláris, tehát csak azt építem be és használom fel, amire valóban szükségem van.

A „devise” a felhasználó jelszavát titkosítja (hash-eli), majd eltárolja az adatbázisban annak érdekében, hogy validálni tudja a felhasználót a bejelentkezés során. Továbbá lehetőséget ad az e-mail cím megerősítésére, jelszó visszaállítására, token alapú belépés megjegyzésére, nyomon követi a bejelentkezések számát és IP címét, ahonnan bejelentkeztünk és még sok-sok más hasznos funkciót ad készen a kezünkbe az előbb felsoroltak mellett. [7]

6.2 Jogosultságkezelés

A jogosultságokat a „CanCanCan” gem-mel fogom megvalósítani. Nulladik lépésként az adatbázisban a felhasználó táblájához egy „role” nevezetű extra mezőt adok hozzá, mely segítségével tárolom, hogy milyen szerepköre van az adott felhasználónak. Ez az érték lehet:

- nil (alapértelmezett null érték)
- writer
- admin

Ezt követően a „CanCanCan” külső könyvtár segítségével korlátozni tudom, hogy milyen erőforrásokhoz férhet hozzá a felhasználó. Minden jogosultság definiálható egy vagy több ability nevű ruby fájlban, ezáltal a logika egy helyen van tárolva. Az én esetemben elég egyetlen egy ilyen fájl mert nem kell sok esetet meghatároznom.

1. kód
User projekthez tartozó jogultságainak definiálása

```
def project_abilities(user)

  if user.has_role? :admin
    can :manage, :all
  elsif user.has_role? :write
    can :manage, Project, company_id: user.company_id
  else
    can :read, Project, company_id: user.company_id
    can :manage, Project, users: { id: user.id }
  end
end
```

Tehát, ha a felhasználó olyan útvonalat próbál elérni, melyhez nem lenne jogosultsága (pl. egy másik cég projektjeit kívánja módosítani), át fogom irányítani a gyökér útvonalra azzal a hibaüzenettel, hogy nincs jogosultsága az oldal elérésére.

2. kód
Szerepkörök enum típusú listája és a metódus ami meghatározza, hogy rendelkezik-e az adott jogosultsággal

```
enum role: {
  admin: 'admin',
  write: 'write'
}

def has_role?(role_desc)
  role.try(:to_sym) == role_desc.to_sym
end
```

6.3 Kommunikáció

Egy üzenetküldő alkalmazás egyik legfontosabb ismérve, hogy az üzenetek azonnal (real-time) megjelennek az üzenet elküldésének pillanatában, az oldal ráfrissítése nélkül. Nagyon kellemetlen viselkedés lenne, ha csak akkor kapnánk meg a kapott és az elküldött üzeneteket, ha az oldalra ráfrissítünk. Hogy ezt elkerüljem, az egyes üzeneteket websocket-en keresztül fogom megjeleníteni.

Hogy ezt megvalósítsam, a Hotwire-t [8] fogom használni, azon belül is a Turbo-t a Hotwire hármasai közül (Turbo, Stimulus, Strada). Turbo segítségével magas performanciájú, modern webalkalmazásokat lehet megvalósítani anélkül, hogy rengeteg egyedi JavaScript kódot kellene írni. A Turbo-n belül is több eszközt meg tudunk különböztetni, melyek a következőek:

- Turbo drive
- Turbo frame
- Turbo stream

- Turbo native

Mindezek közül én a Turbo stream-et fogom használni. A stream lehetővé teszi azt, hogy HTML darabokat (snippet-eket) küldjek a böngésző számára és az kis módosításokat hajtson végre a DOM-on (Document Object Model). A Hotwire úgy képes valós időben frissíteni a DOM-ot, hogy partial-ökön keresztül jeleníti meg a kívánt tartalmat. [9]

Fontolóra kell venni azt is, hogy amikor elküldünk egy üzenetet, akkor elvárnánk azt, hogy minden egyes felhasználó, aki a szobában tartózkodik és figyeli azt, rögtön megjelenjen számukra az üzenet, nem pedig csak a küldő számára. Itt jönnek képbe a broadcast-ok, hogy minden egyes feliratkozott felhasználónak stream-eket tudjunk küldeni egyszerre, egy időben.

3. kód
Broadcast definiálása az üzenetekhez

```
after_create_commit { broadcast_append_to self.room }
```

A kommunikáció folyhat két ember között privát szobában, illetve üzenőszobákban is, ahol egyszerre több ember is tartózkodhat, ahogyan azt a tervezés során (5.2.3 fejezet) korábban részleteztem. Nagyon fontos lekezelni azt, hogy csak és kizárólag azokkal tudjunk beszélgetést kezdeményezni, akikkel egy céghez tartozunk, illetve ne tudjunk más céghez tartozó szobákhoz hozzáférni. Ezt úgy érem el, hogy csak azokat a felhasználókat jelenítem meg a listában, akikkel egy cég alá tartozik az aktív felhasználó. Továbbá, a korábban részletezett CanCanCan gem segítségével egy új kritériumot határozok meg, mégpedig azt, hogy még mielőtt elérnénk a megfelelő vezérlő megfelelő metódusát (controller), ami azért felel, hogy visszaadja a korábban folytatott beszélgetést a felhasználóval, vizsgálom, hogy az aktív felhasználó (aki ez esetben én vagyok) cége megegyezik-e azzal a felhasználóéval, akivel beszélgetni szeretnék.

A két felhasználó közötti privát üzenetküldést úgy valósítottam meg, hogy az adatbázisban létrehoztam egy szobát, aminek a „private” mező értékét „true”-ra, a „name” mező értékét pedig a két felhasználó „id” azonosítójából alkotott „private_X_Y”-ra állítottam. Ezáltal csak és kizárólag egy szoba fog szerepelni a két felhasználó között az adatbázisban.

4. kód
Egyedi szoba létrehozása két felhasználó számára

```
def self.create_private_room(users, room_name, company)

  single_room = Room.create(name: room_name, is_private: true, company_id:
company.id)

  users.each do |user|
    Participant.create(user_id: user.id, room_id: single_room.id)
  end

  single_room
end
```

A terveimben (5.2.3. fejezet) az üzeneteket úgy jelenítettem meg, hogy az aktív felhasználó a saját üzeneteit eltolva látja, viszont egy olyan problémába ütköztem, hogy az aktuális felhasználót (devise gem-nek a current_user segédmetódusa) nem lehet elérni a hotwire partial-jében, így azt az üzenetet is úgy jelenítem meg, mint a beszélgető partner üzeneteit, vagyis eltolás nélkül.

