ZÁRÓDOLGOZAT

Készítették:

Baranyi-Kiss Ágnes – Juhász Zsuzsanna – Mészáros Tibor Szabolcs

Konzulens:

Farkas Zoltán

Miskolc

2025.

Miskolci SZC Kandó Kálmán Informatikai Technikum

Miskolci Szakképzési Centrum

SZOFTVERFEJLESZTŐ- ÉS TESZTELŐ SZAK

Szakdolgozat

HouseOfMysteries

Baranyi-Kiss Ágnes – Juhász Zsuzsanna – Mészáros Tibor Szabolcs

2024-2025

Tartalomjegyzék

Tartalom

Témaválasztás indoklása

A HouseOfMysteries webes alkalmazás célja, hogy egy egyszerű és felhasználóbarát platformot biztosítson egy szabadulószoba foglalási rendszernek. A fontosságát azon üzleti igény adja, hogy a játékosok egy kényelmes felületen foglalhassanak időpontot, ezen kívül egyedi módon az egyes játékoscsapatoknak versenylehetőséget is biztosít egymás között.

A foglalási rendszer az alábbi problémákra ad megoldást:

- Szobánként lehet foglalni az időpontokat, így a játékos átlátja a weboldalt.
- Lehetőség az egyéni- és csapatregisztrációra.
- Dashboardon nyomon követhető eredmények, valós idő alapján.
- A fenti eredményekből adódóan különböző csapatok versenyezhetnek egymással, például egy nagyobb baráti társaság összemérheti ugyanabban a szobában az ügyességüket, problémamegoldó képességüket.

A fenti koncepció azért egyedülálló, mert országosan nem találtunk olyan szabadulós játékot, ahol a csapatok versenyben is összemérhetik a tudásukat, fizikai szobákban.

Felhasznált technológiák

A projektünk készítésekor többféle technológiát használtunk, ügyelve arra, hogy a való életben is használható legyen a végeredmény. Ügyeltünk arra, hogy ne csak üzletileg legyen helytálló, felhasználói élményben is kielégítő legyen egy esetleges megrendelő számára.

MySQL - Adatbázis

A MySQL egy nyílt forráskódú, többfelhasználós, többszálú, SQL-alapú relációs adatbázis-kezelő szerver, melyet adatok tárolására és lekérdezésre használnak. SQL nyelv segítségével lehet adatbázist létrehozni, adatokat lekérni, és módosításokat végrehajtani a táblákon. A MySQL egy népszerű RDBMS webes alkalmazásokhoz, mert megbízható és gyors.

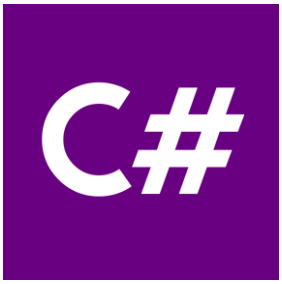
1. ábra – MySQL Logo

Entity Framework Core – Backend

Az EFC egy modern objektum-adatbázis leképező a .NET rendszerhez. Leegyszerűsíti a különböző adatbázisok használatát erősen gépelt .NET objektumokkal és a LINQ támogatásával. Az Entity Framework a legtöbb adatbázisrendszert támogatja, pl.: MySQL, SQLite, PostgreSQL, MariaDB.

2. ábra – Entity FW logo

C# - Backend



A C# egy objektumorientált programozási nyelv. Az objektumorientált programozás négy alapelve:

Absztrakció: Az entitások releváns attribútumainak és interakcióinak modellezése osztályokként a rendszer absztrakciós ábrázolásának meghatározásához.

Beágyazás: Elrejti az objektumok belső állapotát és funkcióit, és csak nyilvános függvénykészleten keresztül engedélyezi a hozzáférést.

Öröklési képesség: új absztrakciók létrehozására a meglévő absztrakciók alapján.

3. ábra – C# logo

Polimorfizmus: Az öröklött tulajdonságok vagy metódusok implementálása különböző módokon több absztrakcióban.

Asp.Net Core API – Backend

Az ASP.NET egy nyílt forráskódú szerveroldali webalkalmazás-keretrendszer, amelyet dinamikus weboldalak előállítására fejlesztettek. http protokollt használ a kliens-weboldali kommunikációhoz, az adatok hozzáférése érdekében.

HTML – Frontend

4. ábra – HTML logo

A HTML nem programozási, hanem leíró nyelv, melyet weboldalak készítéséhez fejlesztettek. A Hypertext Markup Language rövidítése. Mint minden kódnyelvet, ezt is írhatjuk akár jegyzettömbben is. A hipertext jelenti az interneten található dokumentumokat (oldalakat), amelyek szöveg, két, videó, hang, animáció, vagy ezek kombinációjából épülnek fel.

CSS - Frontend

5. ábra – CSS logo

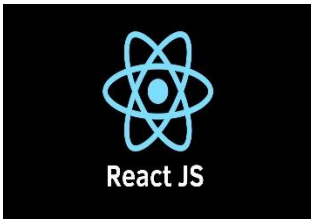
A Cascade Style Sheets rövidítése. Az egyik legfontosabb technológia a weboldalak vizuális kialakításában. A HTML oldalak kinézetét, stílusát az itt megadott feltételekkel tudjuk befolyásolni, és lehetővé teszi különböző elemek, képek, animációk, videók elhelyezését a webes dokumentumainkba. A webböngészők megvizsgálják a HTML oldal CSS kódját, és ez alapján jelenítik meg a különböző elemeket.

Javasript - Frontend

6. ábra – JavaScript logo

A Javasript a webfejlesztés alapvető nyelve. Jellemzően böngészőben futtaták, frontend és backend fejlesztésre is alkalmazható. Kliensoldali szkriptnyelv, amivel interaktívvá tehetjük a weboldalainkat, ezen kívül node.js segítségével szerveroldali alkalmazások is készíthetők vele.

React – Frontend



7. ábra – React logo

A React a Javascript-könyvtárak egyike. Nyílt forráskóddal rendelkezik, így bárki használhatja a projektjei során. Interaktív felhasználói felületek létrehozására szolgál. Ez egy frontend-könyvtár, ami lehetővé teszi összetett felhasználói felületek létrehozását vagy felépítését kis, elszilegetelt kódrészletek, ún. komponensek felhasználásával. Az összetevők fő előnye, hogy egyetlen komponens módosítása nem érinti a teljes alkalmazást.

React Router – Frontend

8. ábra – React Router logo

React alkalmazások útvonaltervezéséhez és navigációjához használt könyvtár. Ez azt jelenti, hogy az URL cím és az alkalmazásunk összhangban van, így a megfelelő komponenst jeleníti meg. Használható egyetlen komponens megjelenítésére, vagy bizonyos részeket (pl. Navbar, Footer, stb.) statikusak maradnak, ami azt jelenti, hogy a felhasználó számára minden aloldalunkról látható és elérhető marad. Ennek segítségével csak az oldal tartalmi része rendelelődik újra, aminek köszönhetően gyorsabban tölt be az oldalunk.

