SZÉCHENYI EGYETEM
UNIVERSITY OF GYŐR

INFORMATIKA TANSZÉK
DEPARTMENT OF INFORMATICS

Egyedi vállalatra szabott bérléskezelő rendszer fejlesztése

Székely Dániel

Mérnökinformatikus MSc

2025

Tartalomjegyzék

1. fejezet

Bevezetés

A mai vállalati világban az idő és az erőforrások hatékony kezelése nem csupán optimális, hanem létfontosságú versenyelőny. A hagyományos, papíralapú vagy szigetszerűen kezelt folyamatok lassúak, átláthatatlanok és hibákra hajlamosak. A vállalatoknak ezért exponenciálisan igénye van olyan rendszerekre, amelyek gyorsítják a munkafolyamatokat, csökkentik a hibalehetőségeket és automatizálják a mindennapi működést.

Ebben a környezetben vállaltam egy önálló projektet a vállalatnak, ahol már évek óta tevékenykedem. Egy **egyedi bérléskezelő rendszer (Rental Management System – RMS)** fejlesztését, amely a vállalat minden projektéhez kapcsolódó folyamatait digitális formába önti. A cél nem csupán a szoftver létrehozása volt, hanem annak teljes életciklusú menedzselése, a tervezéstől és ütemezéstől kezdve a kockázatok azonosításán át a megvalósításig és az átadásig és karbantartásig.

A dolgozat célja a projekt részletes bemutatása: ismertetem az RMS céljait, funkcióit és technológiai hátterét. Szakmai szempontból külön hangsúlyt kapnak a projektmenedzsment folyamatok, különösen a **tervezés**, az **ütemezés** és a **kockázatkezelés**, valamint annak bemutatása, hogy ezek miként járultak hozzá a projekt sikeres megvalósításához.

A dolgozat többek között a következő kérdésekre is választ keres:

- Hogyan tervezhető és menedzselhető hatékonyan egy egyedi vállalati szoftverfejlesztési projekt?
- Milyen módszerek és eszközök biztosítják az erőforrások optimális felhasználását és a kockázatok minimalizálását?
- Milyen szakmai tanulságok vonhatók le az önálló projektmenedzsment gyakorlatából, amelyek más projektekben is alkalmazhatók?

A következő fejezetben bemutatom a projekt konkrét célkitűzéseit és főbb jellemzőit, kiemelve, hogy a fejlesztett rendszer milyen problémákat old meg és milyen értéket teremt a vállalat számára.

1.1. ábra. RMS kreatív ábra

2. fejezet

Projekt áttekintése és részletes bemutatása

2.1. A vállalat és az üzleti igények

A projekt célja a **TéDé Rendezvények** vállalat bérlés és projektkezelési folyamatainak teljes digitalizálása volt. A korábbi papíralapú és Excel-alapú folyamatok nem feleltek meg a vállalat növekvő igényeinek, és egy modern, integrált rendszer kialakítása vált szükségessé. Az online elérhető üzenetküldő és felhő alapú megoldások nem kínáltak megfelelő rugalmasságot és testreszabhatóságot, ezért egy egyedi fejlesztésű rendszer mellett született döntés.

A vállalat számára kiemelten fontos volt egy egységes, modern és felhasználóbarát rendszer, amely:

- automatizálja a bérlési folyamatokat,
- nyomon követi a készleteket,
- biztosítja az adminisztráció teljes körű kezelését,
- egyszerűsíti a munkafolyamatokat,
- csökkenti a hibalehetőségeket,
- javítja a kommunikációt a csapaton belül,
- egységesíti a dokumentációt,
- minden információ egy helyen elérhető,
- lehetővé teszi a gyors árajánlatkészítést.

2.2. A projekt célkitűzései

További célok voltak:

- modern, felhasználóbarát felület
- egyszerűen telepíthető és skálázható **Docker** segítségével
- könnyen karbantartható és bővíthető architektúra
- biztonságos hozzáférés-kezelés és jogosultságok
- részletes riportálási és statisztikai funkciók

2.3. A projekt összetettsége és kihívásai

A projekt felettébb összetettnek tekinthető, mivel:

- többféle felhasználói szerepkört kellett kezelnie (admin, munkatárs, menedzser),
- integrálni kellett különböző adatforrásokat és a bérlési folyamatokat,
- biztonsági és hozzáférés-kezelési követelményeknek kellett megfelelnie,
- a fejlesztés során több technológiát kellett összehangolni a rugalmasság és megbízhatóság érdekében.

2.4. Technológiai háttér

A rendszer fejlesztése a következő technológiákra épült:

- **Backend:** PHP a Twig sablonmotorral,
- **Frontend:** modern responsive felhasználói felület Twig sablonokkal,
- **Konténerizálás és telepítés:** Docker és Docker Compose, előre elkészített konténerekben,
- **Konfiguráció:** testreszabható `.env` fájlok segítségével.
- **Reverse proxy:** Nginx a kérések kezelésére és a statikus fájlok kiszolgálására,
- **Adatbázis:** MySQL a megbízható adatkezelés érdekében,

- **Verziókezelés:** Git a kód nyomon követésére.

Ez a technológiai kombináció biztosítja a rendszer gyors telepítését, stabil működését és könnyű bővíthetőségét.

2.5. A megvalósított szoftver

2.5.1. Bejelentkezés

A Bejelentkezési felület lehetővé teszi a felhasználók számára, hogy biztonságosan hozzáférjenek a rendszerhez. Ezt megtehetik helyi fiókkal vagy Google OAuth2 hitelesítéssel, amely egyszerűsíti a bejelentkezési folyamatot és növeli a biztonságot. A bejelentkezés után amennyiben új felhasználó lép be a rendszerbe, egy egyszeri csatlkozó jelszó megadására van szükség amivel kapcsolódik a vállalathoz, ezzel biztosítva, hogy csak jogosult személyek férjenek hozzá a rendszerhez, és minden tevékenység naplózásra kerül a későbbi ellenőrzés érdekében.

