Vállalati bérlés- és projektmenedzsment rendszer fejlesztése IT projektmenedzsment szempontok szerint

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 projekt célja nem pusztán a rendszer megvalósítása, hanem annak teljes életciklusának lefedése volt: a tervezéstől és ütemezéstől kezdve a kockázatok feltárásán, a fejlesztésen és a bevezetésen át egészen az üzemeltetésig és karbantartásig.

A dolgozatban 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?

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.

További célok voltak:

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

2.2. A projekt összetettsége és kihívásai

A projekt felettébb összetettnek tekinthető, mivel:

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

2.3. Technológiai háttér

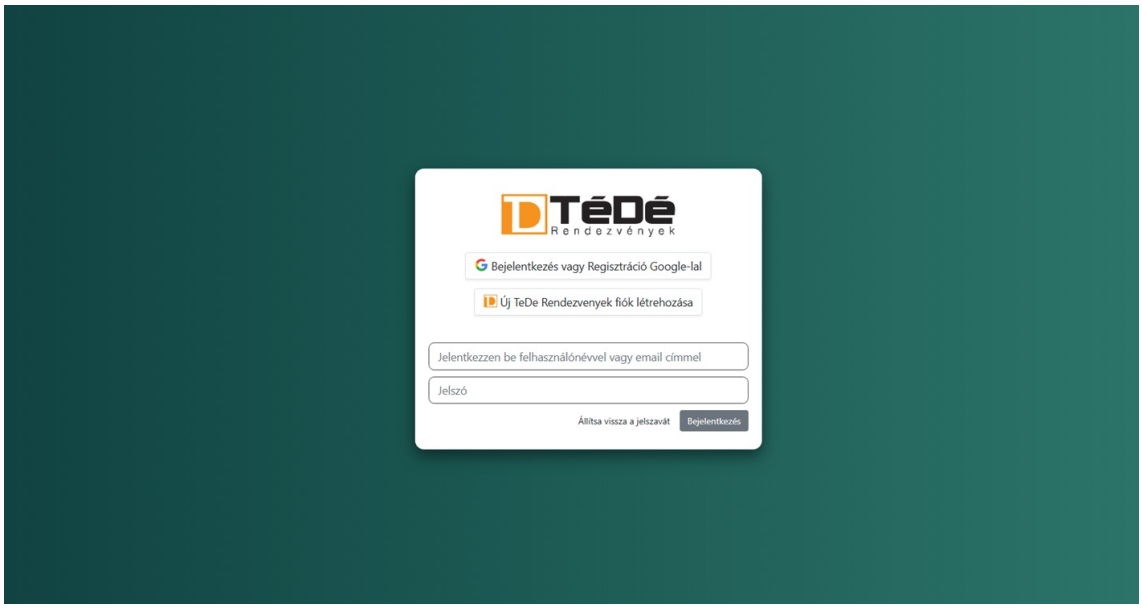
A rendszer fejlesztése a következő technológiákra épült:

- **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.

2.4. A megvalósított szoftver

2.4.1. Bejelentkezés

A bejelentkezési felület biztosítja, hogy a felhasználók biztonságosan és ellenőrzött módon férjenek hozzá 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 csatlakozó 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

2.4.2. 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.

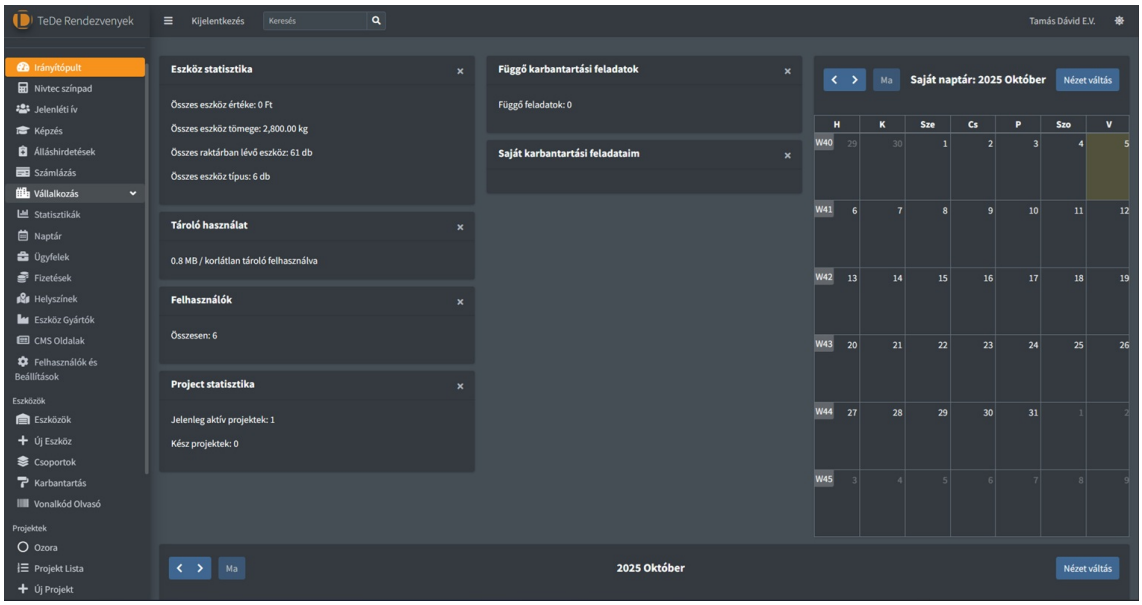
- **Tároló használat:** kijelzi a tárhely aktuális kihasználtságát, ami segíti a rendszeradminisztrátort a tárolókapacitás hatékony felügyeletében, különösen a feltöltött képek és dokumentumok kezelésénél.
- **Felhasználók:** megjeleníti az aktív felhasználók számát, ezzel is elősegítve a jogosultságkezelést 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 az eseményekhez kapcsolódó részletek is megtekinthetőek. 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

2.4.3. 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-vel 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.