5. kód
Az üzenet partial-je

```
.my-2.flex.items-center
  .project-avatar{style: "background: url('#{ user_avatar(message.user.id)
}')"}
  %span.ml-3= message.content
```

Továbbá, szeretném a felhasználó számára megjeleníteni, hogy mennyi olvasatlan üzenete van. Ezt az oldalmenüben fogom megtenni a „Messages” linknél számként egy buborékban. Amint a felhasználó elolvasta azt, az üzenet már nem olvasatlanként fog megjelenni. Ehhez külön el kell tárolnom az üzenetekhez boolean értékként, hogy az olvasott-e vagy sem.

6. kód
Felhasználó olvasatlan üzeneteinek összegyűjtése

```
def self.unread_messages(user)
  query = <<-SQL
    SELECT r.name, m.content
    FROM users u JOIN participants p ON u.id = p.user_id
    JOIN rooms r ON p.room_id = r.id
    JOIN messages m ON r.id = m.room_id
    WHERE p.user_id = #{user.id}
    AND m.user_id != #{user.id}
    AND m.read = FALSE;
  SQL

  ActiveRecord::Base.connection.exec_query(query)
end
```

Annak érdekében, hogy az adatbázisban ne szerepeljen kétszer ugyanaz a szoba két felhasználó számára, meg kell vizsgálni. Ezt a vizsgálatot minden létrehozás elött lefuttatom.

7. kód
Szoba létezésének vizsgálata

```
def confirm_participant
  if self.room.is_private
    is_participant = Participant.where(user_id: self.user.id, room_id:
self.room.id).first
    throw :abort unless is_participant
  end
end
```

6.4 Projekt- és feladatmenedzsment

A „Projekt” és „Task” objektumok egy teljesen egyszerű CRUD objektumok lesznek, amik a RESTful útvonalválasztás módszerét követik. A projekt útvonalait egy példán keresztül [10] mutatom be, mely tökéletesen szemlélteti, hogy milyen útvonalakat tartalmaz a REST architektúra [2. táblázat]:

2. táblázat
RESTful útvonaltervezés

GET	/projects	projektek listázása
GET	/projects/new	visszatér egy form-mal az új projekt létrehozásához
POST	/projects	létrehoz egy új projektet
GET	/projects/:id	visszaad egy projektet
GET	/projects/:id/edit	visszatér egy form-mal egy adott projekt módosításához
PATCH/PUT	/projects/:id	Módosít egy adott projektet
DELETE	/projects/:id	Töröl egy adott projektet

6.5 Sebezhető gem-ek feltárása

A ruby-advisory-db-nek kezeli az adatbázist, ami tárolja a sérült Ruby gem-eket. Ha tudni szeretnénk, hogy az alkalmazásunk használ-e olyan gem-et ami sérült vagy sebezhető, akkor a Gemfile.lock fájlt át kell vizsgálnunk és össze kell egyeztetni az előbb említett adatbázissal. A „bundler-audit" gem pontosan így működik. [11]

Amikor lokálisan használjuk a bundler-audit-ot, a parancs lefutása után minden alkalommal frissíti az adatbázist, vagy ha még nem létezett korábban akkor létrehozza. Ezt lekövetve, nekem is úgy kellett beépítenem az alkalmazásba ezt az eszközt, hogy amikor a „Find vulnerabilities" gombra kattint a felhasználó, az előbbi viselkedés történjen.

A bundler-audit a többi gem-hez hasonlóan nyílt forráskódú, éppen ezért a kód és a dokumentáció alapos áttanulmányozását követően be tudtam építeni az alkalmazásomba az igényeimnek megfelelően.

A megvalósításhoz két lehetőséget vázoltam fel az ügyfélnek. Az első opció az volt, hogy a lockfile-t a projekthez a felhasználó (aki jogosult a projektek menedzselésére) kézzel adja hozzá a saját számítógépéről beolvasva. A másik megoldás, ami sokkal dinamikusabb és hatékonyabb az az, hogy nem kézzel töltjük fel a fájlt, hanem URL-en keresztül rámutatunk a Gemfile.lock fájl git repository útvonalára. Ha ezt a megoldást választjuk, akkor a lockfile tartalma a mindenkori legfrissebb lesz, hiszen, ha bármikor frissül, felülíródik, akkor erről a rendszer tudni fog. Viszont, ez a megvalósítás jelentősen növeli az alkalmazás komplexitását, mert szükséges tudni azt, hogy mikor kell a git repository-ban található lockfile-t letölteni, hogy az felülírja az alkalmazásban található lockfile tartalmát. Ezt a mechanizmust automatizálással lehetne megoldani.

Az ügyfél a javaslatomra az első opciót választotta, ami ugyan nem dinamikus megoldás, de annál egyszerűbb és időt nyerhetek vele.

Mindezek mellett szükséges a lockfile-t eltárolni. Erre is több lehetőség áll rendelkezésre. A legjobb megoldás az lenne, ha felhő alapú szolgáltatáson keresztül (pl. AWS, Azure stb.) tárolnánk őket, így, ha több millió fájl tárolására van szükség, az sem jelentene problémát. Az ügyfél a javaslatomra inkább a lokális tárolás megoldását választotta. Ezáltal a komplexitás jelentősen csökkeni fog, hiszen nem kell egy külső rendszert beintegrálni az alkalmazásunkba, illetve ezek a lockfile-ok nagyon kis méretűek (1-2 Kilobyte), így egyelőre nem kell a

tárhelykapacitás miatt aggódni. Ha mégis szükségessé válik a bővítés, akkor egy felhő alapú tárhely bekötése nélkülözhetetlen lesz.

A projekt saját oldalán az 5.2.4 fejezetben említett „Find vulnerabilities” gombra kattintva átvizsgálom a lockfile-t és az eredményt megjelenítem egy külön oldalon. Hogy mely gem-ek a támadhatóak, az a bundler-audit adatbázisában vannak nyilvántartva. Amikor lefuttatjuk a vizsgálatot, ezt az adatbázist (ruby-advisory-db) letöltöm (feltéve, hogy még nem létezik, ha igen, akkor nem). Ha a projektben lévő gem egy adott verziója megtalálható ebben az adatbázisban, akkor azt hozzáadom a riporthoz és továbbítom ezt a felhasználó felé.

Továbbá azt is ellenőrzöm a vizsgálat során, hogy a gem forrása biztonságos-e vagy sem. Egy gem forrása lehet *http://*, vagy *git://*. A nem biztonságos források szintén nyilván vannak tartva az adatbázisban, így, ha megtalálható benne, akkor azt szintén sebezhetőnek fogja találni és a végeredményben meg fogom jeleníteni a felhasználó számára.