2.1. ábra. Bejelentkezés

2.5.2. Dashboard

Az alkalmazás indítása után az alapértelmezett nézet a **Dashboard**, amely a rendszer központi irányítópultjaként szolgál. Célja, hogy a felhasználó egyetlen felületen, átlátható formában kapjon átfogó képet a vállalkozás aktuális működéséről, az eszközállományról, a projektek státuszáról és a saját napi tevékenységeiről.

A felület moduláris felépítésű, azaz minden információ önálló panelen jelenik meg, így a felhasználó gyorsan hozzáférhet az őt érdeklő adatokhoz anélkül, hogy menük között kellene navigálnia. A legfontosabb panelek a következők:

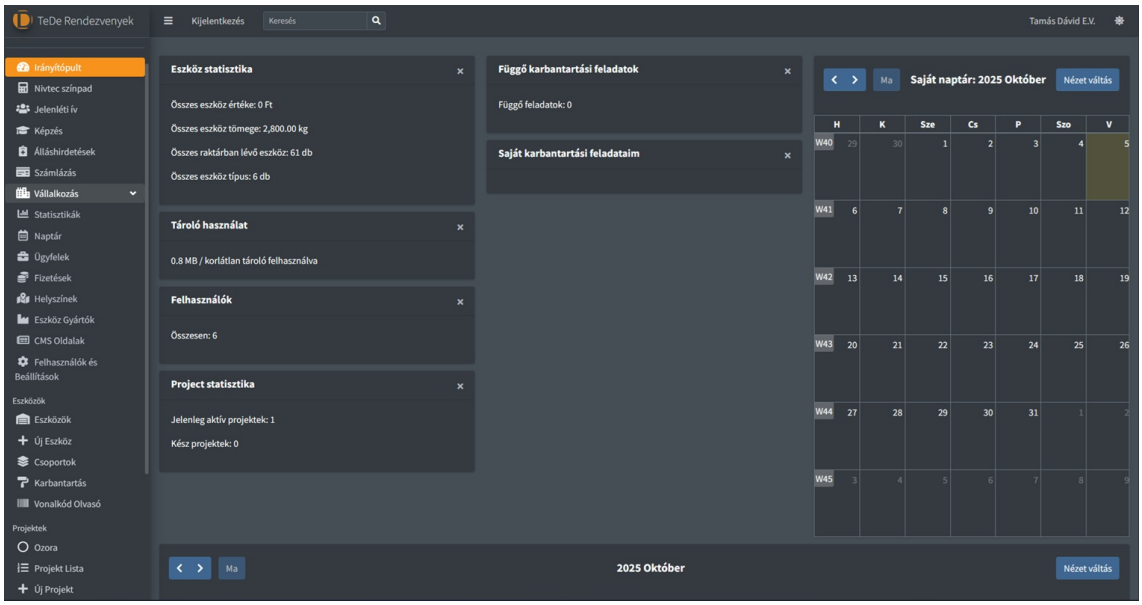
- **Eszköz statisztika:** összesített információkat jelenít meg az aktuális eszközpark állapotáról, beleértve az összértéket, az össztömeget, a raktárban lévő mennyiséget és az eszköztípusok számát. Ez különösen hasznos a logisztikai és bérléskezelési döntések előkészítésénél.
- **Tároló használat:** kijelzi a tárhely aktuális kihasználtságát, ami segíti a rendszeradminisztrátort a tárolókapacitás hatékony felügyeletében, különösen a feltöltött képek és dokumentumok kezelésénél.
- **Felhasználók:** megjeleníti az aktív felhasználók számát, ezzel is elősegítve a jogosultságkezelést és a hozzáférések ellenőrzését.
- **Projekt statisztika:** az éppen aktív és a lezárt projektek számát mutatja, ami egyértelmű képet ad a vállalat aktuális munkaterheléséről és projektfázisairól.
- **Karbantartási feladatok:** listázza a függő karbantartási műveleteket, illetve a bejelentkezett felhasználó saját feladatait, ezzel biztosítva, hogy a rendszerüzemeltetés és az eszközkarbantartás ne maradjon el.

A jobb oldalon található **naptárnézet** a Dashboard egyik kulcsfunkciója, amely a tervezés és szervezés hatékonyságát támogatja. A naptár kétféle módon használható:

- **Vállalati nézet:** az összes eseményt, projektmérföldkövet és karbantartási ütemezést megjeleníti, így a vezetők és adminisztrátorok teljes képet kapnak a szervezet aktuális feladatairól.
- **Személyes nézet:** kizárólag az adott felhasználóhoz rendelt eseményeket és határidőket mutatja, ezzel segítve az egyéni időbeosztás és feladatprioritás kezelését.

A naptár interaktív: a felhasználók közvetlenül a felületen lépkedhetnek a hetek és hónapok között, illetve eseményekhez kapcsolódó részleteket is megtekinthetnek. Ez a funkció jelentősen növeli a munkaszervezés hatékonyságát, mivel vizuális formában támogatja a tervezést és a határidők követését.

A Dashboard tehát nem csupán egy információs felület, hanem a rendszer működésének központi irányítási pontja, amely valós időben tükrözi a vállalkozás operatív állapotát. Felépítése reszponzív, így asztali és mobil eszközön egyaránt optimálisan használható, miközben a modulok dinamikusan frissülnek a háttérben futó adatbázis-lekérdezések alapján.



2.2. ábra. Dashboard – vállalati áttekintő felület

2.5.3. Eszközök kezelése

Az **Eszközök** modul a rendszer egyik legfontosabb eleme, hiszen itt történik a teljes készlet nyilvántartása, kezelése és karbantartása. A felület célja, hogy a vállalat munkatársai gyorsan és átlátható módon kezelhessék a több száz, esetenként több ezer eszközből álló technikai parkot, valós idejű szűrési és rendezési lehetőségek mellett.