Python – tesztelés, adatvizualizáció

9. ábra – Python logo

A Python egy többparadigmás programozási nyelv. Támogatja az objektumorientált és a strukturált programozást, emellett funkcionális és aspektusorientált programozást biztosít metaprogramozással és metaobjektumokkal.

Selenium – tesztelés

10. ábra – Selenium logo

A Selenium egy webalkalmazások automatikus tesztelésére szolgáló keretrendszer. Széles körben alkalmazható, az egyik legismertebb nyílt forráskódú teszteszköz. Különösen webalkalmazás fejlesztők számára praktikus, mert csökkenti a tesztidőt, ezzel gyorsítva, megbízhatóbbá és rugalmasabbá téve azt.

Fejlesztői környezetek

Ahhoz, hogy a projektünk megfelelően működjön, többféle fejlesztői környezetet használtunk, hogy a teljes felhasználói környezet megfeleljen minden igénynek. Figyelembe vettük, hogy user-friendly környezetet alakítsunk ki, a felület logikusan és áttekinthetően épüljön fel, világos és áttekinthető utasításokkal könnyen elvégezhessék a regisztráció és foglalás folyamatát. Ezen felül arra is ügyeltünk, hogy az adminisztrációs felület könnyen használható legyen a felülettel dolgozó felhasználók számára. Üzleti szempontokat is figyelembe véve alakítottuk ki a különböző felhasználói csoportok igényeit.

XAMPP

11. ábra – XAMPP logo

A XAMPP egy olyan szoftvercsomag, ami többek között Apache webszervert és MySQL adatbáziskezelőt, PHP-t tartalmaz, így könnyen telepíthető és üzembe helyezhető az adatbázis-fejlesztői környezet. A XAMPP segítségével a fejlesztők lokálisan tudnak webalkalmazásokat fejleszteni és tesztelni.

Használatával könnyedén létre tudunk hozni egy teljes webkiszolgáló környezetet a saját számítógépünkön, és tesztelhetjük a weboldalunk működését, illetve könnyedén kezelhetjük a kapcsolódó adatbázisunkat.

Visual Studio Code

12. ábra – VS Code logo

A Visual Studio Code (rövid nevén VS Code) egy nyílt forráskódú, integrált fejlesztői környezet, melyet A Microsoft fejlesztett ki Windows, Linux, macOS és webböngészők számára. A funkciók közé tartozik a hibakeresés támogatása, a szintaxis kiemelése, az intelligens kódkészítés, a kódrészletek, és a kód újrafeldolgozása, valamint a Git beágyazott verziókezelése. A VS Code alkalmas különböző programozási nyelvekhez, keretrendszerekhez, és a fejlesztők egyedi bővítményekkel is testre szabhatják a munkakörnyezetüket.

Visual Studio 2022

13. ábra – VS 2022 logo

A Visual Studio 2022 egy többplatformos fejlesztői környezet, amit elsősorban Windows operációs rendszerre terveztek. Segítségével hatékonyan készíthetünk alkalmazásokat, beleértve az asztali alkalmazásokat, webalkalmazásokat és szolgáltatásokat. A VS 2022 több programozási nyelvet támogat, beleértve a projektünkhöz használt C# nyelvet. Ebben a fejlesztői környezetben lehetőségünk van kódírásra, hibakeresésre, tesztelésre, továbbá vizuális tervezésre.

Swagger

14. ábra – Swagger logo

A Swagger egy nyílt forráskódú keretrendszer az API leírására, egy mindenki számára érthető, közös nyelven. RESTful APIk tervezésére, építésére és dokumentációjára használjuk. Érdemes úgy tekinteni rá, mint egy ház tervrajzára. A dokumentum JSON vagy YAML formátumban készül, bemutatja az API végpontjait és az általuk elfogadott paramétereket, valamint az API által elfogadott válasz- és hibakódokat. Köszönhetően annak, hogy több programozási nyelvet és keretrendszert is támogat, lehetővé teszi a programozók számára, hogy csökkentsék a hibák számát, így elérve a hatékonyabb munkavégzést a fejlesztés során.

Kommunikációs felületek

A projekt során többféle kommunikációs felületet használtunk attól függően, éppen melyik státuszban jártunk a felépítés során. A lentiekben ezeket fogjuk bemutatni.

Discord

15. ábra – Discord logo

A Discord egy ingyenes VoIP alkalmazás, amit első sorban videojáték-közösségek számára fejlesztettek ki, de nem zár ki más témájú közösségeket sem. A platformot úgy fejlesztették ki, hogy nagy rendszerigényű programok futtatása mellett is gördülékenyen lehessen használni. Mivel ingyenes konferenciahívás-lehetőséget is biztosít, a tervezés szakaszában átköltöztettük a napi kommunikációs gyakorlatunkat erre a felületre. Ezen a felületen kényelmesebben tudtuk megosztani egymással a képernyőinket és az elkészült fájljainkat.

Trello

16. ábra – Trello logo

A Trello egy webalapú, Kanban stílusú listakészítő alkalmazás.Itt táblázatos formában tudtuk meghatározni az éppen aktuális feladatokat, határidőket megjelölni, illetve az elkészült részfeladatokat megosztani a csapatban.

GitHub

17. ábra – Git logo

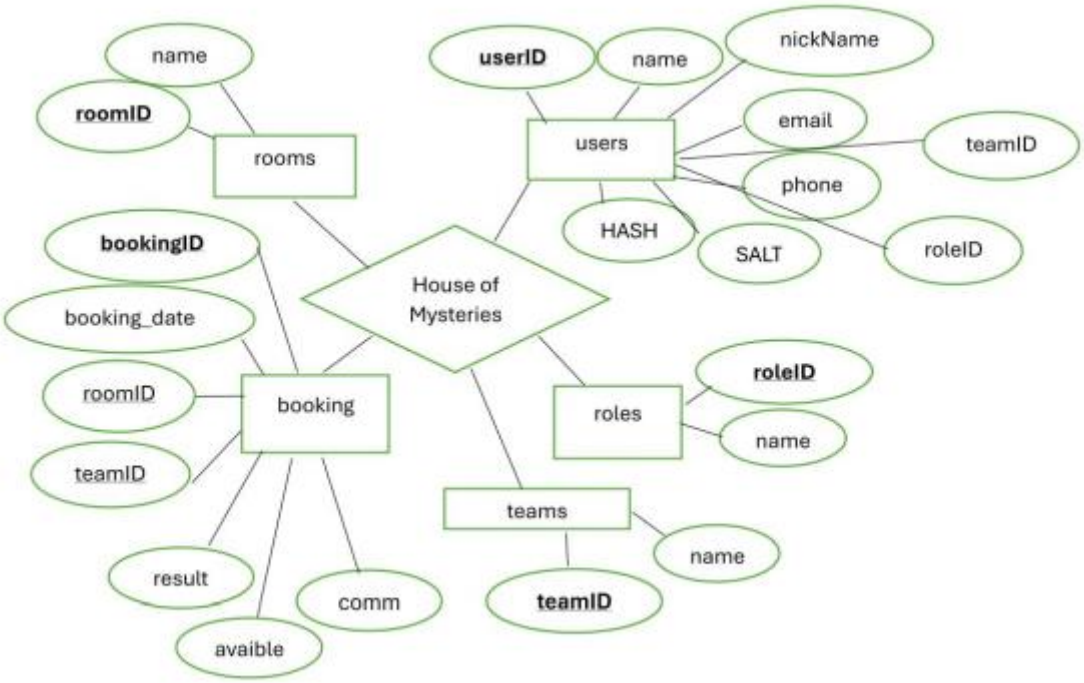
A GitHub Inc. egy amerikai nemzetközi vállalat, amely a Git segítségével szoftverfejlesztési és verziókövetés szolgáltatást nyújt. A projektünk nagyobb részelemeit ide publikáltuk, így a teljes csapat számára nyomon követhető volt, ki hol jár a vállalt feladatával.