8. kód
A lonkfile egyes sirainak (gem-ek) átvizsgálása [12]

```
def scan_sources(options={})
  return enum_for(__method__,options) unless block_given?

  @lockfile.sources.map do |source|
    case source
    when Source::Git
      case source.uri
      when /^git:/, /^http:/
        unless internal_source?(source.uri)
          yield Results::InsecureSource.new(source.uri)
        end
      end
    when Source::Rubygems
      source.remotes.each do |uri|
        if (uri.scheme == 'http' && !internal_source?(uri))
          yield Bundler::Audit::Results::InsecureSource.new(uri.to_s)
        end
      end
    end
  end
end
```

6.6 Adminisztrációs felület

Az adminisztrációs felületet az „Active-admin” külső könyvtár segítségével valósítottam meg. Ez a gem már alapból tartalmaz olyam megoldásokat, amik egy ilyen felületnek tartalmaznia kell, így könnyedén létre tudtam hozni egy olyan oldalt, ami lehetőséget ad az objektumok

menedzselésére.

6.7 Élesítés

Amikor az alkalmazás élesítésére kerül sor, két opció jöhet szóba: VPS, vagy PaaS. Az én alkalmazásomat Heroku-ra fogom élesíteni, ami egy PaaS felhő alapú szolgáltatás. Ingyenesen biztosít alap funkcionalitásokat, amik egyelőre teljesen megfelelnek számomra.

Mielőtt élesítettem a rendszert, a *database.yml* fájl tartalmát az éles környezetnek megfelelően átalakítottam. Ezt követően létrehoztam egy új heroku alkalmazást, majd a kódot feltöltöttem a heroku saját repository-jába. Végezetül az adatbázis migrációk lefuttatásával az alkalmazás éles környezetbe került. A folyamat alatt semmilyen hibába nem ütköztem.

7 TESZTELÉS

Az alkalmazást az RSpec és Capybara gem-ekkel teszteltem le. RSpec használatával BDD módszert alkalmazva a megírt tesztek először hibát dobtak, majd ezt követően írtam meg hozzá a várt funkcionalitást, ami a teszt lefutását, más néven kizöldülését eredményezte. A két eszköz segítségével a webalkalmazás teljeskörűen lett letesztelve, mely magába foglalja a modell, nézet és vezérlők hármasát.

Az alkalmazást ezt követően nem az éles, hanem egy ún. staging, azaz egy tesztelő környezetben élesítettem. Itt a felhasználóknak lehetősége volt end-to-end teszteket csinálni, és ha bármilyen hiba is volt a rendszerben, az itt előjött.

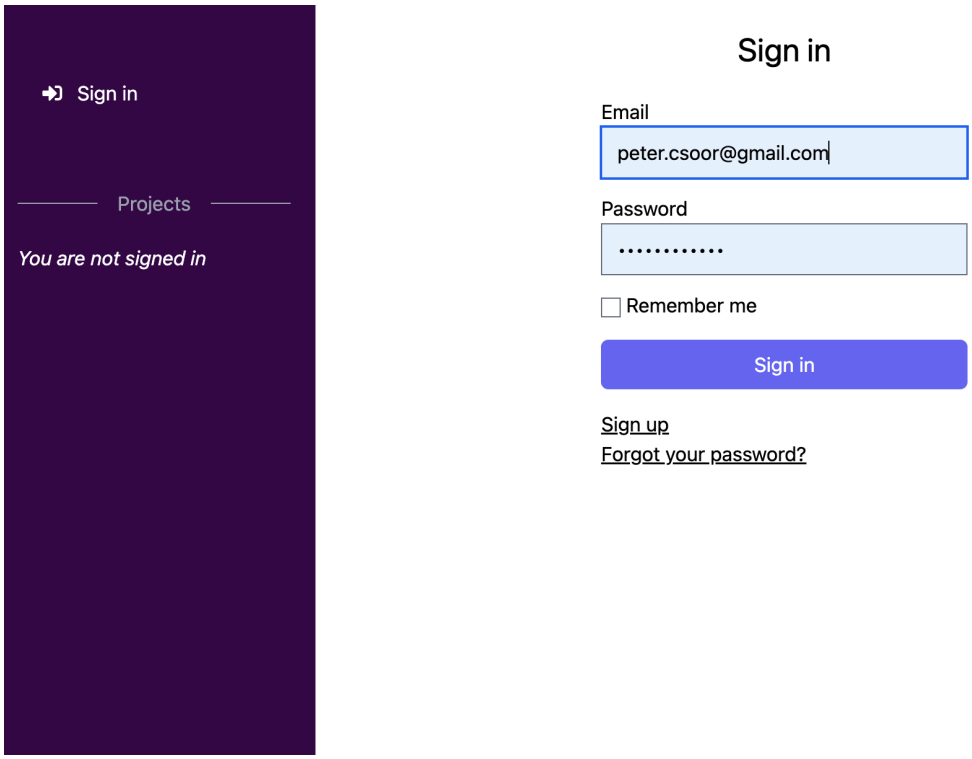
8 EREDMÉNYEK

A tervek és az előzetesen egyeztetett követelmények alapján elkészült Ruby on Rails és PostgreSQL technológiákon futó alkalmazást élesítettem Heroku-n.