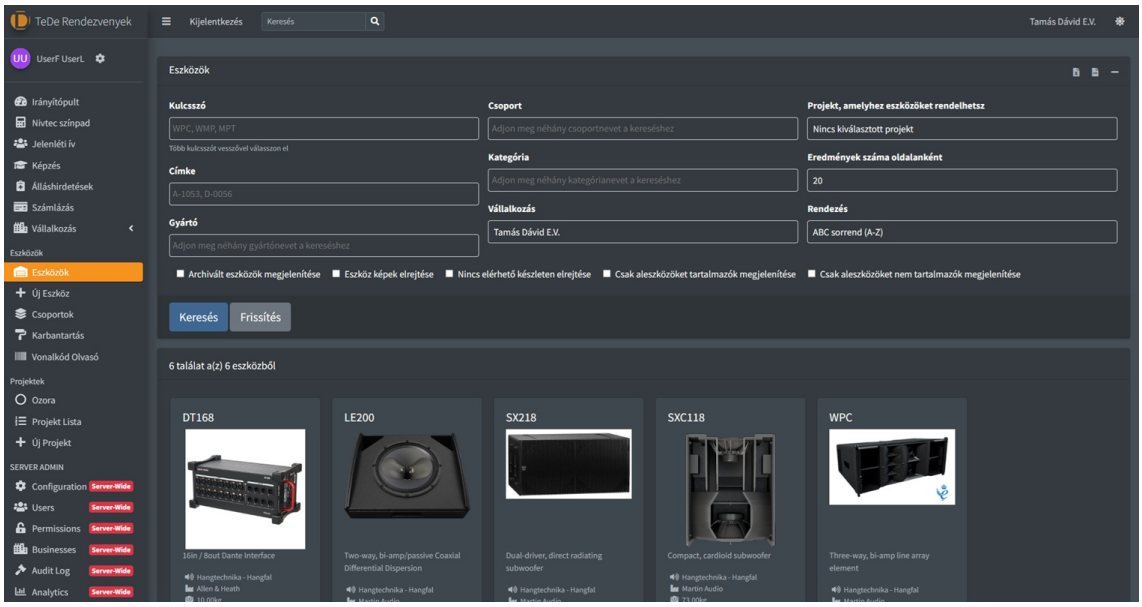
- **Kép:** vizuális azonosítást segítő fotó, amely a felhasználói élményt és a hibamentes választást támogatja.

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:** csak adott projekt-típushoz 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

2.4.4. 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.

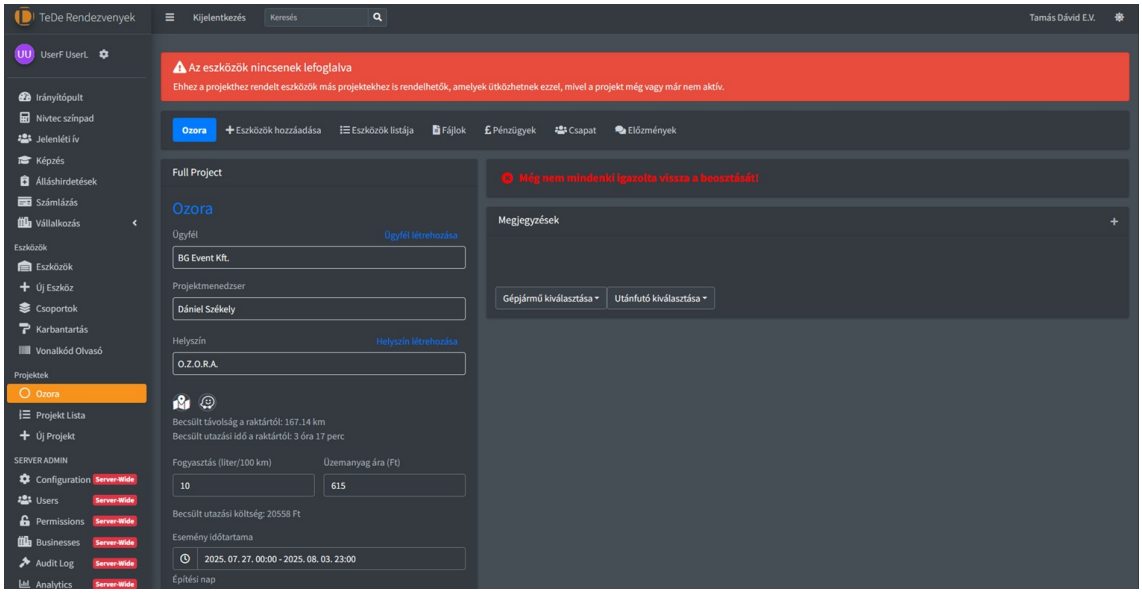
A projekt adatlapja több logikai szekcióra tagolódik:

- **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

2.4.5. 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 által elérhető a verziók követése, a biztonsági mentések készítése és a folyamatos fejlesztés. 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.

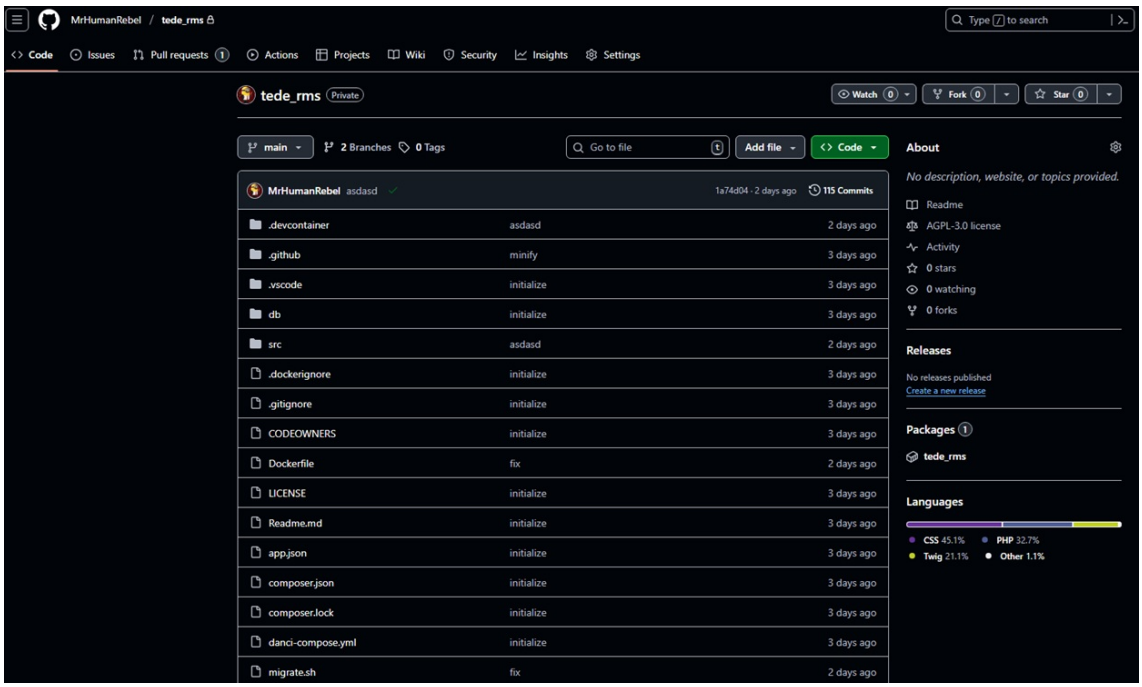
A fejlesztési környezet a **Docker Compose** infrastruktúrán alapul, amely biztosítja a konténerek gyors indítását, újratelepítését és frissítését. A rendszer több szolgáltatásból áll — például webalkalmazás, adatbázis és proxy réteg — amelyek egymással biztonságosan kommunikálnak a konténerhálón belül. 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 segítik 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