Adatbázis

A MySQL adatbázisunkban 5 tábla található, melyekben a következő adatokat tároljuk:

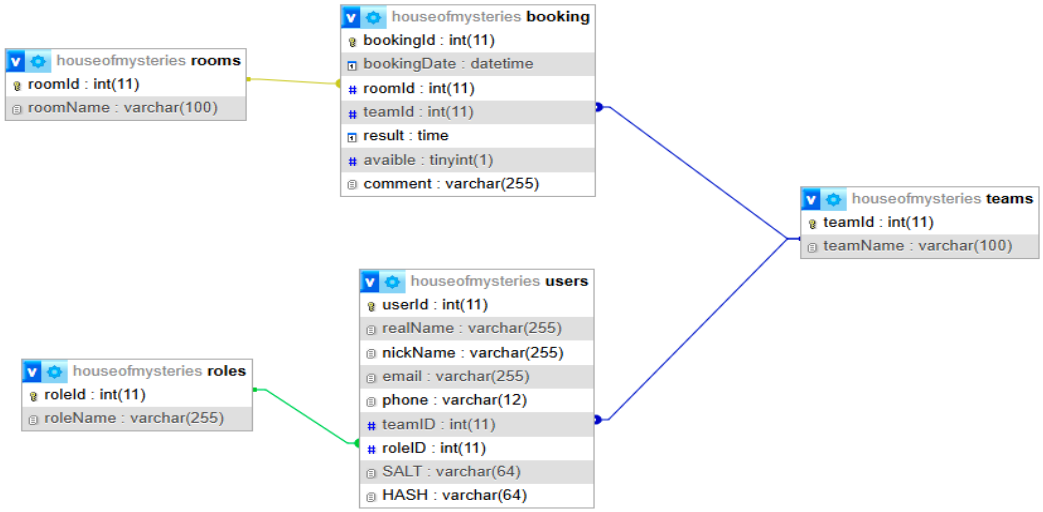
- Felhasználók (users): felhasználok regisztrációs adatait. Megkülönböztetés céljából a felhasználók más-más jogosultsági szinttel rendelkeznek.
- Csapatok (teams): mivel a felhasználók csapatot is regisztrálhatnak, szükség van egy csapat táblára is, amivel a későbbiekben azonosítani tudjuk, melyik felhasználó melyik csapatba tartozik.
- Szobák (rooms): a játszható szobák adatait tartalmazza.
- Jogosultságok (roles): itt tároljuk a felhasználók jogosultsági szintjeinek megfelelő role-ok neveit és az id-val jelöljük őket.
- Foglalás (booking): a foglalási időpontok ebbe a táblába érkeznek be frontendről. Tartalmazza a fenti táblák id-it, továbbá a versenyeredmények kiszámításához szükséges eredményeket, és kommentelési lehetőséget is biztosít.

Az adatbázis tervezésekor több adatmodellt is figyelembe vettünk, többféle EK-modellt felrajzoltunk, míg eljutottunk a végleges modellig. A célunk ezzel az volt, hogy még az adatbázis elkészítése előtt vizuálisan is felépítsük, milyen adatokkal szeretnénk dolgozni backend és frontend oldalon. Ezen a módon könnyebb volt megértenünk a tárolt adatokat, és azok kapcsolatát egymással. Végül az alábbi EK-modell alapján készült el az adatbázis:



18. ábra – ER-model

Az adatbázis létrehozásához a phpMyadmin programot használtuk. A lenti ábrán a tervezői nézetet és a táblák közötti kapcsolatot láthatjuk:



19. ábra – az adatbázis tábláinak kapcsolatai

Users tábla

userId: a felhasználó azonosítója, INT típusú, Primary Key, auto increment.

realName: a felhasználó valódi neve, regisztrációnál szükséges megadni, VARCHAR típusú.

nickName: a felhasználó választott beceneve, regisztrációnál szükséges megadni, VARCHAR típusú.

email: a felhasználó e-mail címe, amit regisztrációnál ad meg, VARCHAR típusú.

phone: a felhasználó telefonszáma, amit regisztrációnál ad meg, VARCHAR típusú.

teamId: amennyiben a felhasználó már egy csapat tagja, a teams táblából hozzárendelésre kerül a csapatnévhez tartozó teamId, INT típusú, Foreign Key.

roleId: regisztrációkor automatikusan kiosztásra kerül egy 1-es (inactive) role ID a felhasználóhoz. Amint rákattint az e-mailben kapott megerősítő oldalra, 2-es (active) felhasználó szintre vált. A további két role ID-t admin jogosultsággal lehet kiosztani, INT típusú, Foreign Key.

SALT: véletlenszerűen generált adat, amit a regisztrációkor a jelszóhoz adunk, mielőtt titkosítanánk, VARCHAR típusú.

HASH: a jelszó tárolása helyett HASH értéket tárolunk az adatbázisban, ezzel biztosítva az adatbiztonságot a felhasználók számára, VARCHAR típusú.

Teams tábla

teamId: a csapat azonosítója, INT típusú, Primary Key, auto increment.

teamName: a csapat által választott név, VARCHAR típusú.

Rooms tábla

roomId: a szabadulószobák azonosítója, INT típusú, Primary Key, auto increment.

roomName: a szabadulószobák neve, VARCHAR típusú.

Roles tábla

roleId: a jogosultságok azonosítója, INT típusú, Primary Key, auto increment.

roleName: a jogosultságok neve, VARCHAR típusú.

Booking tábla

bookingId: a foglalt szoba és időpont azonosítója, INT típusú, Primary Key, auto increment.

bookingDate: a foglalás pontos időpontja év-hónap-nap óra:perc, DATETIME formátumú.

roomId: a foglalt szoba ID-ja, INT típusú, Foreign Key.

teamId: megmutatja, melyik csapat foglalta a szobát, INT típusú, Foreign Key.

result: számláló, ami azt vizsgálja, a csapat mennyi idő alatt jutott ki a szobából, TIME típuú.

isAvailable: a szoba elérhetőségét mutatja egy, például időpont lemondás esetén újra elérhetővé válik a szoba, TINYINT/BOOLEAN típus.

comment: kommentelési lehetőség, VARCHAR típusú.

