Vállalati bérlés- és projektmenedzsment rendszer fejlesztése IT projektmenedzsment szempontok szerint

Székely Dániel

Mérnökinformatikus MSc

2025

Tartalomjegyzék

1. fejezet

Bevezetés

A digitális transzformációban működő vállalatoknál az idő, a költség és a minőség összehangolása döntően projektmenedzsment-feladat, amelyet rendszerszerű tervezés és kockázatkezelés támaszt alá [**?** **?**]. A TéDé Rendezvények vállalatnál szerzett többéves tapasztalatom azt mutatta, hogy a papíralapú vagy táblázatos nyilvántartások lassítják a döntéshozatalt és növelik a hibák esélyét, ezért saját fejlesztésben valósítottam meg a **TeDeRMS** (Rental Management System) megoldást.

A dolgozat célja, hogy röviden bemutassa a rendszer funkcióit és technológiai hátterét, miközben a projektmenedzsment elméleti kereteit végig összekapcsolja a gyakorlati megvalósítással. A hangsúly azokon az elemein van, amelyek a vállalati környezetben bizonyítottan értéket teremtettek: az igényfelmérésen, az ütemezésen, a kockázatkezelésen és a bevezetésen [**?** **?**].

A fő kérdések, amelyekre a fejezetek választ adnak:

- Milyen üzleti igényekre épült a rendszer, és ezek hogyan jelennek meg a funkcionális tervezésben?
- Hogyan alkalmaztam a projektéletciklus fázisait a TeDeRMS fejlesztésére, és milyen mérőszámok jelezték a haladást?
- Mely kockázatok befolyásolták leginkább az önálló fejlesztést, és milyen válaszokat adtam rájuk a szakirodalom ajánlásai alapján?
- Milyen tanulságokat lehet más, hasonló vállalati fejlesztésekre átültetni?

2. fejezet

Projekt áttekintése és részletes bemutatása

2.1. A vállalat és az üzleti igények

A TéDé Rendezvények rendezvénytechnikai szolgáltatóként magas értékű eszközparkot és párhuzamos projekteket kezel. Saját tapasztalatom szerint a papíralapú és Excel-folyamatok lassú egyeztetést, többszörös adatbevitelt és hibás készletinformációt eredményeztek. A szakirodalom is hangsúlyozza, hogy az átlátható erőforrás- és dokumentumkezelés kulcsa az integrált rendszerbe szervezett folyamat [**?** **?**]. A projekt ezért egy egységes, online bérlés- és projektkezelő platform létrehozását tűzte ki célul.

A legfontosabb üzleti elvárások:

- pontos készlet- és eszköznyilvántartás valós idejű státuszokkal;
- gyors ajánlatadás és projektindítás automatizált sablonokkal;
- jogosultságkezelés és naplózás a felelősségi körök elkülönítésére;
- egységes, auditálható dokumentumtár a rendezvénydokumentációhoz.

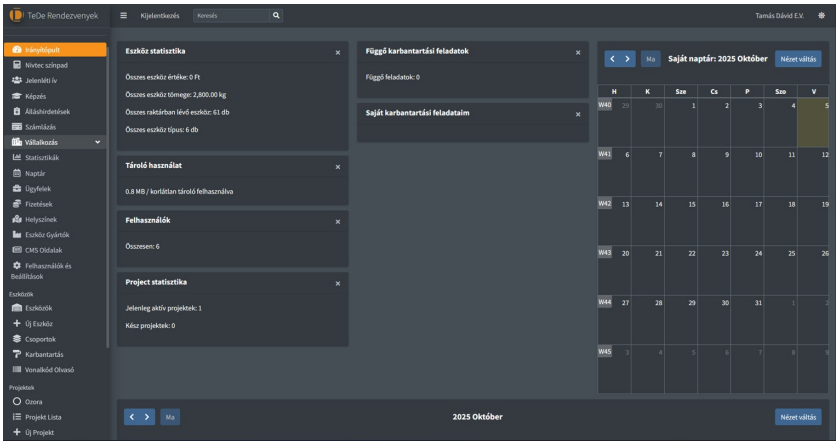
2.2. Funkciók rövid áttekintése

A TeDeRMS moduláris felépítésben készült, hogy a vállalati folyamatokat közvetlenül támogassa.

- **Hitelesítés és onboarding:** helyi fiók vagy Google OAuth2, majd csatlakozó kód a vállalati tenant-hoz. A megoldás megfelel a szakirodalom által kiemelt jogosultsági kontrollnak [**?**].