A rendszerben minden eszköz önálló entitásként szerepel, amelyhez részletes metaadatok és kapcsolódó információk tartoznak. Az eszközök az alábbi attribútumokkal rendelkeznek:

- **Név és kategória:** az eszköz megnevezése és típusbesorolása (pl. hangtechnika, fénytechnika, videótechnika).
- **Címke és egyedi azonosító:** egyedi vonalkódos címkével vagy belső ID-val azonosítható minden eszköz, ami megkönnyíti a gyors leltározást és az eszközmozgások követését.
- **Gyártó és modell:** a pontos technikai paraméterezés és kompatibilitás érdekében.
- **Sorozatszám és állapot:** minden eszköz karbantartási vagy garanciális nyilvántartásának alapja.
- **Leírás és karbantartási információk:** részletes technikai leírás, valamint az eszköz állapotának és karbantartási ciklusainak dokumentálása.
- **Mennyiség és bérlési árak:** az aktuális készlet és a bérlésre vonatkozó díjszabás nyilvántartása.

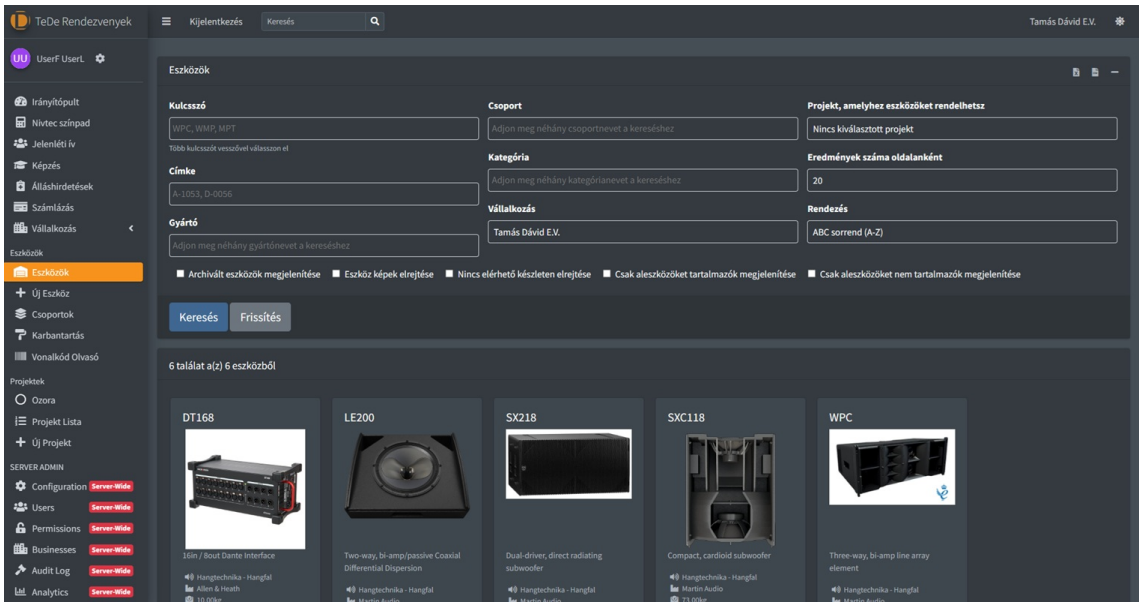
- **Kép:** vizuális azonosítást segítő fotó, amely a felhasználói élményt és a hibamentes választást támogatja.

A felület fejlett szűrési és keresési funkciókat biztosít, így a felhasználók pillanatok alatt megtalálhatják a keresett eszközt akár több száz tétel között is. A szűrési lehetőségek a következők:

- **Kulcsszó alapú keresés:** akár részleges egyezés alapján is szűr a név, címke vagy gyártó szerint.
- **Csoport és kategória szerinti szűrés:** lehetővé teszi, hogy csak adott projekttípushoz vagy technikai kategóriához tartozó eszközök jelenjenek meg.
- **Projekt szerinti rendezés:** megjeleníthetők kizárólag egy adott eseményhez, produkcióhoz vagy ügyfélhez rendelt eszközök.
- **Vállalkozás szűrő:** a különböző alvállalkozókhoz tartozó eszközök elkülönített kezelése.
- **Előre beállított nézetek:** lehetőség van archivált eszközök megjelenítésére, képek elrejtésére vagy a készleten nem elérhető tételek kiszűrésére.
- **Rendezési opciók:** ABC sorrend, gyártó, kategória vagy elérhetőség alapján történő megjelenítés.
- **Speciális szűrők:** csak aleszközöket (pl. kábelek, tartozékok) vagy csak főeszközöket tartalmazó lista megjelenítése.

A keresési rendszer valós időben frissül, így a felhasználó azonnal látja az eredményeket, miközben a rendszer automatikusan optimalizálja a lekérdezéseket a szerver teljesítményének megőrzése érdekében. Az eszközlista kártyás formában jelenik meg, vizuálisan elkülönítve a különböző típusokat és státuszokat, így a kezelő személyzet gyorsan áttekintheti a rendelkezésre álló készletet.

Az **Eszközök** modul tehát nem csupán egy adatbázis-kezelő felület, hanem egy interaktív, rugalmas eszközmenedzsment-rendszer, amely az operatív döntéshozatalt és a napi logisztikai munkát egyaránt támogatja.



2.3. ábra. Eszközök kezelése

2.5.4. Projektek kezelése

A rendszer egyik legfontosabb modulja a **Projektek kezelése**, amely a vállalkozás teljes operatív működését átfogja. Ebben a nézetben a felhasználó részletesen konfigurálhatja és nyomon követheti az egyes eseményekhez, rendezvényekhez vagy megrendelésekhez tartozó projekteket. A cél, hogy minden információ, az ügyféltől kezdve az eszközökön és helyszíneken át egészen a csapatbeosztásig, egy központi felületen legyen kezelhető.