Versenyeredmények

A programunkat úgy találtuk ki, hogy a csapatok versenyezhessenek egymással szobánként. Ehhez többféle lekérdezésre is szükségünk volt, ami mind összetett információkat tartalmaz.

Az alábbi lekérdezés tartalmazza az összes szoba nevét, az első három helyezettet csapatnév szerint, illetve, hogy mennyi idő alatt jutottak ki a szobából. A lekérdezés eredménye a jobb oldalon látható. Ebből készült egy Python adatvizualizáció is, amit a későbbiekben mutatunk be.

```sql
SELECT ranked.roomName, t.teamName, ranked.result
FROM (
    SELECT
        r.roomName,
        b.teamId,
        b.result,
        RANK() OVER (PARTITION BY b.roomId ORDER BY b.result ASC) AS ranking
    FROM booking b
    JOIN rooms r ON b.roomId = r.roomId
    WHERE b.result IS NOT NULL
) AS ranked
JOIN teams t ON ranked.teamId = t.teamId
WHERE ranked.ranking <= 3
ORDER BY ranked.roomName, ranked.result;
```

20. ábra – SQL lekérdezés a 3 legjobb csapatról

roomName ▲ 1	teamName	result
A pedellus bosszúja	AdminTeam	00:51:00
A pedellus bosszúja	AdminTeam	00:59:59
A pedellus bosszúja	AdminTeam	01:05:45
A tanári titkai	Macskák2	00:04:00
A tanári titkai	Macskák	00:54:00
A tanári titkai	Macskák2	00:57:40
Menekülés az iskolából	AdminTeam	00:52:30
Menekülés az iskolából	Kiskacsa	00:55:00
Menekülés az iskolából	Bélák	00:56:20

21. ábra – a lekérdezés eredménye

A backendhez készítettünk egy egyszerűbb lekérdezést is. Ez egyetlen szoba nevét jeleníti meg, az összes hozzá tartozó csapattal, és az eredményükkel:

```sql
SELECT r.roomName, t.teamName, b.result
FROM booking b
JOIN rooms r ON b.roomId = r.roomId
JOIN teams t ON b.teamId = t.teamId
WHERE b.roomId = 1 AND b.result IS NOT null
ORDER BY b.result ASC;
```

22. ábra – SQL lekérdezés szobánként

roomName	teamName	result ▲ 1
Menekülés az iskolából	AdminTeam	00:38:45
Menekülés az iskolából	AdminTeam	00:40:25
Menekülés az iskolából	AdminTeam	00:42:15
Menekülés az iskolából	Időutazók	00:46:15
Menekülés az iskolából	AdminTeam	00:50:10

23. ábra – a lekérdezés eredménye

Adatvizualizáció

A korábbiakban említett lekérés eredményét .csv fájlba mentettük, és írtunk hozzá egy Python kódot. Ez a kód vizualizálja számunkra a versenyeredményeket oszlopdiagram formában.

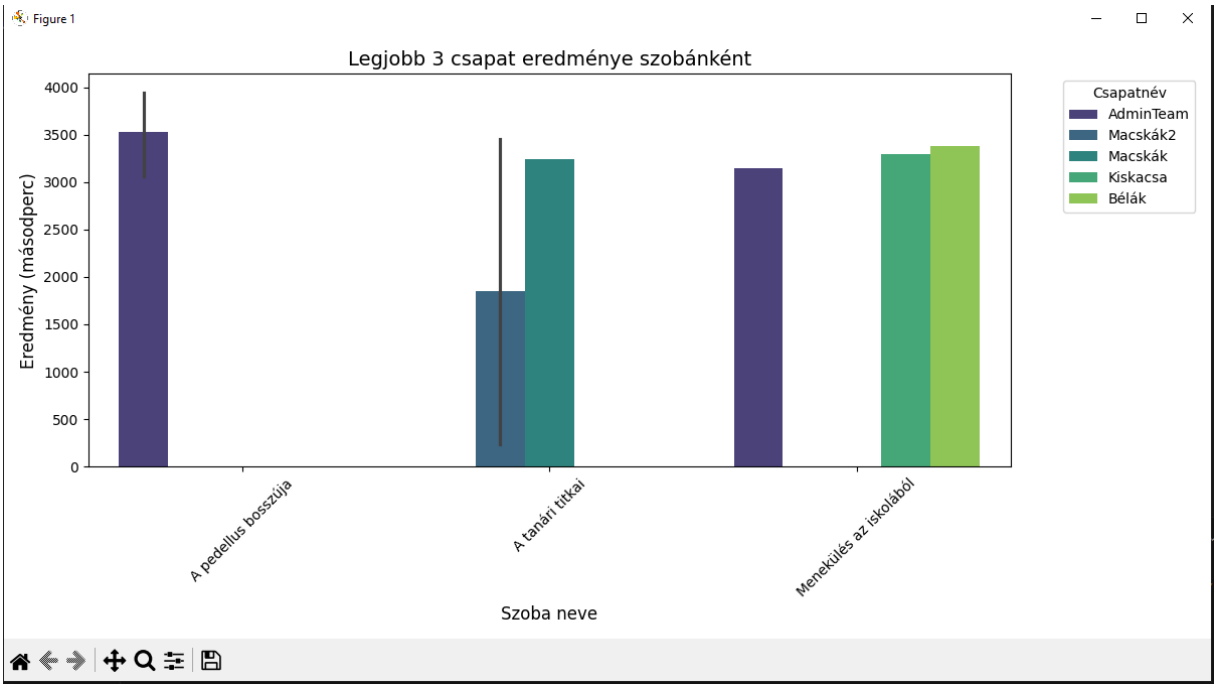
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

24. ábra – felhasznált Python osztályok

A kód megírásához három könyvtárra volt szükségünk.

- Pandas, egy nagy teljesítményű adatkezelő könyvtár, amely főként táblázatos adatok kezelésére szolgál.
- Matlpotlib.pyplot, egy adatvizualizációs könyvtár, segítségével grafikonokat és diagramokat készíthetünk.
- Seaborn, szintén vizualizációs könyvtár, ami a matplotlib alapjaira épül, szebb, részletesebb grafikákat készít.

A kódunkkal egy oszlopdiagramot készítettünk. Az x tengelyen láthatóak a szobák, az y tengelyen helyezkednek el az időadatok másodperc formátumban, maguk az oszlopok pedig a csapatokat jelölik.