- **Dashboard:** valós idejű eszköz-, projekt- és felhasználói mutatók, valamint közös naptár nézet a kapacitástervezéshez [**?**].
- **Eszközkezelés:** címkézett eszközök, részletes metaadatok, fotók és bérlési árak; rugalmas szűrés projektre, kategóriára és státuszra.
- **Projektlap:** ügyféladatok, felelősök, helyszínek, státuszok és erőforrás-hozzárendelés egy nézetben, hogy a tervezés és kontroll összekapcsolódjon [**?**].
- **Riportálás:** alap pénzügyi és kihasználtsági riportok, amelyek a lezárt projektekből generálhatók.

2.1. ábra. Bejelentkezés és csatlakozás a vállalathoz. **Forrás: saját kép**



2.2. ábra. Dashboard a fő erőforrásmutatókkal. **Forrás: saját kép**

2.3. Technológiai háttér

A rendszer fejlesztése a következő technológiákra épül, amelyek a skálázhatóságot és a könnyű üzemeltetést szolgálják:

- **Backend:** PHP és Twig sablonmotor a gyors szerveroldali rendereléshez.
- **Frontend:** reszponzív nézetek és újrafelhasználható komponensek.
- **Adatbázis:** MySQL a strukturált adatokhoz, migrációs scriptekkel.
- **Konténerizálás:** Docker Compose lokális és éles környezethez, előre definiált `.env` változókkal.
- **Reverse proxy:** Nginx a TLS-védelemmel ellátott forgalom és a statikus fájlok kiszolgálására.
- **Verziókezelés:** Git alapú munkafolyamat, hogy a változások visszakövethetők legyenek [**?**].

3. fejezet

Projekt életciklusa

A projektmenedzsment szakirodalom ötfázisú életciklust ír le: indítás, tervezés, megvalósítás, ellenőrzés és lezárás [**?** **?**]. A TeDeRMS fejlesztését úgy szerveztem, hogy minden fázisban mérhető eredmény készüljön, és a gyakorlatban igazoljam az elméleti elveket.

3.1. Projektindítás

A kiinduló pont az igényfelmérés és a scope definiálása volt. A vállalatnál végzett interjúk és a korábbi projektek tapasztalatai alapján üzleti követelmény-listát készítettem, amelyet prioritás szerint rangsoroltam. Ez megfelel a szakirodalom által javasolt érintetti elemzésnek és üzleti célokhoz kötött célfa készítésének [**?** **?**]. A projektindító dokumentumban rögzítettem az elvárt átfutási időt (10 hónap) és a minőségi kritériumokat (adatpontosság, rendelkezésre állás), így később ezekhez mértem a teljesítést.

3.2. Tervezés

Az ütemezést Gantt-szerkezetben készítettem el: igényértelmezés (2 hét), architektúra-terv (2 hét), modulfejlesztés (16 hét), integrációs teszt (4 hét), bevezetés (2 hét). A mérföldkövekhez konkrét átadási tételeket rendeltem, hogy a controlling szakaszban objektív mérőpontok legyenek [**?**]. Az erőforrás-terv az önálló fejlesztés miatt heti 25 munkaórával számolt, és tartalékidőt is tartalmazott a kockázatos modulokra (auth, jogosultság, riportok) [**?**]. A kockázatok előzetes feltárását kockázatmátrixba rögzítettem, amely a valószínűség-hatás párok alapján priorizált.

3.1. ábra. Fő mérföldkövek időterve. **Forrás: saját kép**

3.3. Megvalósítás

A fejlesztés moduláris architektúrán alapult (auth, dashboard, eszközkezelés, projektek, riportok), ami lehetővé tette a párhuzamos, de kontrollált fejlesztést. A Git-flow szerinti ágkezelés és a konténerizált környezet biztosította, hogy minden módosítás reprodukálható legyen [**?**]. A backend és frontend komponenseket iteratív sprintekben szállítottam: minden sprint végén működő, tesztelt modul készült, így a projekt több alkalommal produkált használható verziót, ahogy az iteratív modellek ajánlják [**?**].

3.4. Ellenőrzés és irányítás

Heti státusz-összefoglalót készítettem az előrehaladásról, amelyben a becsült és tényleges időráfordításokat hasonlítottam össze. Az eltérések esetén újraterveztem az ütemezést (pl. a jogosultsági modul több időt igényelt). A minőségbiztosítást unit- és funkcionális tesztekkel végeztem: kritikus út moduljai (auth, eszközkeresés) minden sprintben regressziós tesztet kaptak [**?**]. A monitorozás során azonosított hibákat issue-ként rögzítettem, és a kockázati szintjük alapján priorizáltam.

3.5. Lezárás

A bevezetés előtt pilotot futtattam a vállalat egyik kisebb projektjén, amely során valós adatokkal ellenőriztem a folyamatokat. A felhasználói visszajelzések alapján javítottam a naptárnézetet és az eszközszűrést. A lezárás részeként átadtam a rendszer- és üzemeltetési dokumentációt, valamint az üzemeltetési checklistát. A tapasztalatok összegzése a záró fejezetben található, követve a tudásmenedzsmentre vonatkozó ajánlásokat [**?**].