3. fejezet

Projekt életciklusa

A projektmenedzsment egyik alapvető elve, hogy minden projekt rendelkezik egy jól definiált életciklussal, amely több, egymásra épülő fázisra bontható. Ez az életciklus nemcsak logikai és időbeli folyamatot ír le, hanem szervezeti, irányítási és ellenőrzési szempontból is meghatározó, hiszen lehetővé teszi a projekt strukturált tervezését, a folyamatok nyomon követését, a kockázatok kezelését és a teljesítmény értékelését [2, 5, 4].

A hazai szakirodalom rendszerint öt alapvető fázist különít el [1, 3]:

1. **Projektindítás (Initiation):** a projekt céljainak, indokoltságának, és a legfontosabb paraméterek meghatározása.
2. **Tervezés (Planning):** részletes ütemezés, erőforrás-tervezés, kockázatelemzés és feladatdefiniálás.
3. **Megvalósítás (Execution):** a projekt tényleges kivitelezése, fejlesztés és implementáció.
4. **Ellenőrzés és irányítás (Monitoring & Controlling):** az előrehaladás, a költségek, az idő és a minőség folyamatos követése.
5. **Lezárás (Closure):** a projekt hivatalos befejezése, átadás, visszacsatolás és tapasztalatok dokumentálása.

3.1. ábra. Projekt kreatív vizualizáció

A fázisok egymásra épülnek, de gyakran átfedésben is zajlanak: például az ellenőrzési és irányítási tevékenység a megvalósítás teljes időtartama alatt folyamatosan jelen van. A modern, iteratív fejlesztési modellek — mint az Agile vagy Scrum — nem feltétlenül követik a klasszikus lineáris struktúrát, hanem sprint-alapú, ciklikus megközelítést alkalmaznak [5, 4].

A klasszikus ötfázisú modell azonban stratégiai és vállalati projekteknél továbbra is nélkülözhetetlen, mivel átlátható keretet biztosít a projekt teljes életciklusára. A **TeDeRMS** fejlesztése során is ez a modell szolgált alapul, amelyet a projekt sajátosságaihoz — önálló fejlesztés, korlátozott erőforrások és vállalati integráció — igazítottam.

3.1. Projektindítás (Initiation)

A projektindítás szakasza a projekt életciklusának alapvető lépése, mivel meghatározza a projekt céljait, irányát és kereteit. A kezdeti fázis során végzett tevékenységek minősége közvetlen hatással van a projekt kockázatkezelésére, a mérföldkövek elérésére és a költséghatékonyságra. A projekt sikeressége nagymértékben függ a kezdeti igényfeltárás alapos elvégzésétől [5, 2, 1, 4].

3.1.1. Igényfelmérés

A vállalat bérléskezelési folyamatait részletesen feltérképeztem, azonosítva a problémás pontokat, és meghatározva a rendszer követelményeit [2]. Az igényfelmérés célja a folyamatok és problémás területek feltárása, amely biztosítja a projekt következő szakaszainak sikeres végrehajtását. A részletes igényfeltárás segít a kockázatok korai azonosításában és a funkciók valós vállalati igényekhez való illesztésében. Emellett fontos a szervezeti kontextus és az érintettek szempontjainak figyelembevétele, mivel ez növeli a projekt elfogadottságát és a végrehajtás hatékonyságát [1, 4].

3.1.2. Célok meghatározása

Világos, mérhető célokat tűztem ki, például az adminisztrációs idő csökkentését, az adatpontosság javítását és az automatizált riportok bevezetését. A célok meghatározása biztosítja a projekt fókuszát és elősegíti az erőforrások optimális elosztását. A világos, mérhető célok segítik a projekt mérföldköveinek követését, és biztosítják a teljesítmény objektív értékelését. A célok hierarchikus rendszere támogatja a stratégiai, taktikai és operatív döntések összehangolását, valamint a projektmenedzsment folyamatok átláthatóságát [5, 2].

3.1.3. Erőforrás-tervezés előkészítése

A projekt kezdetén meghatároztam a fejlesztés technológiai és időbeli erőforrásigényét. Mivel a munkát önállóan végeztem, kiemelten fontos volt a prioritások pontos kijelölése és a munkaidő hatékony beosztása. Az erőforrás-tervezés előkészítése hozzájárult a rendelkezésre álló kapacitások optimális elosztásához, a munkafolyamatok hatékony megszervezéséhez, valamint a feladatok közötti egyensúly megőrzéséhez.

A részletes erőforrás-tervezés segíti a rugalmasság megőrzését, a váratlan események gyors kezelését, és a mérföldkövek teljesítését a tervezett idő- és költségkereten belül [1, 4, 5]. A gondos projektindítás tehát kulcsfontosságú, mivel biztosítja a projekt későbbi fázisainak sikeres végrehajtását, miközben minimalizálja a félreértésekből és az erőforráspazarlásból eredő kockázatokat [2].