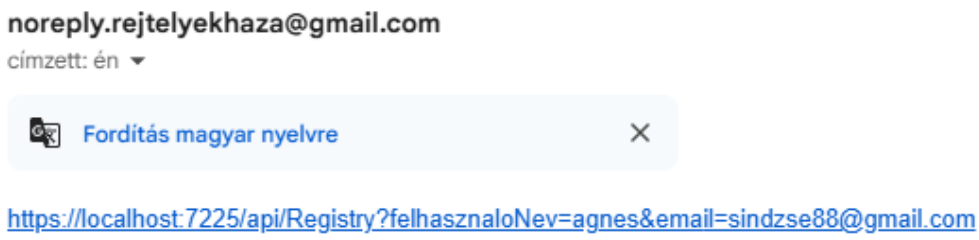


25. ábra – Python kóddal létrehozott adatvizualizáció

E-mail

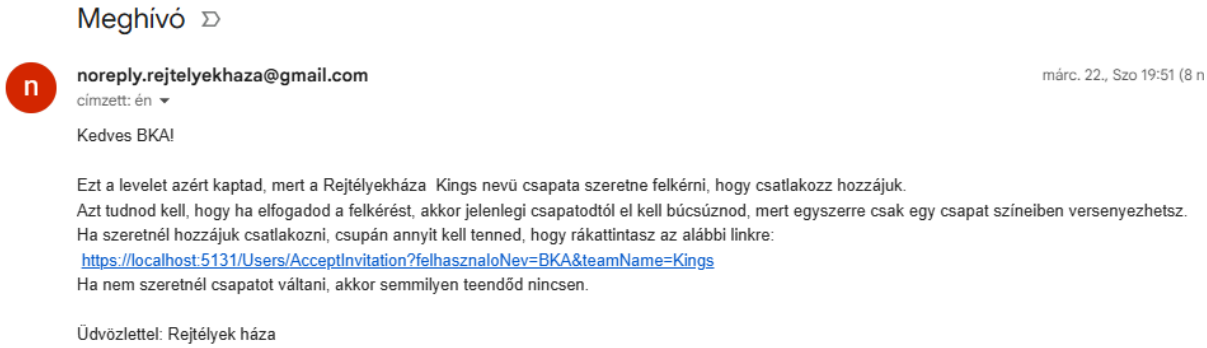
A regisztráció során automatikus e-mailt küldünk a felhasználóknak egy megerősítő linkkel. Amikor a felhasználó rákattint a linkre, a regisztráció megerősítésre kerül, így a felhasználóhoz rendelt roleId 1-esről 2-esre vált.

A megerősítő e-mail az alábbiakban látható:



26. ábra – megerősítő e-mail a regisztrációhoz

Lehetőséget adunk arra is, hogy a játékos csapatot váltson, figyelve az e-mail szövegezésében a biztonságra. Egy meghívóban jelezzük a usernek a felkérést, és amennyiben rákattint a linkre, a hozzá kapcsolódó csapatId ennek megfelelően fog átállni az adatbázisban.



27. ábra – e-mailes meghívó egy csapatba

Backend

A backend foglalkozik az adatkezeléssel, üzleti logikával, és segíti a frontend kommunikációját az adatbázissal, és biztosítja a felhasználók számára az adataik biztonságát.

Adatkezelés

A backend felelős az adatok adatbázisba helyezéséért, lekérdezéséért, módosításáért és törléséért.

Üzleti logika

A backend tartalmazza az alkalmazás üzleti szabályait és folyamatait.

Kommunikáció a frontenddel

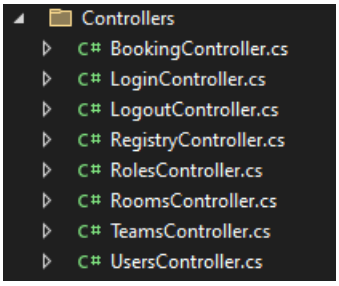
A frontend különböző kéréseket küld a backend felé, ami megválaszolja ezeket. Például, a backend fogadja a booking táblára vonatkozóan a frontend kéréseit, és megválaszolja, foglalható-e egy bizonyos szoba adott időpontban.

Biztonság

A backend felelős az alkalmazás biztonságáért, a felhasználók hitelesítéséért, és a jogosultságok kezeléséért.

Összefoglalva, a backendünk a teljes alkalmazás motorja, ami a felhasználó számára nem látható, de elengedgetetlen a helyes működéshez.

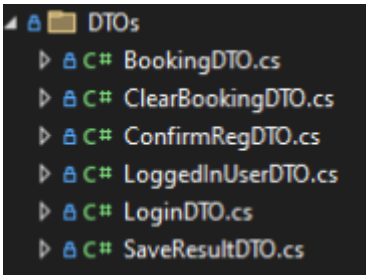
Controllers



28. ábra - Controllers

A controller fájljainkat úgy hoztuk létre, hogy teljes mértékben átlátható legyen, mit és hol találunk. Ennek segítségével nem szükséges hosszas keresgéléssel tölteni az időt, ha valamit módosítani szeretnénk, azonnal látható, hol helyezkedik el a struktúrában.

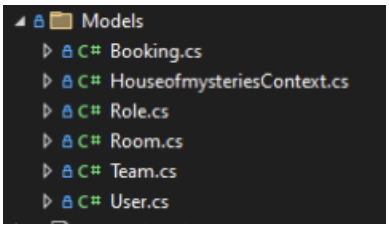
DTO-s



29. ábra - DTOs

Bizonyos funkciókat jobbnak láttunk DTO-ba szervezni. Ezzel elsődleges célunk az adatok hatékony és strukturált átvitele volt a backend különböző rétegei között. A DTO-k lehetővé teszik, hogy pontosan azokat a mezőket adjuk át, amelyre az adott rétegnek szüksége van, ezzel javítva a teljesítményt, mivel kevesebb adatot kell mozgatni a hálózaton. Itt az adatok könnyen alakíthatóak, amire szükség is volt például a Booking esetén, mert a frontend külön adja át a dátumot és az időt, viszont az adatbázis egyben kezeli a foglalási időpontot, a redundancia elkerülése érdekében.

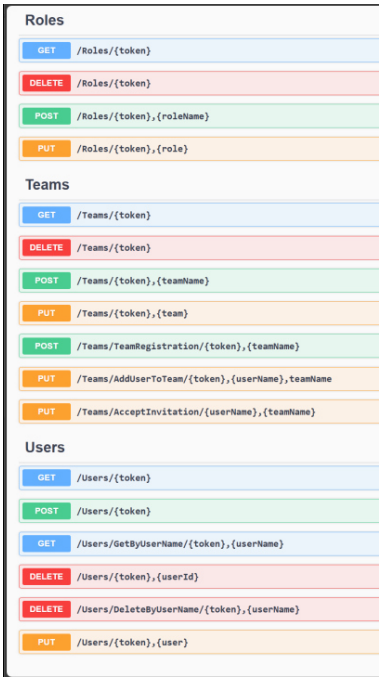
Models



30. ábra - Models

A Models mappánk tartalmazza az adatbázisból kapott adatmodelleket, és azon tulajdonságokat, amellyel az alkalmazásunk dolgozik. Mivel Entity Framework-öt használtunk, gyakorlatilag ezek az osztályok reprezentálják az adatbázisunk tábláit, és azok kapcsolatát.

Swagger



31. ábra - Swagger

A Swaggerrel folyamatosan nyomon tudtuk követni a backend-adatbázis kommunikációt, és tesztelni is tudtuk a megfelelő működést. Ezzel lényegében dokumentáltuk a backend felépülésének folyamatát.