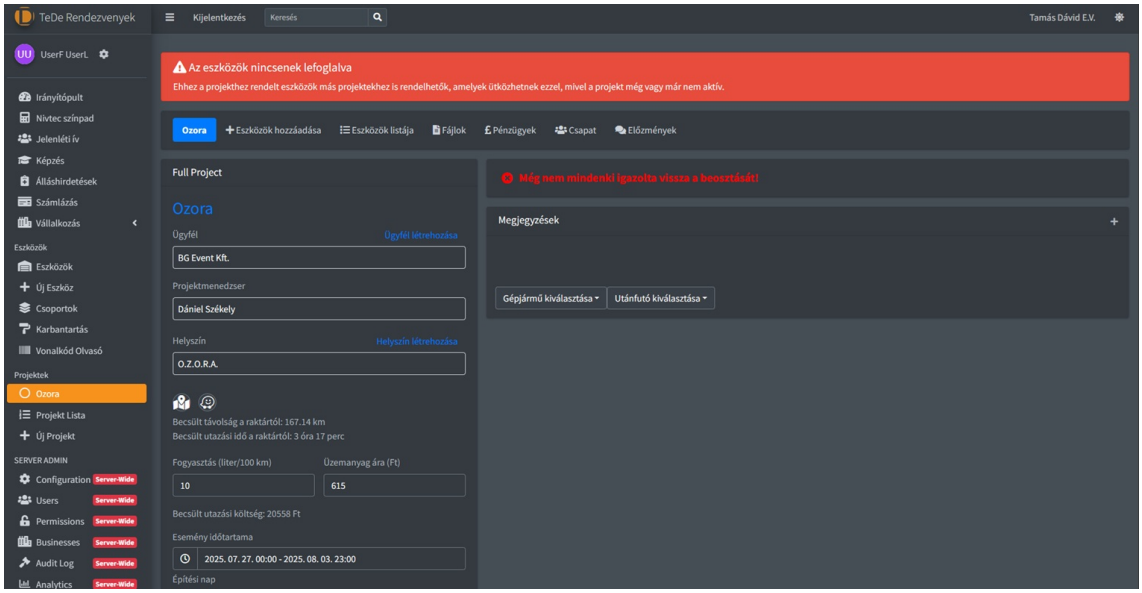
A projekt adatlapja több logikai szekcióra tagolódik:

- **Ügyfélkezelés:** a projekthez tartozó ügyféladatokat jeleníti meg, illetve új ügyfél is létrehozható közvetlenül innen. Ez leegyszerűsíti a megrendelések adminisztrációját és a CRM-folyamatok integrációját.
- **Projektmenedzser és felelősök:** minden projekthez hozzárendelhető egy vagy több felelős személy, akik a feladatok koordinálásáért és a végrehajtás felügyeletéért felelősek. A rendszer automatikus értesítéseket küld, ha egy új projektet hoznak létre vagy módosítanak.
- **Helyszínkezelés:** a rendezvény vagy telepítés pontos helyszínét tartalmazza. Az adatbázis képes korábbi helyszínek újrafelhasználására, így a gyakran ismétlődő helyszínek (pl. fesztiválok, partnerrendezvények) gyorsan kiválaszthatók.
- **Logisztikai kalkuláció:** a rendszer automatikusan kiszámítja a raktártól való távolságot, az utazási időt, az üzemanyag-felhasználást és a várható utazási költséget. Ez

a funkció különösen fontos a terepi projektek (például rendezvényhelyszínek) előkészítése során, mivel a logisztikai tervezés a költségvetés egyik kritikusabb eleme.

- **Erőforrás-hozzárendelés:** a projektfelülethez közvetlenül csatolhatók eszközök, járművek és utánfutók. A rendszer automatikusan figyelmeztet, ha egy eszköz már másik projekthez van rendelve, vagy ha a projekt inaktív, így elkerülhetők az ütközések és a duplikációk.

- **Eseményidőtartam és ütemezés:** a projekt időtartamát kezdő- és záródátum alapján rögzíti a rendszer, amely később összekapcsolódik a naptármodullal. A felhasználók így a teljes munkafolyamatot, az előkészítéstől az építési napig, kronológiai sorrendben követhetik.



2.4. ábra. Projekt részletes adatlap amit az adminok és menedzserek látnak

A projektekhez további funkciók is társulnak, például fájlkezelés (dokumentumok, szerződések, műszaki rajzok feltöltése), pénzügyi modul (költségvetés és elszámolás), valamint **csapatkezelés**, amelyben a rendszer minden résztvevőt listáz és státuszukat (jóváhagyott / függő) megjeleníti. Ezáltal a projektmenedzser azonnal látja, ha valaki még nem erősítette meg a beosztását, ezt vizuálisan is jelzi a felület figyelmeztető sávja.

A **Projektek kezelése** modul így nem csupán adminisztrációs célokat szolgál, hanem valódi menedzsmenteszközzé válik, amely integrálja a logisztikai, humán és technikai folyamatokat. Ezzel biztosítja, hogy minden esemény előkészítése és lebonyolítása transzparensen, ellenőrzötten és hatékonyan történjen.

A rendszer minden egyes módosítást naplóz, így a projekt életciklusa teljes egészében visszakövethető, ki mikor és milyen változtatást hajtott végre. Ez különösen fontos a

későbbi auditok és elemzések szempontjából, valamint amennyiben vitás helyzetek merülnének fel a projekt végrehajtása során, hiszen minden lépés dokumentált és ellenőrizhető marad.