3.2. Tervezés (Planning)

A tervezés szakasza kritikus jelentőségű a projekt sikerének biztosításában, mivel ekkor kerülnek meghatározásra az ütemezés, az erőforrások elosztása és a kockázatok kezelése. A részletes és előrelátó tervezés segít minimalizálni a váratlan problémák hatását, előre jelzi a potenciális kockázatokat, és támogatja a projekt teljesítményének nyomon követését [5, 2, 1, 4].

3.2.1. Ütemterv készítése

A projekt kezdeti szakaszában elkészítettem a részletes ütemtervet, amely a fő mérföldköveket és feladatokat logikai és időbeli sorrendbe rendezte. Ez a folyamat biztosította a fejlesztés előrehaladásának nyomon követését, valamint a feladatok közötti függőségek pontos feltérképezését.

Az ütemterv elkészítése során a feladatok időbeli és logikai összefüggéseinek feltárása biztosította a projekt átláthatóságát és a folyamatok szisztematikus követését. A jól felépített ütemterv hozzájárult a mérföldkövek teljesítésének ellenőrzéséhez, és segített az erőforrások hatékony elosztásában [1, 4]. Emellett az ütemterv a projekt előrehaladásának dokumentálását is támogatta, ami elengedhetetlen a projektmenedzsment ellenőrzési és értékelési folyamataihoz.

3.2.2. Erőforrás-tervezés

A projekt megvalósítása során kiemelt figyelmet fordítottam a napi és technológiai erőforrások pontos meghatározására, valamint a munkabeosztás optimalizálására, amely biztosította az önálló fejlesztés gördülékeny előrehaladását.

Az erőforrás-tervezés során a rendelkezésre álló humán és technológiai kapacitások felmérése és optimális elosztása kulcsfontosságú volt. A részletes erőforrás-tervezés előmozdította a projekt rugalmasságát, elősegítette a prioritások helyes meghatározását, és csökkentette a késedelmek kockázatát [5, 2]. Saját tapasztalatom alapján különösen kiscsapatos vagy önálló fejlesztési környezetben a munkaidő hatékony beosztása, valamint a terhelés és pihenőidő egyensúlya alapvető fontosságú. Ez a szemlélet nemcsak a projekt előrehaladását, hanem a minőségi kivitelezést is jelentősen támogatja.

3.2. ábra. Naptári ütemterv vizualizáció

3.2.3. Kockázatelemzés

A projekt tervezési szakaszában részletes kockázatelemzést végeztem, amelynek során azonosítottam a főbb veszélyforrásokat — többek között a technikai hibákat, az esetleges adatvesztést és az üzleti logika működésében rejlő kockázatokat. Ezekre kidolgoztam mind megelőző, mind elhárító intézkedéseket, amelyek célja a problémák előfordulásának minimalizálása, illetve gyors kezelése volt.

A kockázatelemzés során alkalmazott kockázatmátrix megkönnyítette a lehetséges események valószínűségének és hatásának objektív értékelését, ezáltal támogatva a priorizálást és a döntéshozatalt [1, 4]. Ez a módszertan nagymértékben hozzájárult a projekt stabilitásához és a váratlan események hatékony kezeléséhez.

Tehát nem a kockázatelemzés csupán egy formális projektmenedzsment-eszköz, hanem egy gondolkodásmód is, amely segít előre látni a problémákat és felkészülni rájuk. A feladatlista, az ütemezett mérföldkövek és a kockázatmátrix kombinációja, kiegészítve a folyamatos önellenőrzéssel és visszacsatolással, biztosította a projekt strukturált, átlátható és kontrollált előrehaladását [5, 2].

3.3. Megvalósítás (Execution)

A megvalósítás szakasza a projekt tényleges kivitelezését foglalja magában, amely során a tervezett funkciók, folyamatok és erőforrások gyakorlati alkalmazásra kerülnek. Ebben a fázisban a projektmenedzsment és a technikai végrehajtás szoros összhangja kulcsfontosságú a kívánt eredmények eléréséhez [5, 2, 4].

3.3.1. Fejlesztési folyamatok

A fejlesztési folyamatok során kiemelt szerepet kapott a moduláris architektúra kialakítása, amely megalapozta a rendszer különálló komponenseinek önálló fejlesztését és tesztelését. Ez a megközelítés nemcsak a párhuzamos munkavégzést tette lehetővé, hanem a hibák gyorsabb azonosítását és javítását is segítette, így növelve a fejlesztés hatékonyságát és stabilitását.

A moduláris szerkezet mellett a verziókövető rendszer, konkrétan a GitHub, alapvető eszközként szolgált a fejlesztés átláthatóságának fenntartásához és a kód integritásának megőrzéséhez. A verziókezelés segítségével minden módosítás pontosan visszakövethetővé vált, ami különösen fontos volt az önálló fejlesztés során, hiszen így a hibák eredete gyorsan beazonosítható volt [1, 4].

A backend és frontend komponensek párhuzamos fejlesztése szintén elősegítette az erőforrások hatékonyabb kihasználását. Ez a munkaszervezési stratégia nemcsak időmegtakarítást eredményezett, hanem lehetőséget teremtett a különböző funkcionális modulok egyidejű tesztelésére is, így a projekt előrehaladása folyamatos és jól kontrollált maradt.

3.3.2. Tesztelés

A tesztelés a megvalósítási fázis egyik legkritikusabb eleme, mivel közvetlenül befolyásolja a rendszer stabilitását, megbízhatóságát és hosszú távú fenntarthatóságát. A folyamatos és módszeres tesztelés segítette a hibák korai felismerését, ezáltal csökkentve azok későbbi előfordulásának kockázatát. A rendszeres ellenőrzési ciklusok hozzájárultak ahhoz, hogy a fejlesztés iteratív módon, kontrolláltan és minőségbiztosítási szempontból is megalapozottan haladjon előre.

A tesztelési folyamat két fő szintre tagolódott: egyrészt az *unit tesztek* az egyes modulok működését ellenőrizték, biztosítva, hogy minden komponens önállóan is megfelelően funkcionáljon, másrészt a *funkcionális tesztek* a rendszer egészének integrált működését vizsgálták, különös tekintettel az üzleti logika helyességére és a felhasználói élmény konzisztenciájára [2, 5].