Egyedi tokenkezelés

Különlegességünk az egyedi tokenkezelés, két osztály segítségével: CustomToken és TokenHolder.

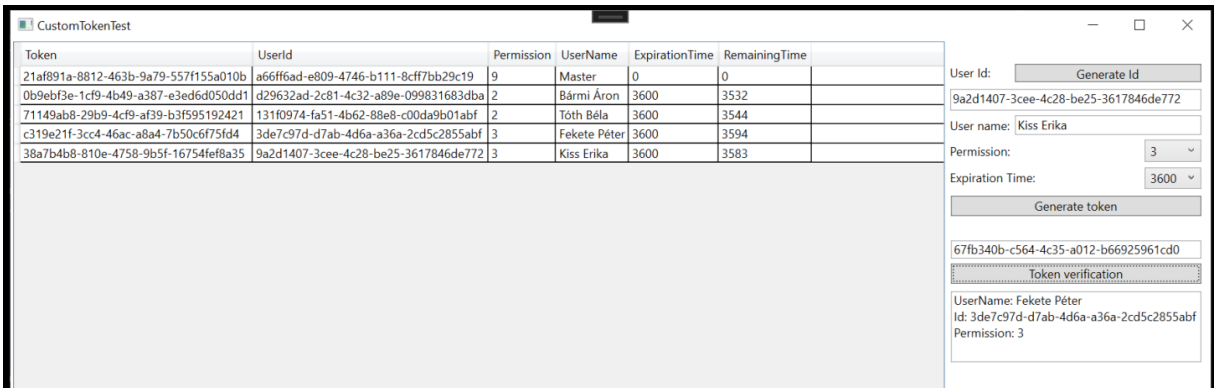
A CustomToken osztály egy adatszerkezetben tárolja a bejelentkezett felhasználó adatait, és hozzárendel egy egyedi tokent.

A TokenHolder osztály a bejelentkezett felhasználókhoz generált tokenek tárolására, validálására, és az érvényességi idő kezelésére szolgál.

A token létrehozásakor meghatározhatjuk annak érvényességi idejét (Alapértelmezetten 3600sec). Ha validáljuk a tokent, azaz valamelyik végpont felé kérést küld a felhasználó, az érvényességi idő visszaáll a kezdőértékre, vagyis ameddig a felhasználó aktívan használja a frontendet, a token érvényes marad. Az érvényességi idő lejárta után a token és a hozzárendelt felhasználói adatok autómatikusan törlődnek, és ha validáljuk a lejárt tokent, egy üres felhasználói profilt kapunk, érvénytelen jogosultsággal.

A TokenHolder osztály példányosításakor annak konstruktora létrehoz egy master tokent, ami a legmagasabb jogosultsági szinttel rendelkezik, és ezáltal lehetőség van akkor is használni a backend-et, ha az adatbázis nincs feltöltve érvényes adatokkal. Természetesen ezt a tokent soha nem továbbítjuk a frontend felé, csak az admin ismeri.

Mellékeltünk egy tesztprogramot, aminek segítségével betekinthetünk a tokenkezelés folyamatába, és tesztelhetjük a token használatot működés közben.

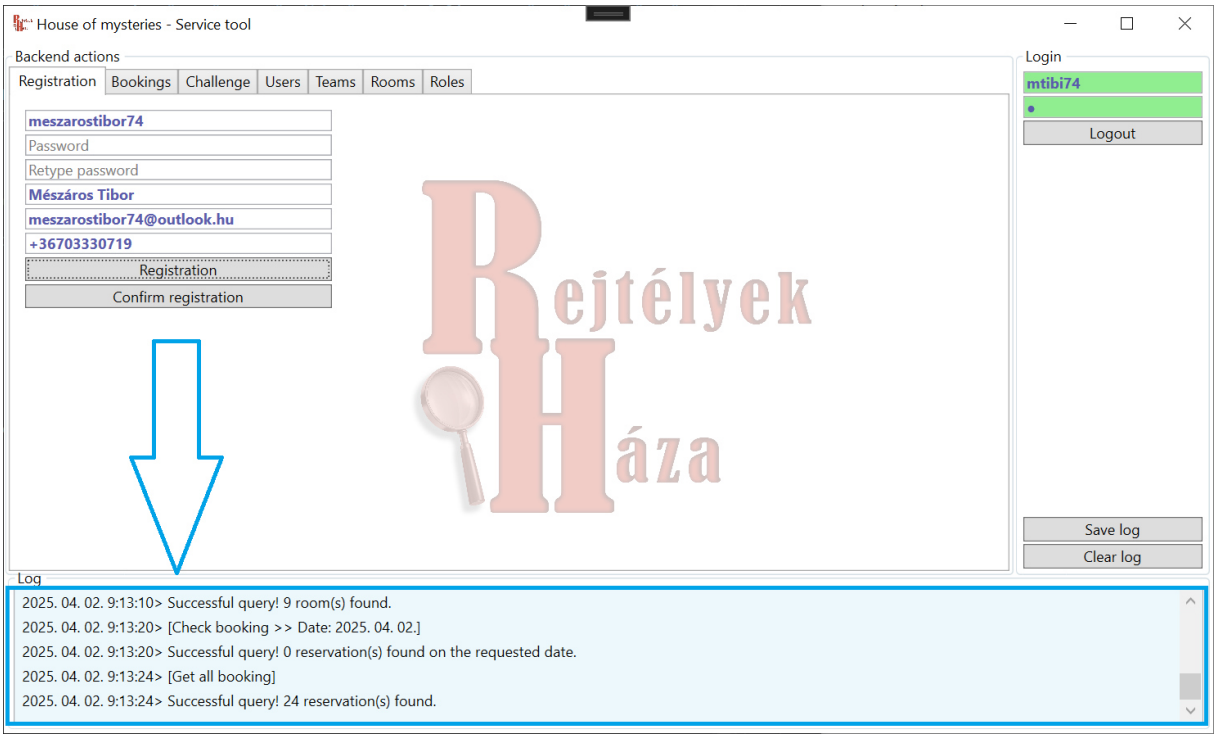


32. ábra – tokenkezelés a WPF alkalmazásban

Asztali alkalmazás

Az adminisztrátorok számára létrehoztunk egy WPF asztali alkalmazást is, a foglalások egyszerűbb nyomon követhetősége érdekében.

A wpf szervízprogram kialakításakor a funkcionalitás mellett fontosnak tartottuk, hogy könnyen kezelhető, felhasználóbarát kezelőfelületet hozzunk létre. Ahol csak tehettük, kerültük a MessageBox használatát. Erre a feladatra létrehoztunk egy LogWindow-t, ami rögzíti a felhasználói beavatkozásokat, az ezekre kapott válaszokat, esetleges hibaüzeneteket. Ennek előnye, hogy az eltűnő üzenetek helyett egy dátumbélyegzővel ellátott eseménylista segítségével dokumentáljuk a teljes munkamenetet.