2.5. ábra. Projekt kompakt adatlap amit a munkavállalók látnak

2.5.5. GitHub és Reverse Proxy

A verziókezelés és a biztonságos hálózati hozzáférés is kiemelt szerepet kapott a projekt során. A projekt forráskódja a **GitHub** platformon található, amely lehetővé teszi a verziók követését, a biztonsági mentések készítését és a folyamatos fejlesztést. A GitHub használata növeli a fejlesztési folyamat átláthatóságát, támogatja a csapatmunkát, és biztosítja a kód minőségének megőrzését.

A fejlesztés egy főágon (`main branch`) zajlik, ehhez a lokális környezetből `git push/pull` műveletekkel történik a szinkronizálás. Minden új funkció fejlesztése külön fejlesztői ágon (`feature branch`) történik, ezáltal a stabil főverzióba csak tesztelt és hibamentes kód kerülhet. A commit-ok egységes elnevezési sémát követnek (pl. `feat:`, `fix:`, `refactor:`), ami megkönnyíti a változások visszakövetését és a hibakeresést. A verziókezelés így nem csupán biztonsági, hanem projektmenedzsment-eszközként is működik.

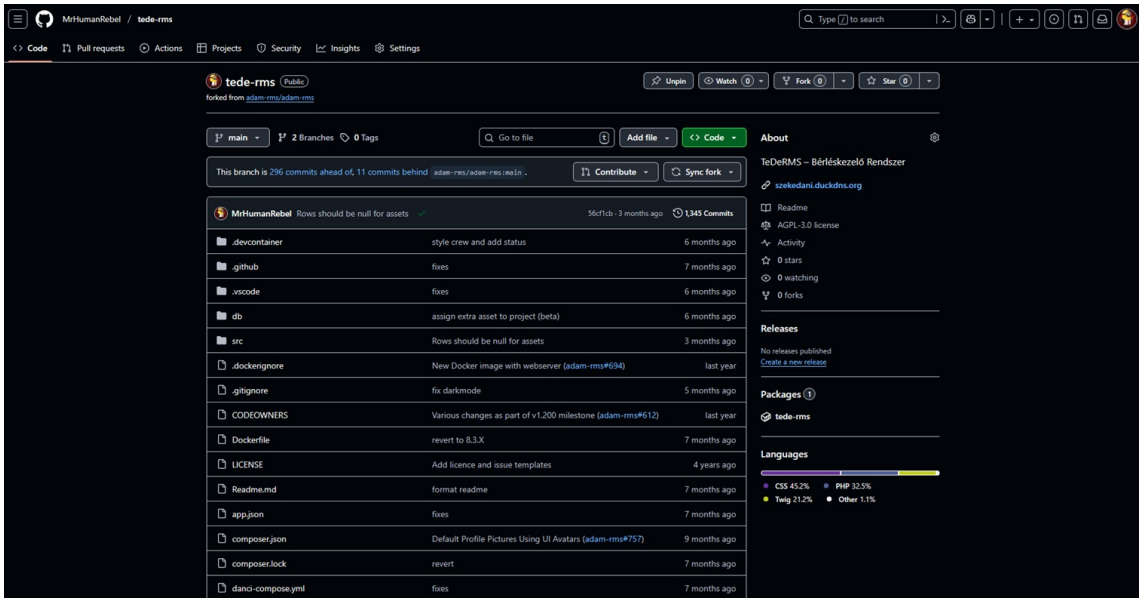
A fejlesztési környezet a **Docker Compose** infrastruktúrán alapul, amely biztosítja a konténerek gyors indítását, újratelepítését és frissítését. A rendszer több szolgáltatásból áll — például webalkalmazás, adatbázis és proxy réteg — amelyek egymással biztonságosan kommunikálnak a konténerhálón belül. A külső hálózat felé a kapcsolatot egy **Reverse Proxy** réteg biztosítja, amely a beérkező HTTP(S) kéréseket a megfelelő szolgáltatáshoz irányítja.

A reverse proxy réteg **NGINX** alapokon működik, amely:

- biztosítja a HTTPS titkosítást és a tanúsítványkezelést (`Let's Encrypt`);
- lehetővé teszi az aldomének és portok dinamikus irányítását;
- csökkenti a külső támadások kockázatát az alkalmazásrétegek elszigetelésével;
- elősegíti a terheléselosztást, ezáltal növeli a rendszer válaszképességét és stabilitását.

A proxy réteg nemcsak hálózati védelmet nyújt, hanem támogatja a fejlesztés és üzemeltetés szétválasztását is. A GitHub repository tartalmazza a `docker-compose.yml` és `nginx.conf` fájlokat, amelyek lehetővé teszik a rendszer gyors telepítését bármely környezetben — fejlesztői, teszt vagy éles szerveren — egyetlen parancs segítségével.

Ez az architektúra a modern **DevOps-elvek** alapján működik, ahol a fejlesztés, verziókezelés, biztonság és üzemeltetés integrált egységet alkot. A GitHub–Docker–NGINX kombináció garantálja a kód stabilitását, az infrastruktúra rugalmasságát és az adatbiztonságot, ami alapvető követelmény egy vállalati szintű rendszer esetében.