A folyamatos validálási szemlélet nemcsak a hibák számát csökkentette, hanem hozzájárult a fejlesztés átláthatóságához és a minőség folyamatos fenntartásához is, ezáltal megalapozva a rendszer hosszú távú megbízhatóságát és üzemeltetési biztonságát.

3.3.3. Dokumentáció

A dokumentáció a projekt hosszú távú fenntarthatóságának és minőségbiztosításának alapvető pillére. A fejlesztés során készített részletes kód- és rendszerleírás nem csupán a karbantartás és a jövőbeli bővítések megkönnyítését szolgálta, hanem a fejlesztési folyamat átláthatóságát és visszakövethetőségét is biztosította.

A dokumentáció elkészítése során különös figyelmet fordítottam a logikai felépítésre, a modulok közötti összefüggésekre és az alkalmazott technológiák részletes ismertetésére, így a rendszer működése és szerkezete bármikor visszakövethetővé vált. Ez különösen fontos volt az önálló fejlesztés szempontjából, hiszen a dokumentálás nemcsak a saját munkám későbbi értelmezését segítette, hanem megteremtette a lehetőséget a projekt más fejlesztők általi folytatására is [1, 4, 2].

A részletes dokumentáció továbbá hozzájárult a tudás megőrzéséhez és a szervezeti tanulás támogatásához, ami a vállalati rendszerek fejlesztésében kiemelt szerepet játszik. A megvalósítás szakasz így nemcsak a szoftver működési stabilitását, hanem a projekt átláthatóságát és minőségét is megalapozta, biztosítva, hogy a rendszer minden funkciója a terveknek megfelelően valósuljon meg, és hosszú távon is hatékonyan szolgálja a vállalat igényeit [5].

3.4. Ellenőrzés és irányítás (Monitoring és Controlling)

Az ellenőrzés és irányítás fázisa a projekt sikerének egyik legfontosabb garanciája, mivel biztosítja, hogy a tervezett célok és mérföldkövek teljesüljenek, az erőforrások optimálisan legyenek felhasználva, és a projekt kockázatai időben kezelhetők legyenek [5, 2, 1, 4]. A monitoring és controlling nem csupán a problémák felderítésére szolgál, hanem a proaktív beavatkozást és a projekt folyamatos finomhangolását is.

3.4.1. Haladás nyomon követése

A projekt sikeres megvalósítása érdekében a haladás folyamatos nyomon követése alapvetően szükséges. A mérföldkövek rendszeres nyomon követése segít a projekt ütemtervének betartásában, és időben feltárja az esetleges eltéréseket.

A folyamatos monitorozás segít a késedelmek és az erőforrás-túlhasználat megelőzésében, valamint támogatja a szükséges korrekciók gyors és célzott végrehajtását. Ez

a megközelítés javítja a projekt átláthatóságát és erősíti a vezetői döntések megalapozottságát, hiszen az objektív teljesítménymutatók alapján lehet meghatározni a szükséges beavatkozásokat [1, 4].

A mérföldkövek nyomon követése nem csupán a projekt aktuális állapotának felmérését szolgálja, hanem hosszú távon hozzájárul a projekt strukturált és ellenőrizhető előrehaladásához, ezáltal a teljes folyamat menedzsmentje hatékonyabbá válik.

3.4.2. Kockázatok kezelése

A projekt sikerességének egyik kulcseleme a kockázatok folyamatos kezelése. A potenciális problémák hatásának minimalizálása érdekében elengedhetetlen, hogy a kockázatokat folyamatosan azonosítsuk, értékeljük és nyomon kövessük.

A kockázatmátrix folyamatos frissítése által, az új kockázatok időbeni azonosítása, a valószínűség és a hatás pontos értékelése, valamint a megfelelő megelőző intézkedések és vészforgatókönyvek kidolgozása könnyebbé válik. Ez a proaktív megközelítés nemcsak a váratlan események negatív következményeit csökkenti, hanem hozzájárul a projekt idő- és költségkeretének betartásához, és növeli a fejlesztés egészének sikerességét [5, 2].

A kockázatkezelés ezen módja adja a döntéshozatal megalapozottságát, és biztosítja, hogy a projekt minden fázisában a problémák hatékonyan, gyorsan és kontrolláltan legyenek kezelve.

3.3. ábra. Kockázatkezelés példa

3.4.3. Minőségellenőrzés

A minőségellenőrzés célja, hogy a rendszer minden funkciója megfeleljen a tervezett követelményeknek, és a hibák a lehető legkorábban azonosításra kerüljenek. A folyamatos ellenőrzés elősegíti a rendszer stabilitását és megbízhatóságát, támogatja az iteratív javítási folyamatokat, és hozzájárul a projekt sikeres lezárásához [1, 4, 2]. Emellett az irányítási és monitoring tevékenységek biztosítják, hogy a mérföldkövek teljesüljenek, a

minőségi elvárások teljesüljenek, valamint lehetőséget adnak a folyamatok optimalizálására és a kockázatok előrejelzésére.

3.5. Projekt lezárás (Closure)

A projekt lezárása a projektmenedzsment egyik kritikus fázisa, amikor a fejlesztett rendszer átadásra kerül, és a projekt során szerzett tapasztalatok összegzésre kerülnek. Ez a szakasz biztosítja a rendszer hosszú távú fenntarthatóságát, a felhasználói igények teljesítését, valamint a projekt teljesítményének és minőségének dokumentált értékelését [5, 2, 1, 4].

3.5.1. Rendszer átadása

A **TeDeRMS** telepítése a vállalat környezetében során a rendszer funkcionalitásának és stabilitásának biztosítása az elsődleges cél. Fontos, hogy az átadás során a rendszer zökkenőmentesen üzemeljen, és a felhasználók számára azonnal használható legyen.

Az átadás során a folyamatok zavartalan működését elősegítő intézkedések, például a telepítési ellenőrzőlisták és a felhasználói oktatás, jelentősen csökkentik a támogatási igényt és növelik a rendszer elfogadottságát. Ez a lépés így nem csupán technikai feladat, hanem kulcsfontosságú a projekt sikeres lezárása és a vállalat napi működésének támogatása szempontjából.