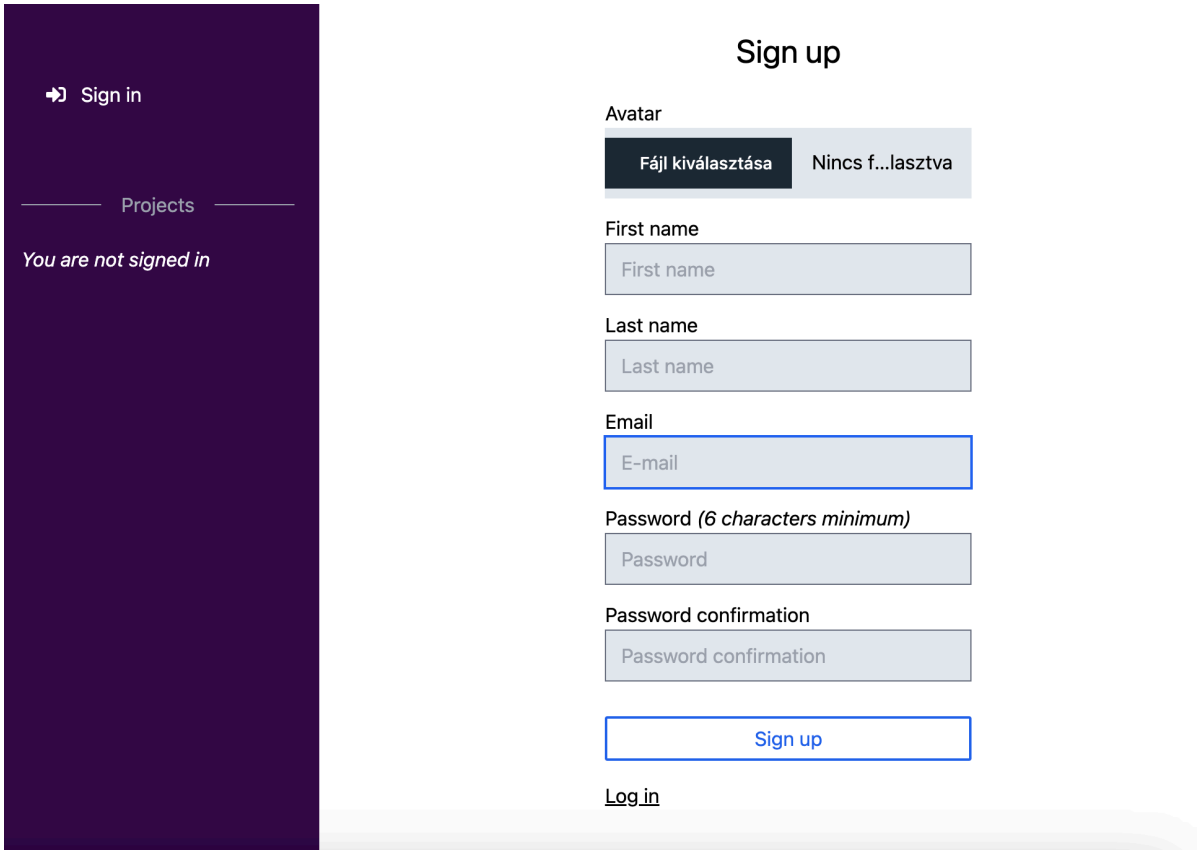
Az élesítést követően az ügyfél letesztelte az alkalmazást és a saját cégéhez tartozó projekteket létrehozta. A dolgozók felregisztráltak, majd ezt követően az admin (aki az ügyfél) hozzárendelte a dolgozókat a projektekhez és megkezdődött az agilis projekt- és feladatmenedzsment.

Az ügyfél elégedett volt a a végeredménnyel, és sikerült egy olyan alkalmazást elkészítenem és élesítenem, ami akár több száz dolgozó hasznára fog válni a jövőben. A specifikációban előírt és megkövetelt funkcionalitások közül sikerült mindet beépítenem az alkalmazásba.