33. ábra – az asztali alkalmazás belépési felülete

Természetesen lehetőség van egy fájlba menteni a LogWindow teljes tartalmát, így később kiértékelhetjük a felhasználó által végzett beavatkozásokat.

34. ábra – logok

A LogWindow használata felvetett egy problémát. Mivel az aszinkron műveletek külön szálon futnak, nem tudtuk az eredményt közvetlenül a LogWindow-ba írni a kérés hívásakor, hiszen a kért művelet akkor még nem fejeződött be.

A megoldás egy saját eseménykezelő osztály létrehozása volt: SendLogEvent.cs.

Ennek a használata nagyon egyszerű: minden egyes service-ben példányosítjuk az osztályt, így lehetőségünk van a feladatok végrehajtásának a befejezésekor eseményt generálni, amihez hozzárendelhetjük az átadni kívánt üzenetet.

A gyakorlatban a következőképpen néz ki a megvalósítása:

első lépésként példányosítjuk az osztályt.

```
public class UserService
{
    public static SendLogEvent sendLogEvent = new SendLogEvent();
```

35. ábra – kódrészlet1

Amikor az aszinkron művelet befejeződött, eseményt generálunk:

```
if (response.IsSuccessStatusCode)
{
    sendLogEvent.SendLog(content);
```

36. ábra – kódrészlet2

Ezután már csak annyi a dolgunk, hogy a MainWindow.xaml.cs-ben feliratkozunk erre az eseménykezelőre:

```
UserService.sendLogEvent.LogSent += SendLogEvent_LogSent;
```

37. ábra – kódrészlet3

Majd feldolgozzuk az eseményt:

```
private void SendLogEvent_LogSent(object sender, string e)
{
    WriteLog(e);
}
```

38. ábra – kódrészlet4

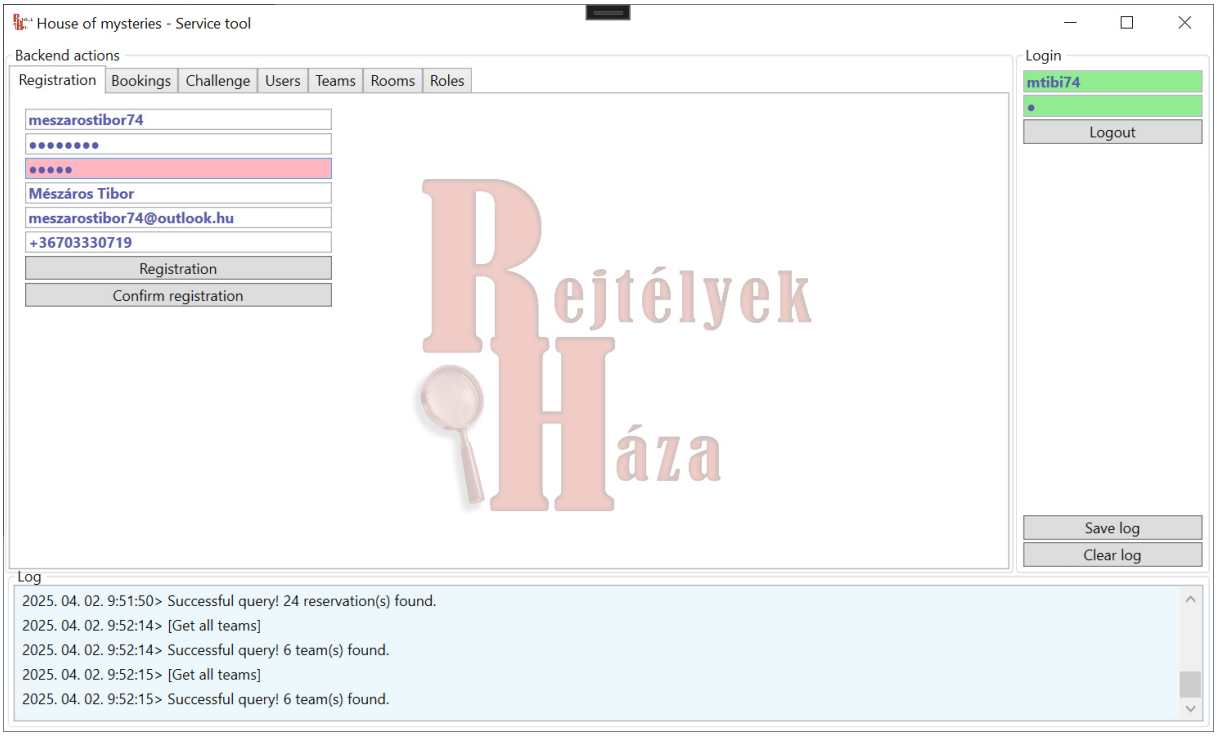
Így bármelyik service-ből könnyen tudunk üzenetet küldeni a LogWindow-ba.

Másik probléma volt, a felhasználótól érkező hibás információk kiszűrése, mielőtt az futási hibát okozna.

Természetesen egyszerűen megoldhatjuk ezt kivételkezeléssel, de inkább egy, a felhasználó számára is barátságosabb megoldást használtunk.

A beviteli mezőben minden billentyűlenyomáskor megvizsgáljuk a bejövő adatot, és azonnal kiszűrjük a hibát okozó karaktert. Így biztosan nem tud például egy szám típusú adatot váró mezőbe betűt írni.

Ha formailag meg is felel az adat, de tartalmi hibát találunk benne (esetleg üresen marad), azt az adott mező színének megváltoztatásával jelezzük a felhasználónak, így azonnal látja, hol kell javítania, hogy végrehajtható legyen a kérése.

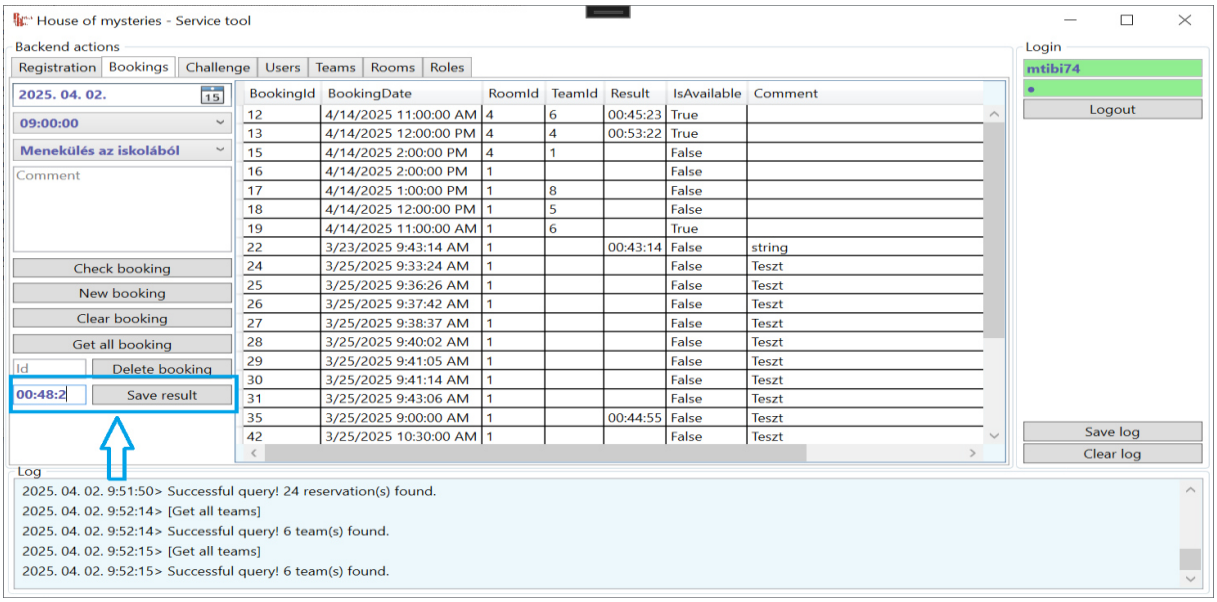


39. ábra - a rózsaszínnel jelzett területen látható, hogyan jelezzük a hibás adatbevitelt