3.5.2. Felhasználói és adminisztrátori dokumentáció

A projekt lezárásának fontos része a részletes dokumentáció elkészítése, amely felhasználói és adminisztrátori kézikönyveket foglal magában. Ez lehetővé teszi, hogy a rendszer működését, konfigurációját és funkcióit később bárki könnyen megértse és kezelje.

A dokumentáció nem csupán útmutató a felhasználók számára, hanem támogatja a karbantartást, a jövőbeli bővítéseket, valamint biztosítja az auditálás és a minőségellenőrzés alapját. A jól strukturált, részletes dokumentáció így közvetlenül megerősíti a rendszer hosszú távú fenntarthatóságához és a projekt eredményességéhez [5, 4].

3.5.3. Oktatás és képzés

A projekt lezárásának kulcsfontosságú része a felhasználók képzése, amely során a kulcsfelhasználók megismerkednek a rendszer működésével és funkcióival. A megfelelő

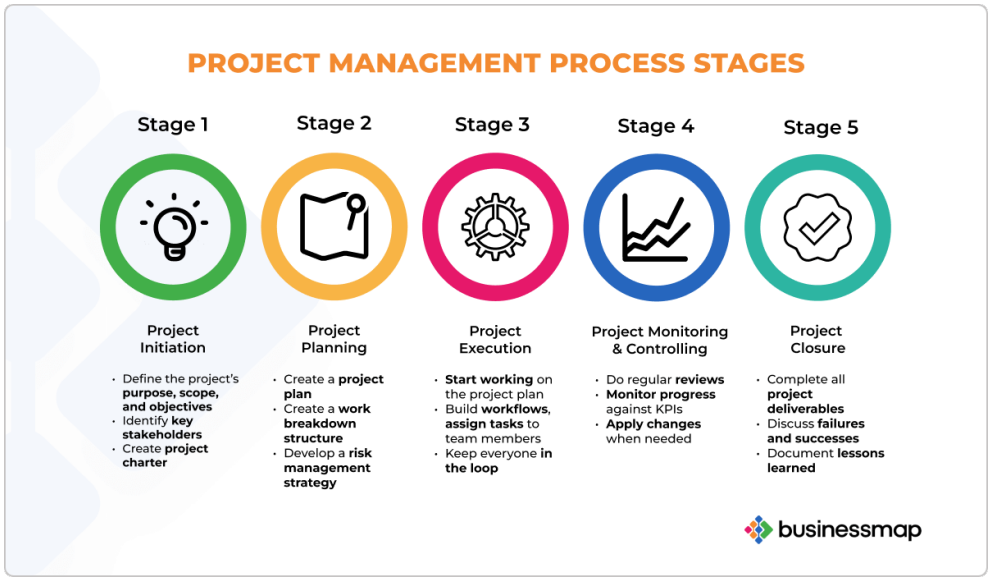
oktatás biztosítja, hogy a rendszer a vállalatnál maximális hatékonysággal legyen alkalmazható, és növeli a felhasználói elfogadottságot.

A képzés közvetlenül segít a hibák csökkentéséhez és a napi működés zavartalanságához, mivel a felhasználók magabiztosan tudják kezelni a rendszert. Emellett az oktatás révén a vállalat csökkentheti a belső támogatási igényeket, és a rendszer önálló, hatékony használatát [2, 1].

3.5.4. Tanulságok összegzése

A projekt lezárásának kulcsfontosságú eleme a tanulságok összegzése, amely elősegíti a szervezeti tudás megőrzését és a folyamatok folyamatos fejlesztését. A projekt során szerzett tapasztalatok dokumentálása segíti a jövőbeli projektek tervezését, a hibák elkerülését, és támogatja a hatékonyabb projektmenedzsment kialakítását.

A tapasztalatok rögzítése hozzájárul a vállalati projektek hosszú távú sikerességéhez és stabilitásához. Emellett a tanulságok összegzése lehetőséget ad a meglévő folyamatok finomhangolására, az erőforrások jobb elosztására, és a stratégiai döntéshozatal megalapozására [5, 2, 4]. Ez a szakasz biztosítja, hogy a rendszer fenntarthatóan, hatékonyan működjön, miközben a projektmenedzsment tapasztalatai a későbbiekben is hasznosíthatók maradnak.



3.4. ábra. Projektmenedzsment folyamatok fázisai

4. fejezet

Kockázatkezelés és problémamegoldás

A **TeDeRMS** projekt során a kockázatkezelés meghatározó 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 számos kihívással és nehézséggel szembesültem, amelyek hatékony kezelése elengedhetetlen volt a projekt sikeres megvalósításához, valamint a rendszer megbízható és teljes funkcionalitásának biztosításához. A tapasztalatok alapján megfigyelhető, hogy a projektmenedzsment klasszikus fázisai jól előre jelzik a problémák jellegét, de a gyakorlatban gyakran szükség van rugalmasságra és kreatív megoldásokra is [5, 2].

4.1.1. Technológiai problémák

A rendszer stabilitása és a Docker-konténerek kezdeti konfigurációja nem volt optimális, ami futtatási hibákhoz vezetett. Ez a probléma rámutatott arra, hogy a modern szoftverfejlesztési környezetekben a technológiai infrastruktúra optimalizálása legalább olyan fontos, mint a szoftver logikájának megtervezése. A stabil technológiai alap kritikus a rendszer megbízhatósága szempontjából, és a kezdeti hibák megfelelő dokumentációval és iteratív konfigurálással kezelhetők [1, 4]. Az ilyen problémák korai felismerése jelentősen csökkentette a későbbi üzemeltetési nehézségeket.

4.1.2. Funkcionális kihívások

A bérléskezelési folyamatok pontos leképezése a szoftverben komoly kihívást jelentett, különösen a különböző státuszok és jogosultságok kezelése során. A funkcionalitás implementálása során világossá vált, hogy a vállalati folyamatok összetettsége gyakran túlmutat a papíron megadott szabályokon.

A rendszertervezés során a folyamatok részletes feltérképezése és a valós működés elemzése fontos a hibák minimalizálásához. Megfigyelésem alapján a moduláris architektúra alkalmazása és a folyamatok iteratív tesztelése kell a pontos implementáció eléréséhez, biztosítva, hogy a rendszer a vállalat tényleges igényeit hűen tükrözze.