2.6. ábra. GitHub repository

2.5.6. Kód részletek

Code Blame 66 lines (55 loc) · 2.27 KB

```
{% block content %}
  <div class="container">
    {% if USERDATA.users_dark_mode %}
      <link rel="stylesheet" href="dark-style.css">
      <a href="{{ CONFIG.ROOTURL }}/">
        <img src="{{ CONFIG.ROOTURL }}/static-assets/img/logo_trasparent_dark.png" alt="logo" class="brand-image img-circle elevation-3">
      </a>
    {% else %}
      <link rel="stylesheet" href="style.css">
      <a href="{{ CONFIG.ROOTURL }}/">
        <img src="{{ CONFIG.ROOTURL }}/static-assets/img/logo_trasparent.png" alt="logo" class="brand-image img-circle elevation-3">
      </a>
    {% endif %}

    <h1>{{ title }}</h1>

    <div class="input-group">
      <label for="length">Színpad szélessége (méterben):</label>
      <input type="number" id="length" placeholder="Például: 6">
      <div id="length-error" class="error-message">Kérlek adj meg érvényes szélességet!</div>
    </div>

    <div class="input-group">
      <label for="width">Színpad mélysége (méterben):</label>
      <input type="number" id="width" placeholder="Például: 4">
      <div id="width-error" class="error-message">Kérlek adj meg érvényes mélységet!</div>
    </div>

    <div class="rotate-group">
        <label for="rotate-checkbox">
            Forgatás
            <input type="checkbox" id="rotate-checkbox">
        </label>
    </div>
```

2.7. ábra. Kód részlet - Nivtec kalkulátor integráció

Code Blame 59 lines (56 loc) · 1.26 KB

```
services:
  db:
    image: index.docker.io/mysql/mysql-server:8.0
    command: --default-authentication-plugin=mysql_native_password --innodb-thread-concurrency=0 --sort_buffer_size=512K
    container_name: db
    volumes:
      - ./db_data:/var/lib/mysql
      - /etc/localtime:/etc/localtime:ro
    restart: always
    environment:
      - MYSQL_DATABASE=TeDeRMS
      - MYSQL_ROOT_HOST=%
      - MYSQL_USER=userDocker
      - MYSQL_PASSWORD=TeDeDocker
    env_file:
      - .env
    healthcheck:
      test:
        [
          "CMD",
          "mysqladmin",
          "ping",
          "-h",
          "localhost",
          "-u",
          "root",
          "-p$$MYSQL_ROOT_PASSWORD",
        ]
      timeout: 20s
      retries: 10

  s3ninja:
    image: scireum/s3-ninja:latest
    container_name: s3ninja
    ports:
```

2.8. ábra. Kód részlet - Docker Compose fájl

3. fejezet

Projekt életciklusa

A projektmenedzsment egyik legfontosabb alapelve, hogy minden projektnek megvan a maga életciklusa, amely jól elkülöníthető fázisokra bontható. Ezek a fázisok nemcsak logikai, hanem szervezeti és irányítási szempontból is meghatározóak, hiszen lehetővé teszik a projekt strukturált tervezését, nyomon követését és értékelését. A hazai szakirodalom egyaránt öt alapvető fázist különít el [1, 2]:

1. **Projektindítás (Initiation)** a projekt céljainak, indokoltságának és alapvető paramétereinek meghatározása.
2. **Tervezés (Planning)** az ütemezés, erőforrások, kockázatok és feladatok részletes kidolgozása.
3. **Megvalósítás (Execution)** a tényleges fejlesztési és kivitelezési folyamatok végrehajtása.
4. **Ellenőrzés és irányítás (Monitoring & Controlling)** a projekt előrehaladásának, költségeinek és minőségének nyomon követése.
5. **Lezárás (Closure)** a projekt hivatalos befejezése, átadás és értékelés.

3.1. ábra. Projekt kreatív vizualizáció

A projektciklus fázisai egymásra épülnek, ugyanakkor gyakran átfedésben is lehetnek: az ellenőrzési és irányítási folyamat például a megvalósítás teljes ideje alatt folyamatosan zajlik. A modern megközelítések különösen az iteratív és ismétlődő fejlesztési modellek nem mindig lineáris struktúrát követnek, hanem sprint vagy iteráció formájában valósítják meg a fejlesztést.

Ennek ellenére a klasszikus projektciklus továbbra is nélkülözhetetlen a stratégiai és vállalati projektekben, mivel jól követhető keretet biztosít az egész folyamat számára.

A **TeDeRMS** fejlesztésénél egy klasszikus, ötfázisú modell szolgált alapul, melyet a projekt egyedi körülményeihez önálló fejlesztés, korlátozott erőforrások, vállalati integráció igazítottam. Az alábbiakban részletesen bemutatom, hogyan valósult meg a projekt életciklusa a gyakorlatban.

3.1. Projektindítás (Initiation)

A projektindítás során a következő lépések történtek:

- **Igényfelmérés:** a vállalat bérléskezelési folyamatait elemeztem, hogy pontosan feltérképezzem a fejlesztendő rendszer funkcióit és a problémás területeket.
- **Célok meghatározása:** világos, mérhető célkitűzéseket rögzítettem, például az adminisztrációs idő csökkentését, a hibák minimalizálását és az automatizált riportok bevezetését.
- **Erőforrás-tervezés előkészítése:** meghatároztam a projekthez szükséges technológiai és időbeli erőforrásokat. Mivel a fejlesztést önállóan végeztem, kiemelten fontos volt a prioritások és a munkaidő hatékony beosztása.

Ez a fázis biztosította, hogy a projekt kiindulópontja egyértelmű legyen, és a fejlesztés a vállalat igényeinek megfelelően induljon el. Amennyiben a projektindítás nem lett volna alapos, a későbbi fázisokban jelentős problémák merülhettek volna fel, például félreértett követelmények esetében a fejlesztés nem a kívánt irányba haladt volna ezzel erőforrásokat és időt pazarolva.

3.2. Tervezés (Planning)

A tervezési fázisban a projekt sikerének záloga a részletes ütemezés és a kockázatok előrejelzése volt:

- **Ütemterv készítése:** a projekt főbb mérföldköveit és feladatait időrendi sorrendbe állítottam, így követhetővé vált a fejlesztés előrehaladása.

- **Erőforrás-tervezés:** részletesen meghatároztam az idő- és technológiai erőforrásokat, valamint a napi munkabeosztást, hogy az önálló fejlesztés gördülékeny legyen.
- **Kockázatelemzés:** azonosítottam a legfontosabb kockázatokat (pl. technikai hibák, adatvesztés, hibás üzleti logika), és kidolgoztam a megelőző és elhárító intézkedéseket.