1. kép
Bejelentkezési felület

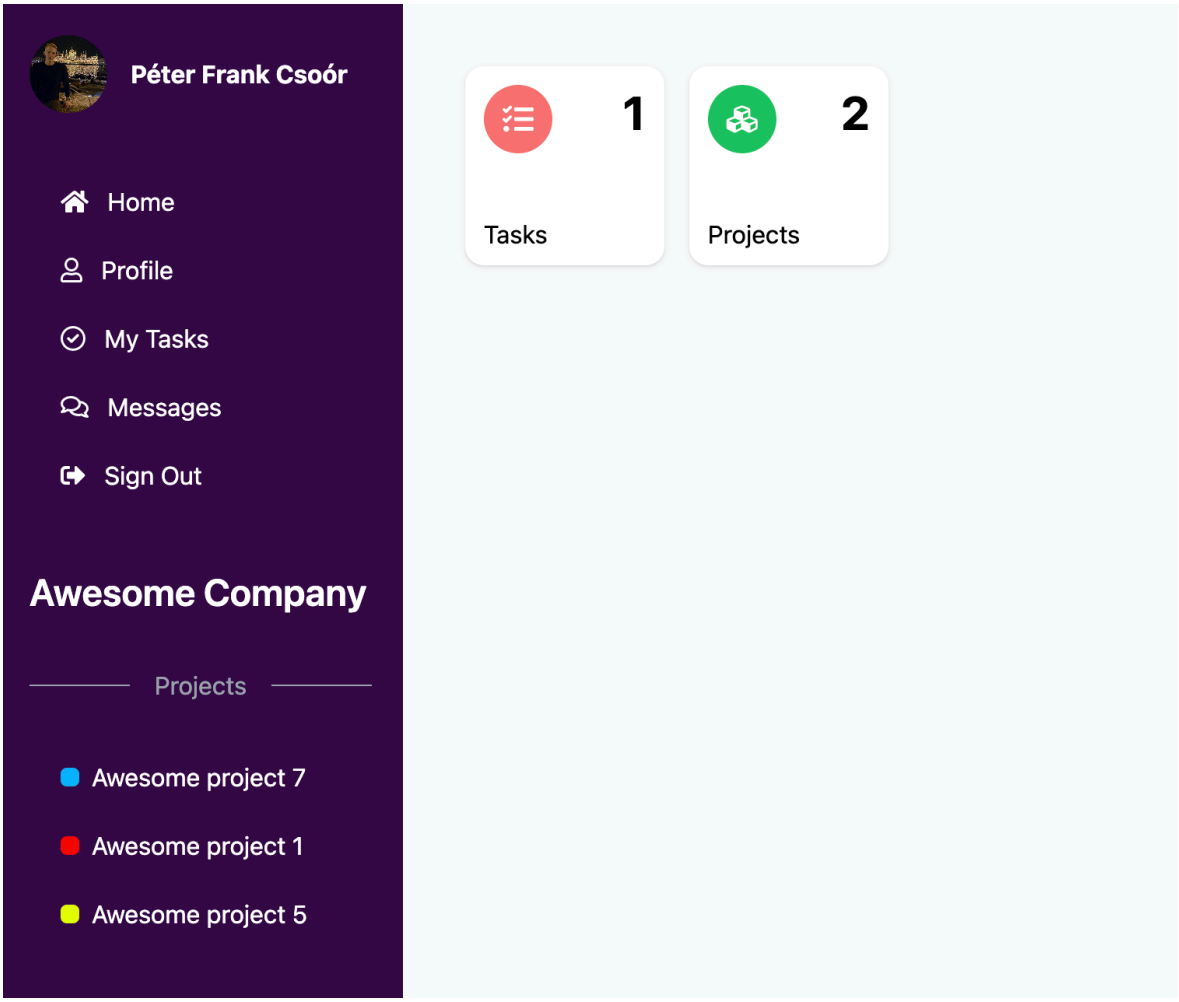


2. kép
Regisztrációs felület



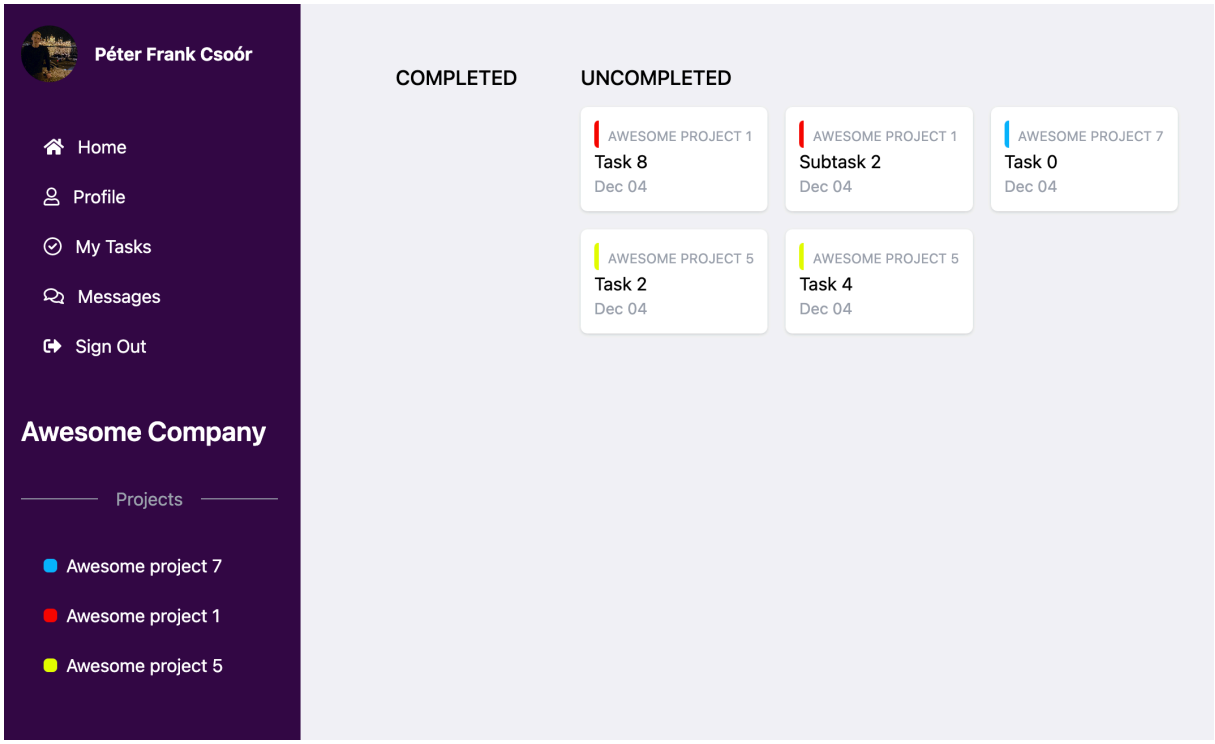
A könnyebb átláthatóság érdekében már a kezdőoldalon megtalálható az aktuális felhasználóhoz rendelt feladatok és projektek száma [3. kép].

3. kép
Kezdőoldal

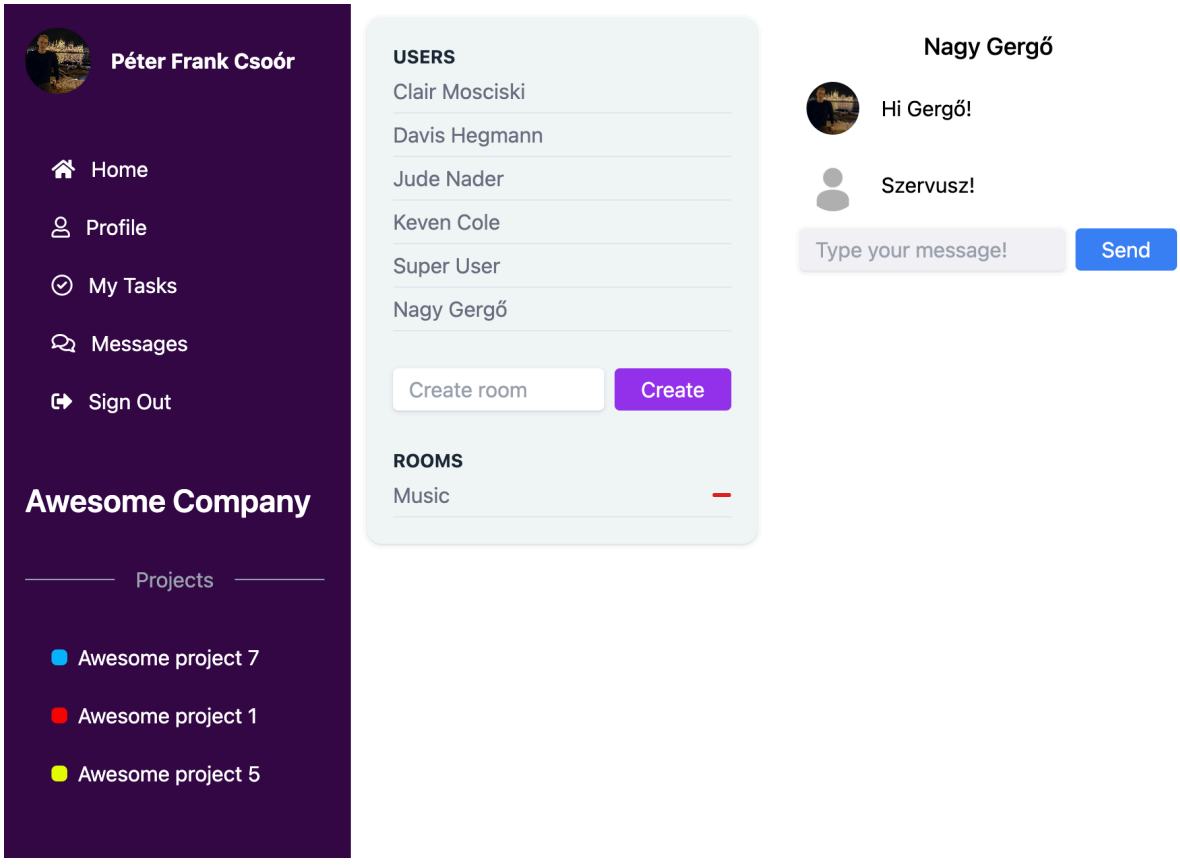


A tervek alapján megvalósított feladatmenedzsment oldal, ami csokorba szedi a felhasználóhoz rendelt feladatokat az alábbi képen [4. kép] látható. Az oldalon külön oszlopokba rendelve látható az elkészült és még elkészítésre váló feladatok listája.