4. fejezet

Kockázatkezelés és problémamegoldás

A kockázatkezelést már a projektindításkor rendszerbe szerveztem, mert egyszemélyes fejlesztésnél minden késedelem közvetlenül az átfutási időt veszélyezteti [**?**]. A kockázatmátrixot a szakirodalom javaslatai szerint a valószínűség és hatás alapján töltöttem ki, majd minden tételhez megelőző és reagáló intézkedést rendeltem [**? ?**].

4.1. Főbb kockázatok és válaszok

Kockázat	Valószínűség	Hatás	Válasz
Docker/Nginx konfigurációs hiba	Közepes	Magas	Előre összeállított Compose profilok, automatikus healthcheckek, rollback script.
Adatvesztés vagy sérülés	Alacsony	Magas	Napi dump cron, verziózott migrációk, tesztadatbázis bevezetése.
Funkcionális félreértelmezés (bérlés státuszok)	Közepes	Közepes	Részletes use case leírások, prototípus-demó minden sprint végén.
Időcsúszás az önálló fejlesztés miatt	Magas	Közepes	Heti 25 órás keret, buffer a kockázatos modulokra, scope-fagyasztás a 3. sprint után.
Jogosultsági hibák vagy jogosulatlan hozzáférés	Alacsony	Magas	Role-based access control, kötelező audit log, automatizált belépési tesztek.
Teljesítmény problémák nagy eszközlistán	Közepes	Közepes	Indexelt adatbázis mezők, lapozás és gyorsszűrés beépítése, célzott terheléses tesztek.
Külső szolgáltatás (Google OAuth) elérhetetlensége	Alacsony	Közepes	Lokális fiók fallback, auth endpoint monitoring, riasztás beállítása.

4.1. táblázat. Kockázatkezelési mátrix a TeDeRMS projekthez. **Forrás: saját kép**

4.2. Problémák kezelése a gyakorlatban

Technológiai kockázatok: a konténerindításkor jelentkező hibákra automatizált ellenőrző szkripteket írtam, így a futási problémák már a fejlesztői környezetben feltűntek. Ez összhangban van a megelőző kontrollokra építő ajánlással [**?**].

Funkcionális eltérések: a bérlés státuszmátrixa többször pontosításra szorult. Minden módosítás után rövid használhatósági tesztet futtattam a vállalat kulcsfelhasználóival, és az eredményeket jegyzőkönyvben rögzítettem.

Idő- és erőforráskockázat: ha a tényleges ráfordítás meghaladta a tervezettet, a következő sprintben kevesebb új funkciót vállaltam, és a hibajavításra fókuszáltam. A módszer megfelelt a szakirodalomban javasolt gördülő tervezésnek [**?**].

4.1. ábra. Problémamegoldási folyamat vizualizációja. Forrás: saját kép

5. fejezet

Tanulságok és szakmai összegzés

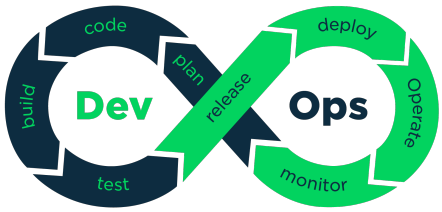
A TeDeRMS projekt során szerzett tapasztalatok bemutatják, hogyan kapcsolható össze a projektmenedzsment elmélet és a vállalati gyakorlat.

5.1. Projektmenedzsment tanulságok

- **Célok és mérföldkövek:** a mérhető mérföldkövek (auth, eszközmodul, riportok) folyamatos visszajelzést adtak, és csökkentették a scope-kúszás kockázatát [**?**].
- **Rendszeres monitorozás:** a heti státusz-összefoglalók segítették a becslések kalibrálását, a kockázatos feladatokat pedig előrehozták a következő sprintekbe [**?**].
- **Stakeholder bevonás:** a kulcsfelhasználókkal tartott demók biztosították, hogy az elméleti követelmények (RACI, felelősségi körök) valódi működési folyamatokká váljanak [**?**].

5.2. Fejlesztési és technológiai tapasztalatok

- **Modularitás:** az önállóan fejlesztett modulok (auth, eszközkezelés, projektek) külön tesztkörnyezetet kaptak, ami gyors hibakeresést tett lehetővé és csökkentette a regressziókat [**?**].
- **Konténerizálás:** a Docker alapú környezet minimalizálta a „működik a gépemen” típusú problémákat, és biztosította a reprodukálható bevezetést [**?**].
- **Dokumentáció:** a rendszer- és üzemeltetési dokumentáció elkészítése megfelelt a lezárási fázis tudásmegosztási elvárásainak, és a pilot után finomítottam őket a felhasználói visszajelzések alapján [**?**].



5.1. ábra. A DevOps szemlélet szerepe a projektben. Forrás: saját kép

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

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