A tervezés során alkalmazott módszerek: feladatlista, ütemezett mérföldkövek, kockázatmátrix és rendszeres önellenőrzés.

3.3. Megvalósítás (Execution)

A megvalósítás során a projekt tényleges fejlesztési munkája zajlott:

- **Fejlesztési folyamatok:** moduláris felépítésben, backend és frontend párhuzamos fejlesztése, verziókövetés GitHub-on.
- **Tesztelés:** folyamatos unit és funkcionális tesztek, hogy a rendszer stabil és hibamentes legyen.
- **Dokumentáció:** a kód és a rendszer működésének részletes dokumentálása, hogy a későbbi karbantartás és bővítés egyszerű legyen.

Ez a fázis biztosította, hogy a rendszer minden funkciója a terv szerint valósuljon meg, és a vállalat igényei teljesüljenek.

3.4. Ellenőrzés és irányítás (Monitoring és Controlling)

A projekt előrehaladásának nyomon követése kritikus volt a siker szempontjából:

- **Haladás nyomon követése:** a mérföldkövek teljesülésének ellenőrzése, eltérések azonosítása és korrekciója.
- **Kockázatok kezelése:** a kockázatmátrix folyamatos frissítése, a problémák gyors azonosítása és megoldása.
- **Minőségellenőrzés:** a rendszer funkcionalitásának és stabilitásának folyamatos ellenőrzése a hibák minimalizálása érdekében.

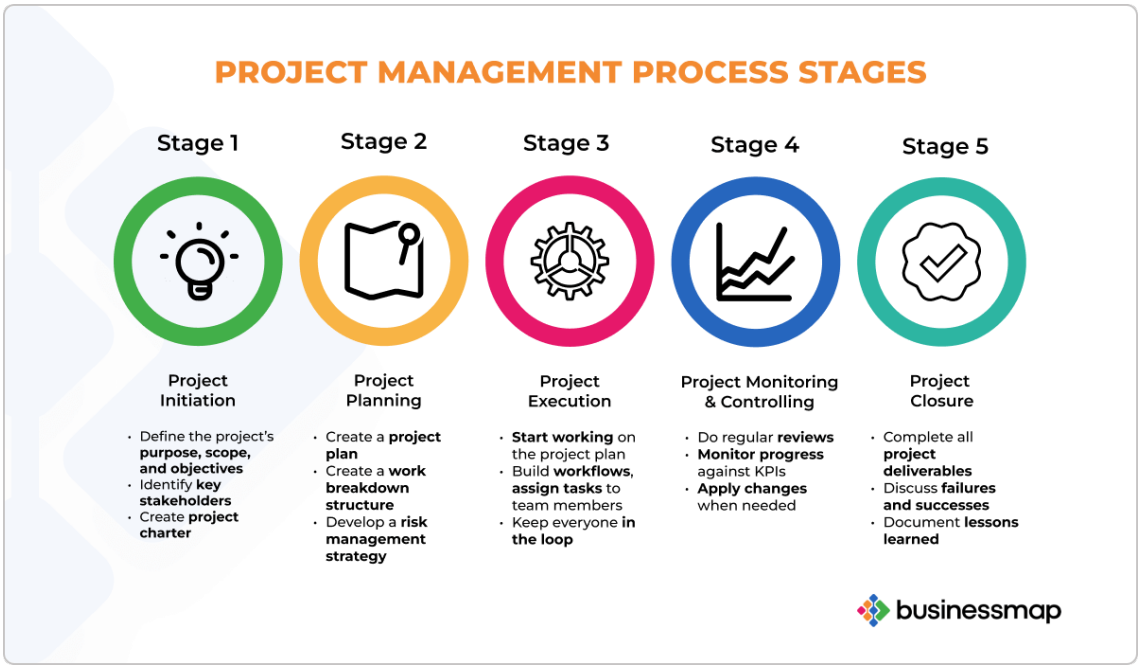
Ez a fázis biztosította, hogy a projekt ne csússzon ki a tervezett keretek közül, és minden mérföldkő a kívánt minőségben valósuljon meg.

3.5. Projekt lezárás (Closure)

A projekt lezárásakor a következő tevékenységek történtek:

- **Rendszer átadása:** a **TeDeRMS** teljes körű telepítése a vállalat környezetében.
- **Dokumentáció:** részletes használati útmutatók és adminisztrátori kézikönyvek készítése.
- **Oktatás:** a kulcsfelhasználók képzése a rendszer hatékony használatára.
- **Tanulságok összegzése:** a projekt során szerzett tapasztalatok dokumentálása, javaslatok a jövőbeli bővítésekhez.

Ez a fázis biztosította, hogy a rendszer hosszú távon stabilan és hatékonyan működjön, miközben a projektmenedzsment folyamatok tanulságai később is felhasználhatók.



3.2. ábra. Projektmenedzsment folyamatok fázisai

4. fejezet

Kockázatkezelés és problémamegoldás

A **TeDeRMS** projekt során a kockázatkezelés kulcsfontosságú szerepet játszott, mivel a projekt önálló fejlesztésként valósult meg, így minden döntés és probléma gyors és hatékony kezelést igényelt. A kockázatkezelés célja a potenciális problémák előrejelzése, azok hatásának minimalizálása, valamint a projekt sikerének biztosítása volt.

4.1. Felmerült problémák

A fejlesztés során több jelentősebb kihívás merült fel:

- **Technológiai problémák:** a rendszer stabilitása és a Docker-konténerek konfigurációja kezdetben nem volt optimális, ami futtatási hibákat okozott.
- **Funkcionális kihívások:** a bérléskezelési folyamatok pontos leképezése a szoftverben nehézkes volt, különösen a különböző státuszok és jogosultságok kezelése.
- **Időmenedzsment:** mivel a projektet önállóan végeztem, a napi munka és a projekt előrehaladása közötti egyensúly fenntartása komoly kihívást jelentett.
- **Tesztelés és hibajavítás:** a rendszer moduláris felépítése miatt a hibák lokalizálása és javítása több iterációt igényelt.