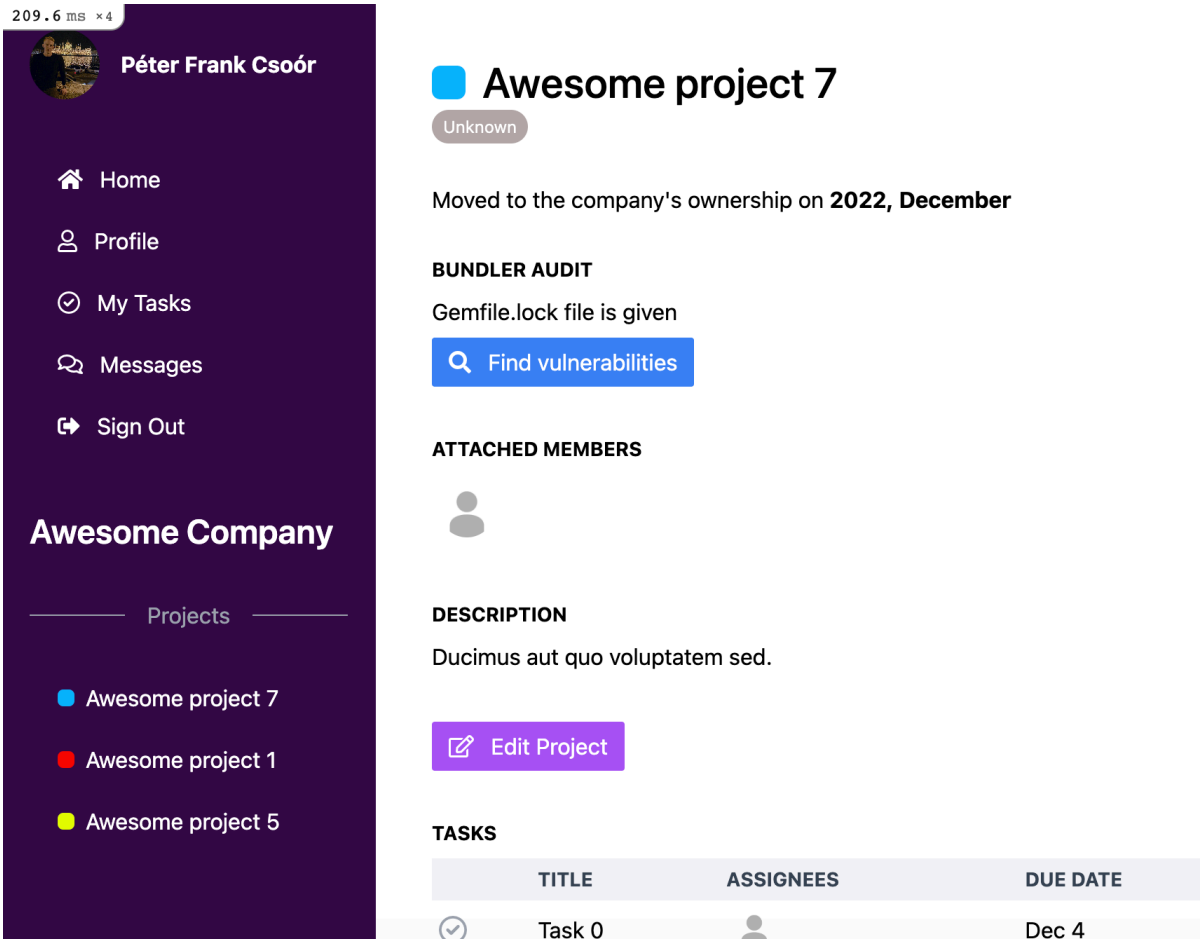
4. kép
Felhasználóhoz rendelt feladatainak oldala