4.1.3. Időmenedzsment

Az önálló projektvégzés során különösen nehéz volt összehangolni a napi feladatokat a projekt folyamatos előrehaladásával. Az időmenedzsment kérdése kiemelten fontos volt, mivel a korlátozott erőforrások és a napi feladatok összehangolása folyamatos priorizálást igényelt. A projekt időbeli koordinációja, a mérföldkövek betartása és a személyes produktivitás optimalizálása kritikus tényezők az önálló fejlesztések során [4, 1]. Értékelésem alapján a heti tervezés és a feladatok rendszeres felülvizsgálata segített a haladás fenntartásában.

4.1.4. Tesztelés és hibajavítás

A rendszer moduláris felépítése miatt a hibák lokalizálása és javítása több iterációt igényelt. A hibák azonosítása és javítása a moduláris felépítés miatt komplex feladat volt, mivel egy-egy modul problémája láncreakcióként hatott a rendszer más részeire. A folyamatos unit- és integrációs tesztelés, valamint a hibák dokumentálása és visszacsatolása alapvető a projekt stabilitása szempontjából [5, 2]. Személyes tapasztalatom szerint az iteratív hibajavítás és a részletes tesztelési protokollok alkalmazása biztosította a moduláris rendszer hatékony működését, miközben a megszerzett módszerek későbbi projektekben is hasznosíthatók

4.2. Kockázatok azonosítása és priorizálása

A projekt kezdeti fázisában kiemelten fontos volt a potenciális problémák előrejelzése és kezelése, mivel a kockázatok megfelelő azonosítása és priorizálása közvetlen hatással van a projekt sikerére és a rendszer minőségére. A kockázatmenedzsment alapja

a kockázatok strukturált feltérképezése, az értékelésük és a megfelelő megelőző intézkedések kialakítása [5, 2, 1].

4.2.1. Magas prioritású kockázatok

A magas prioritású kockázatok közé tartoznak a kritikus hibák a backend működésében, az adatvesztés és a biztonsági hiányosságok. Ezek a problémák a rendszer alapvető működését fenyegették, ezért azonnali figyelmet igényeltek. A magas prioritású problémák kezelése kulcsfontosságú a rendszer stabilitásának és megbízhatóságának biztosításához [4, 2]. A kritikus hibák korai azonosítása és a preventív intézkedések bevezetése jelentősen csökkentette a rendszer működési kockázatát, és elősegítette a fejlesztés zavartalan folytatását.

4.2.2. Közepes prioritású kockázatok

A közepes prioritású kockázatok a felhasználói felület hibáit és a riportok pontosságát érintették. Bár ezek nem veszélyeztetik közvetlenül a rendszer alapvető működését, hosszú távon befolyásolhatják a felhasználói élményt és az elfogadottságot [5, 1]. Megítélésem szerint a közepes prioritású problémák folyamatos monitorozása és iteratív javítása hozzájárult a felhasználói élmény optimalizálásához, anélkül, hogy a kritikus funkciókat veszélyeztettük volna

4.2.3. Alacsony prioritású kockázatok

Az alacsony prioritású kockázatok főként a rendszer megjelenését vagy a jövőbeli bővítési igényeket érintették. Bár ezek kevésbé sürgetők, hosszú távon hozzájárulnak a rendszer karbantarthatóságához és a fejlesztési terv fenntarthatóságához [2, 4]. Általam szerzett tapasztalatok alapján az alacsony prioritású problémák nyomon követése támogatta, hogy a projekt fókusza a kritikus funkciók biztosításán maradjon, miközben a jövőbeli bővítések előkészítése is folyamatban volt.

4.3. Megoldási stratégiák

A felmerült problémák kezelése során a cél a rendszer stabilitásának, funkcionalitásának és a projekt határidőinek biztosítása volt. A problémákra alkalmazott stratégiák strukturáltsága és következetessége jelentősen befolyásolja a projekt sikerét [5, 2, 4].

A technológiai problémák, különösen a Docker-konténerek és a PHP/Twig környezet konfigurációja kezdetben instabilitást okoztak, ami futtatási hibákhoz vezetett. Az iteratív

finomhangolás és a rendszeres tesztelés kulcsfontosságú a szoftver stabilitásának biztosításához [1]. A konfigurációk fokozatos javítása, a hibák dokumentálása és a rendszeres tesztkörök alkalmazása jelentősen csökkentette a technikai kockázatot.

A funkcionális kihívások, például a bérléskezelési modulok pontos leképezése a szoftverben, szintén komoly figyelmet igényeltek. A komplex üzleti folyamatok modellezése iteratív tesztelés és visszacsatolás nélkül gyakran vezet félreértésekhez és hibákhoz [2, 4]. A moduláris fejlesztés és az ismételt tesztelés adott lehetőséget, hogy a rendszer logikája a vállalati igényekhez igazodjon, miközben a hibák gyorsan lokalizálhatók és javíthatók voltak.

Az időmenedzsment kritikus tényező volt, mivel a projektet egyedül végeztem. A napi és heti ütemtervek kialakítása, valamint a feladatok rangsorolása segítette a munka és a projekt előrehaladásának egyensúlyban tartását. Az időbeosztás és a feladatpriorizálás kulcsfontosságú a projekt határidőn belüli teljesítéséhez és a kiégés elkerüléséhez [1, 5]. A tervezett és dokumentált ütemezés hozzájárult a hatékony munkavégzéshez és a stressz csökkentéséhez.

A tesztelés és hibajavítás terén a moduláris stratégia alkalmazása biztosította, hogy minden komponens külön-külön ellenőrzésen essen át, majd a teljes rendszer integrációját is alaposan teszteltem. A folyamatos integráció és a moduláris tesztelés szerepét a hibák korai felismerésében és a rendszer megbízhatóságának növelésében [2, 4].

4.4. Tanulságok a kockázatkezelésből

A projekt során szerzett tapasztalatok egyértelműen alátámasztották, hogy a kockázatok előrejelzése és kezelése alapvető fontosságú az önálló fejlesztések sikeréhez. Az előzetesen kialakított kockázatmátrix biztosította a legkritikusabb problémák azonosítását, így a figyelmet a rendszer stabilitását leginkább veszélyeztető területekre tudtam összpontosítani, miközben a kisebb kockázatok kezelését későbbre halaszthattam.