4.2. Kockázatok azonosítása és priorizálása

A projekt elején egy **kockázatmátrixot** készítettem, amely segített azonosítani a legkritikusabb problémákat:

- **Magas prioritású kockázatok:** kritikus hibák a backend működésében, adatvesztés, biztonsági hiányosságok.

- **Közepes prioritású kockázatok:** kisebb felhasználói felületbeli problémák, riportálási hibák.
- **Alacsony prioritású kockázatok:** kisebb esztétikai hibák vagy későbbi bővítési igények.

A priorizálás lehetővé tette, hogy a projekt során először a legkritikusabb problémákra koncentráljak, így a rendszer stabilitása és funkcionalitása biztosított volt.

4.3. Megoldási stratégiák

A felmerült problémákra a következő stratégiákat alkalmaztam:

- **Technológiai problémák kezelése:** a Docker-konfiguráció és a PHP/Twig környezet iteratív finomhangolása, rendszeres teszteléssel.
- **Funkcionális kihívások kezelése:** a bérléskezelési modulok folyamatos tesztelése és a logika iteratív javítása a hibák kiküszöbölésére.
- **Időmenedzsment:** napi és heti ütemtervek készítése, a feladatok priorizálása a projekt előrehaladása érdekében.
- **Tesztelés és hibajavítás:** moduláris tesztelési stratégia alkalmazása, ahol minden modul külön ellenőrzésen esett át, majd a teljes rendszer integrációját is teszteltem.

4.4. Tanulságok a kockázatkezelésből

A projekt során szerzett tapasztalatok alapján a kockázatkezelés kulcsfontosságú az önálló fejlesztésekben:

- A problémák előrejelzése és a priorizálás jelentősen növeli a projekt sikerességét.
- A rendszeres tesztelés és dokumentáció minimalizálja a hibák előfordulását és gyorsítja a hibajavítást.
- Az időmenedzsment és a feladatpriorizálás elengedhetetlen a projekt gördülékeny előrehaladásához.

5. fejezet

Tanulságok és szakmai összegzés

A **TeDeRMS** projekt nem csupán egy vállalati bérléskezelő rendszer létrehozását jelentette, hanem egy teljes önálló projektmenedzsment gyakorlatot is. A fejlesztés során szerzett tapasztalatok világosan megmutatták, hogy a siker kulcsa a tudatos tervezés, a kockázatok proaktív kezelése és a moduláris, skálázható fejlesztési megközelítés.

5.1. Projektmenedzsment tapasztalatok

Tervezés, ütemezés és fókusz: A részletes ütemterv és mérföldkövek alkalmazása biztosította, hogy minden kritikus feladat időben elkészüljön. Az önálló munkavégzés során a prioritások helyes meghatározása kulcsfontosságú volt: a fókusz a rendszer stabil és hibamentes működésén maradt.

Proaktív kockázatkezelés: A kockázatmátrix és a folyamatos kockázatfigyelés lehetővé tette, hogy a problémákat még a bekövetkezésük előtt azonosítsam és minimalizáljam. Ez a stratégia elengedhetetlen volt a projekt gördülékeny előrehaladásához.

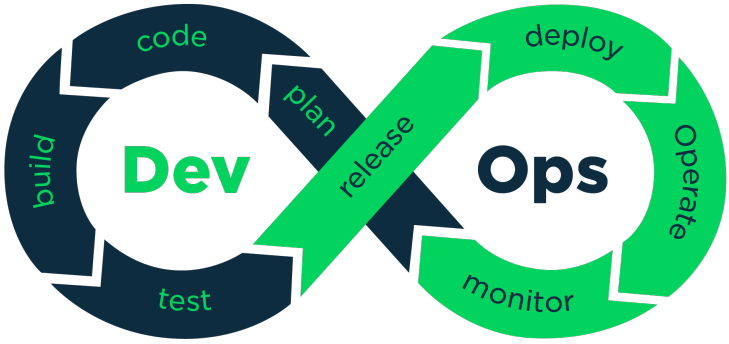
Önálló erőforrás-menedzsment: Az önálló projekt során az idő és a technológiai erőforrások optimalizálása kritikus volt. A napi és heti ütemezés, valamint a feladatprioritások betartása biztosította a folyamatos előrehaladást.

5.2. Fejlesztési és technológiai tanulságok

Moduláris, skálázható architektúra: A rendszer moduláris felépítése lehetővé tette a könnyű karbantartást, gyors hibajavítást és a jövőbeni bővítéseket. Ez a rugalmasság a vállalat hosszú távú igényeit is kiszolgálja.

Folyamatos tesztelés és dokumentáció: A moduláris tesztelés és a részletes dokumentáció biztosította, hogy a rendszer stabilan működjön, a hibák gyorsan javíthatók legyenek, és a felhasználók könnyen eligazodjanak a rendszerben.

Technológiai integráció: A PHP/Twig backend és a Docker alapú konténerizálás kombinációja biztosította a rendszer könnyű telepíthetőségét, stabilitását és rugalmasságát, így a vállalat gyorsan és biztonságosan tudta használni az új rendszert.



5.1. ábra. DevOps szemlélet a projektben

Ábrák jegyzéke

Irodalomjegyzék

[1] Szalay Imre: *Projektmenedzsment alapok: módszertan és gyakorlat*. Budapest, 2018, Budapesti Műszaki és Gazdaságtudományi Egyetem. ISBN 9789633133377.

[2] Hajdu Miklós: *Projektmenedzsment*. Budapest, 2014, Akadémiai Kiadó. ISBN 9789634540181.