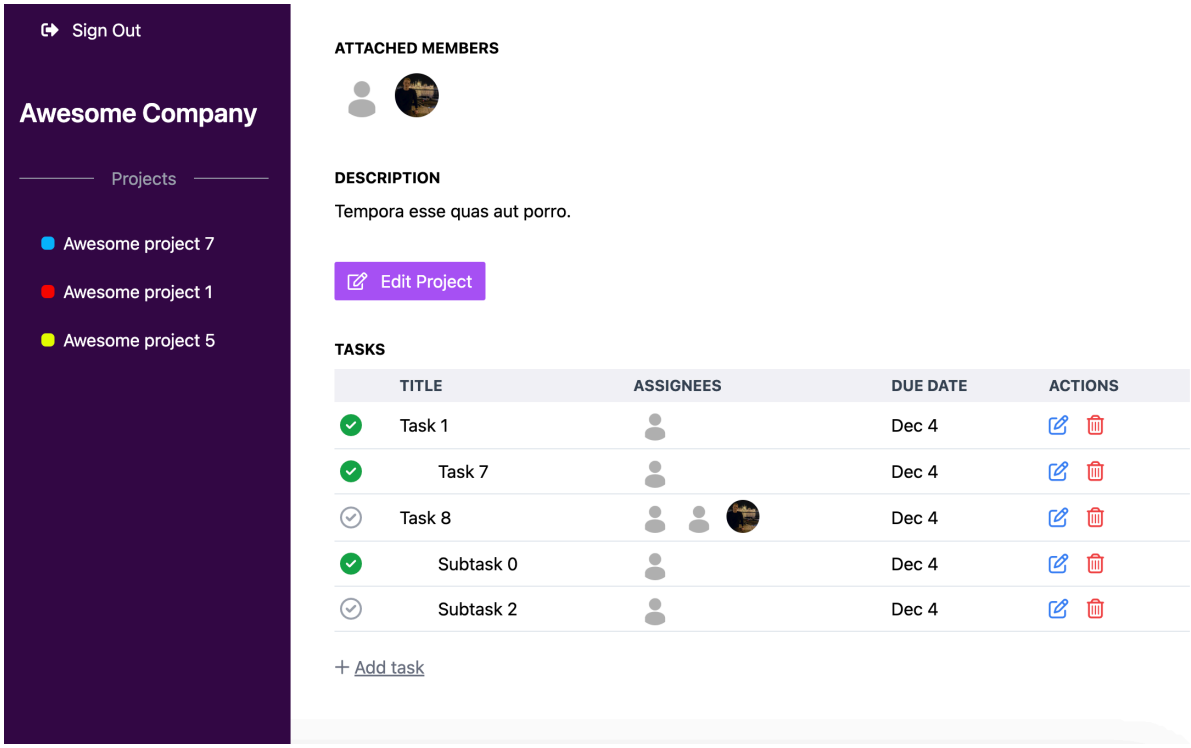
5. kép
Üzenetküldő felület



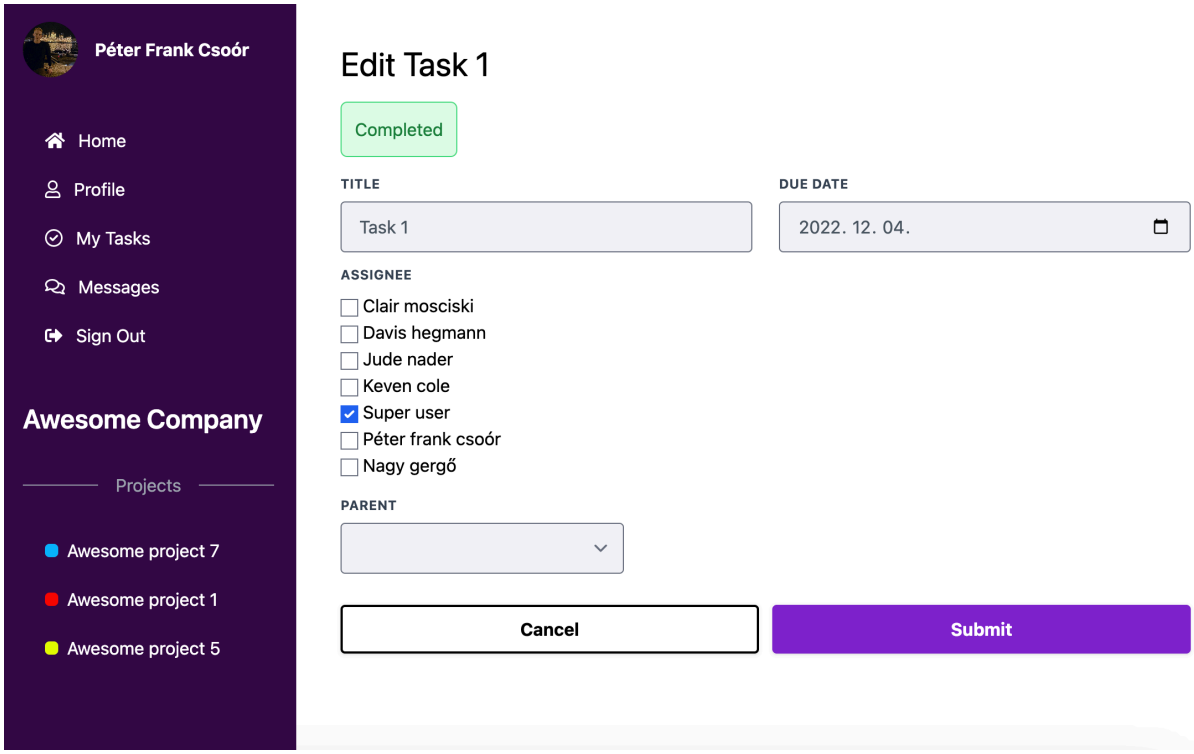
6. kép
Egy projekt show oldala



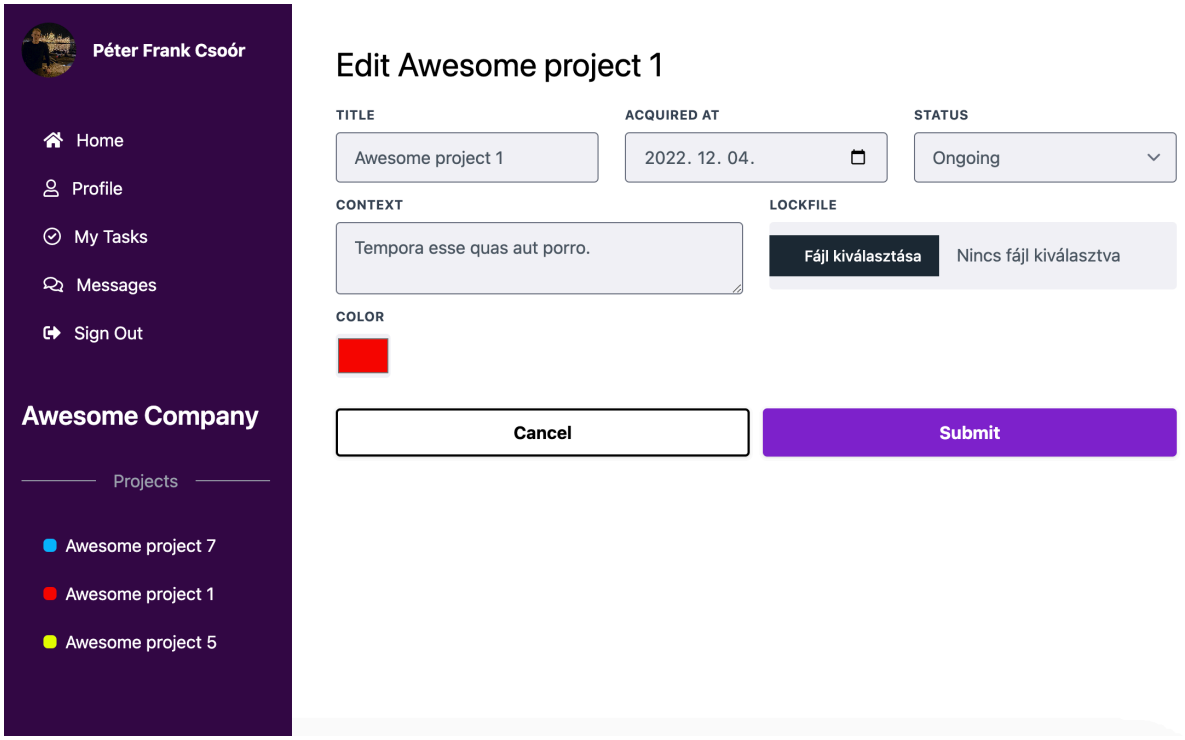
7. kép
Projekthez tartozó feladatok listája