A rendszeres tesztelés és részletes dokumentáció elősegítette a hibák gyors felismerését és javítását, növelte a fejlesztés átláthatóságát, valamint támogatta a későbbi bővítések gördülékeny megvalósítását. A moduláris tesztelési stratégia és az iteratív javítások kombinációja biztosította, hogy a rendszer megbízhatóan működjön, miközben a fejlesztési folyamat hatékonyan szervezett maradt.

Az időmenedzsment és a feladatpriorizálás kulcsfontosságú tényezőnek bizonyult a projekt folyamatos előrehaladásában. A napi és heti ütemtervek, valamint a rangsorolt feladatlista lehetővé tették, hogy a projekt haladása ne veszélyeztesse a munkavégzés minőségét és a határidők betartását. A strukturált időbeosztás és a fókuszált munkavégzés döntő jelentőségű a sikeres, önálló fejlesztéshez.

5. fejezet

Tanulságok és szakmai összegzés

A **TeDeRMS** projekt nem csupán egy vállalati bérléskezelő rendszer létrehozását jelentette, hanem egy teljesen 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

A projekt során szerzett tapasztalataim alapján a részletes tervezés és az ütemezés kulcsfontosságú volt az önálló fejlesztés sikeréhez. A mérföldkövek alkalmazása segített abban, hogy a kritikus feladatokra mindig a megfelelő időben tudjak koncentrálni, és ne veszítsem el a fókuszt a rendszer stabil és hibamentes működésén. Saját tapasztalatom szerint, ha a feladatokat nem a mérföldkövekhez igazítottam volna, gyakran előfordult volna, hogy a kisebb, kevésbé fontos részletek lekötik az időt és energiát, így a projekt határideje veszélybe került volna.

A kockázatok előrejelzése és figyelése rendkívül hasznosnak bizonyult. A projekt során a kritikus problémákat igyekeztem még a bekövetkezésük előtt felismerni, így minimalizálni tudtam az előre nem látható akadályok okozta fennakadásokat. Ez a proaktív megközelítés nemcsak a projekt menetét tette gördülékenyebbé, hanem a saját döntéshozatali képességemet is fejlesztette, hiszen mindig tisztában voltam a legfontosabb teendőkkel és a várható kihívásokkal.

Az önálló erőforrás-menedzsment terén az idő és az eszközök tudatos optimalizálása vált a siker egyik kulcstényezőjévé. A napi és heti ütemtervek betartása, valamint a feladatok rangsorolása biztosította, hogy a projekt folyamatosan haladjon, anélkül hogy túlterhelt vagy demotivált lettem volna. Ebből a szempontból úgy érzem, hogy a projekt nem csupán a technikai készségeimet, hanem a saját felelősségvállalásomat és önálló munkavégzési képességemet is jelentősen fejlesztette.

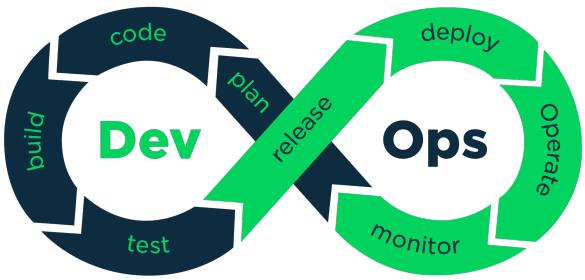
Összességében úgy látom, hogy a tudatos tervezés, a proaktív kockázatkezelés és az önálló erőforrás-menedzsment integrált alkalmazása nemcsak a projekt céljainak elérését segítette, hanem hozzájárult a saját fejlődésemhez is. A projekt során szerzett tapasztalatok alapján a jövőbeni önálló fejlesztések során ezek az elvek elengedhetetlenek a hatékony és sikeres munkavégzéshez.

5.2. Fejlesztési és technológiai tanulságok

A projekt során szerzett tapasztalataim alapján a moduláris és skálázható architektúra kialakítása kulcsfontosságú volt. A rendszer moduláris felépítése lehetővé tette, hogy a különböző funkcionális egységeket egymástól függetlenül fejlesszem, teszteljem és javítsam, ami jelentősen gyorsította a hibajavítási folyamatokat, és könnyebbé tette a későbbi bővítéseket. Tapasztalatom szerint ez a rugalmasság nem csupán a fejlesztői munka hatékonyságát növelte, hanem hosszú távon a vállalat igényeinek kiszolgálását is biztosítja.

A folyamatos tesztelés és a részletes dokumentáció szintén meghatározó tényezőnek bizonyult. A moduláris tesztelési stratégia megteremtette a lehetőségét, hogy a hibákat gyorsan lokalizáljam és javítsam, miközben a rendszer teljes integritása megőrződött. A dokumentáció elkészítése nem csupán a rendszer áttekinthetőségét segítette, hanem a felhasználók számára is könnyebbé tette az eligazodást, ami a rendszer elfogadottságát és a hatékony működést növelte.

A technológiai integráció terén a PHP/Twig backend és a Docker alapú konténerizálás kombinációja jelentős előnyöket biztosított. A konténerizált környezet stabilitást és rugalmasságot biztosított, miközben a rendszer telepítése és frissítése gyorsan elvégezhető volt. Saját tapasztalatom szerint ez a megoldás csökkentette a konfigurációs problémák és futtatási hibák előfordulását, valamint biztosította a feltételeit, hogy a vállalat az új rendszert azonnal, biztonságosan és hatékonyan használja.



5.1. ábra. DevOps szemlélet a projektben

Ábrák jegyzéke

Irodalomjegyzék

[1] Kovács Gábor: *Projektmenedzsment a gyakorlatban*. Budapest, 2016, Typotex Kiadó. ISBN 9789632798201.

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

[3] Simon István: *Modern projektmenedzsment módszertanok*. Budapest, 2015, Complex Kiadó. ISBN 9789632972435.

[4] Kaposi József: *Agilis és hagyományos projektmenedzsment: elmélet és gyakorlat*. Budapest, 2019, Budapesti Corvinus Egyetem Kiadó. ISBN 9789635301293.

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