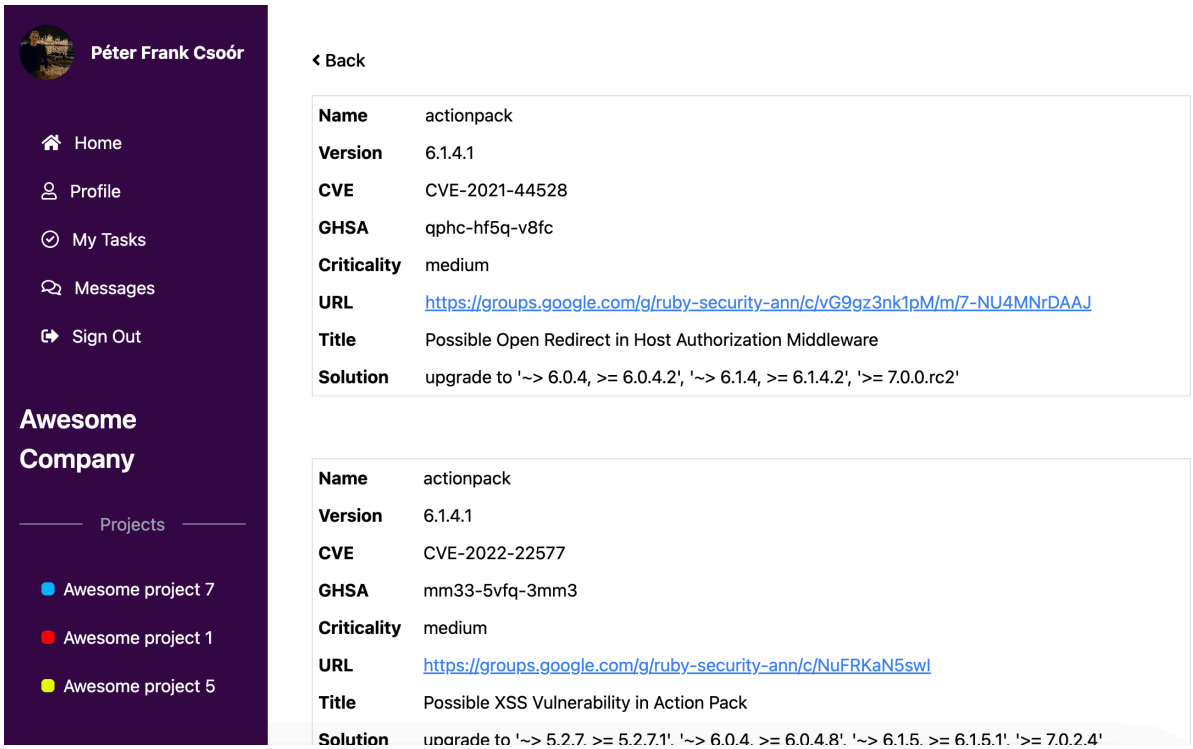
8. kép
Egy feladat szerkesztésének vagy létrehozásának oldala



9. kép
Egy projekt szerkesztésének vagy létrehozásának oldala



10. kép
A projekthez tartozó sérült gem-ek listájának oldala



9 FEJLESZTÉSI LEHETŐSÉGEK

Ahol lehetőséget látok a fejlesztésre azok az alábbi szegmensei az alkalmazásnak:

1. A sebezhetőségfeltáró eszköz dinamikusabbá tétele. Jelenleg ahhoz, hogy ez működni tudjon a felhasználónak kézzel fel kell tölteni a Gemfile.lock lockfile-t. Ez sokkal rugalmasabbá lehetne tenni, ha egy verziókezelőnek a repository-jára lenne ráirányítva és azt figyelné. Viszont ehhez be kell vonni egy kis automatizálást, ugyanis meg kell mondani a rendszernek, hogy mikor van az a pillanat, amikor újra lehúzza a tárolóból azt a fájlt (pl. minden hónap első napján).
2. Egy kezelhetőbb admin felület megvalósítása. Jelenleg a rendszer az active-admin gem által generált felületet használja, ami roppant jól működik egy alkalmazás elindítása esetén, amikor még kevés adat áll rendelkezésre. Továbbá, annak ellenére, hogy rengeteg hasznos dologgal fel lehet okosítani egy active-admin felületet, a lehetőségei korlátozottak. Éppen ezért, egy saját fejlesztésű adminisztrációs felület implementálása jobb megoldást jelenthet a jövőben.

10 ÖSSZEGZÉS

A dolgozatom célja egy olyan webalkalmazás létrehozása volt, mely segítséget nyújt IT cégek belsős csoportjainak a projektjeik és a feladataik menedzselésében és lehetőséget biztosítson az üzenetküldésre a dolgozók számára.

Az alkalmazásba épített egyik fő funkcionalitás egy monitorozó eszköz, mely a Ruby on Rails technológián alapuló programokhoz tartozó *Gemfile.loc*k fájlokat vizsgálja át és tárja fel a sebezhető gem-eket és tesz javaslatot egy olyan verzióra való váltásra, ahol a hibák javításra kerültek. Mindezek mellett lehetőség van a céges projekteket és az ezekhez tartozó feladatokat menedzselni, továbbá valós időben kommunikálni a dolgozókkal csoportos- és privát szobákban egyaránt.

Az alkalmazást a körülbelül egy hónapot felölelő kutatómunkám után két hónap alatt készítettem el, majd élesítettem.

11 SUMMARY

My task was to implement a web application that helps groups within an IT company to manage their projects and tasks and provide messaging for employees.

One of the main functionalities built into the application is a monitoring tool that examines *Gemfile.lock* files belonging to programs based on Ruby on Rails technology and reveals vulnerable gems and suggests moving to a version where the bugs have been fixed. In addition to all of this, it is possible to manage company projects and related tasks and to communicate with employees with real time in both group and private rooms.

After my research work of about one month, I created and deployed the application in two months.

12 IRODALOMJEGYZÉK

[1] I. Arghire, „Security Wekk,” 09 Május 2022. [Online]. Available: https://www.securityweek.com/rubygems-fixes-critical-gem-takeover-vulnerability. [Hozzáférés dátuma: 12. December 2022].

[2] Oracle, „Technical Articles - What is Ruby?,” [Online]. Available: https://developer.oracle.com/learn/technical-articles/what-is-ruby. [Hozzáférés dátuma: 06. június 2022].

[3] invozone, „Benefits of using Ruby on Rails for your startup,” [Online]. Available: https://invozone.com/blog/benefits-of-ruby-on-rails/. [Hozzáférés dátuma: 12. December 2022].

[4] TK, Artist, *MVC architektúra.* [Art].

[5] AWS, „What is PostgreSQL,” [Online]. Available: https://aws.amazon.com/rds/postgresql/what-is-postgresql/. [Hozzáférés dátuma: 06. Június 2022].

[6] Wikipedia, „Facebook,” [Online]. Available: https://hu.wikipedia.org/wiki/Facebook. [Hozzáférés dátuma: 24. November 2022].

[7] Heartcombo, „GitHub Devise repository,” [Online]. Available: https://github.com/heartcombo/devise. [Hozzáférés dátuma: 18. November 2022].

[8] Hotwire, „Hotwire,” [Online]. Available: https://hotwired.dev/. [Hozzáférés dátuma: 15. November 2022].

[9] Hotwired, „Turbo,” [Online]. Available: https://turbo.hotwired.dev/handbook/introduction. [Hozzáférés dátuma: 16. November 2022].

[10] R. o. Rails, Artist, *Rails RESTful routes.* [Art].

[11] BigBinary, „Secure gems with bundle audit,” [Online]. Available:

https://www.bigbinary.com/books/learn-rubyonrails-book/bundler-audit. [Hozzáférés dátuma: 16. November 2022].

[12] Rubysec, „Bundler audit documentation,” [Online]. Available: https://rubydoc.info/gems/bundler-audit/. [Hozzáférés dátuma: 09. December 2022].

13 MELLÉKLETEK