Ha nem a billentyűzet segítségével akarná bevinni az adatokat a felhasználó, akkor úgy akadályozzuk meg az esetleges hibás adatbevitelt, hogy tiltjuk az egérrel, vagy billentyűkombinációval történő beillesztést.

A legnehezebb dolgunk a játékeredmény idejének a bevitelénél volt, mert a számokon kívül egyéb karakterek, például kettőspontok is szerepelnek az adatformátumban, és ezek csak adott pozícióban helyezkedhetnek el.

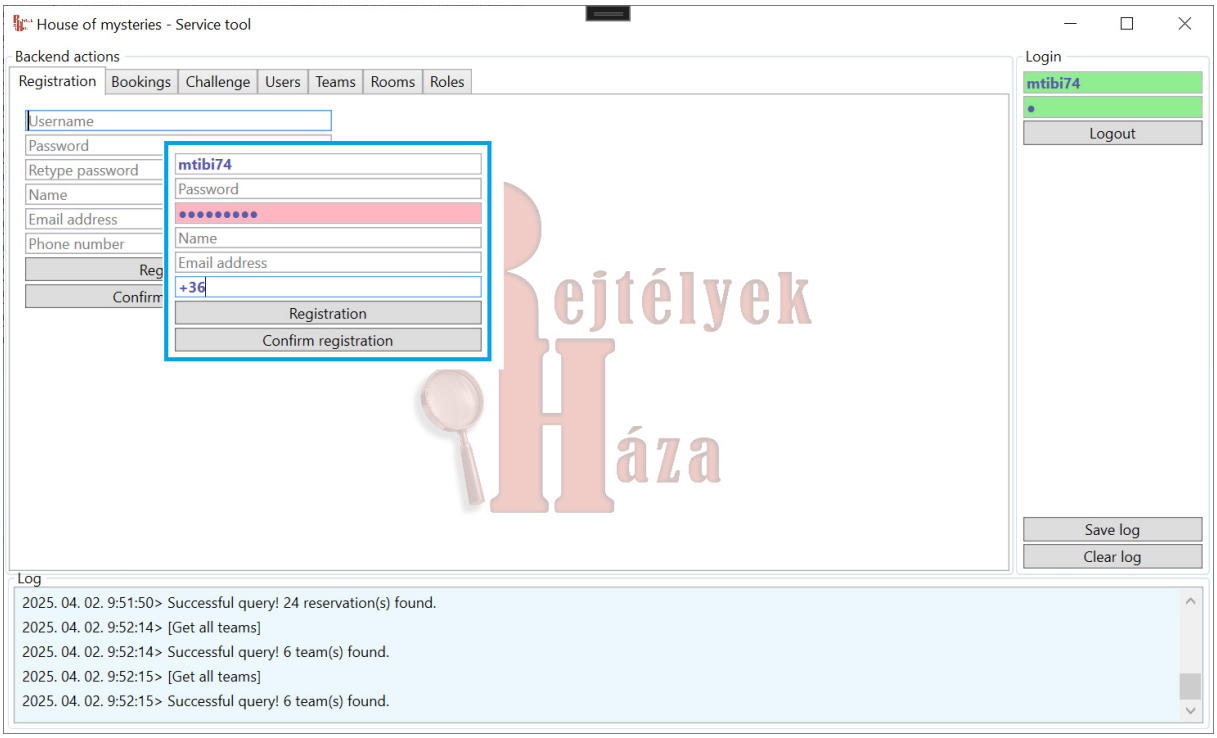
Ezt több billentyű esemény felhasználásával és kettőspont automatikus beszúrásával oldottuk meg. Ennek köszönhetően a felhasználó számára is egyszerűbbé vált az idő megadása.



40. ábra – a játékosok idejének felvitele

Letisztultabb, és áttekinthetőbb felületet próbáltunk létrehozni azáltal, hogy a beviteli mezők funkcióját tartalmazó címkét nem a beviteli mezőn kívül helyeztük el, hanem vele azonos pozícióban, a beviteli mező mögött, amit áttetszővé tettünk.

Amint a felhasználó elkezdi az adatbevitelt, a hátteret aktiválva eltakarjuk az információs sávot.



41. ábra – a beviteli mezők átláthatósága

Frontend

A frontend felelős az alkalmazás grafikus felhasználói felületének (röviden GUI, esetünkben webes alkalmazás, honlap) megjelenítéséért. Kezeli a felhasználói interakciókat, továbbítja a felhasználók által rögzített adatokat a backend felé, biztosítja a gombokat az adatok lekéréséhez, hozzáadásához, módosításához, törléséhez, segíti a listából való kiválasztás lehetőségét, kommunikál a backenddel.

Felhasználói felület megjelenítése

Webes felületen lehetővé teszi, hogy a felhasználó interaktív felületen használhassa a programunkat. Biztosítja a regisztrációt, időpontfoglalást, csapatversenyek nyomon követését, ezen kívül az admin felület a dolgozóknak biztosítja az üzleti háttér működtetését.

Felhasználói interakciók kezelése

A felület kezeli a felhasználói igényeknek megfelelően a weboldal működését. Különböző gombokkal, textboxokkal és legördülő menükkel segíti a GUI használatát a userek számára.

Kommunikáció a backenddel

A frontendünk kommunikál a backenddel, adatokat kér le, vagy küld el. Például a regisztráció során a felhasználó által a frontenden megadott adatokat a backend segítségével továbbítja az adatbázisba, vagy bejelentkezéskor a már regisztrált felhasználó adatait kéri le.

A frontend tehát az alkalmazásunk arca, amivel a felhasználó találkozik.

Főoldal

42. ábra – a weboldalunk főoldala

A Főoldalunkat igyekeztünk a lehető legfigyelemfelkeltőbb módon létrehozni. A szabadulószobák weboldalainak hagyományaihoz híven sötét hangulattal dolgoztunk. A navbar bármelyik aloldalról elérhető, így a felhasználók könnyedén válthatnak a különböző felületek között.

Rólunk

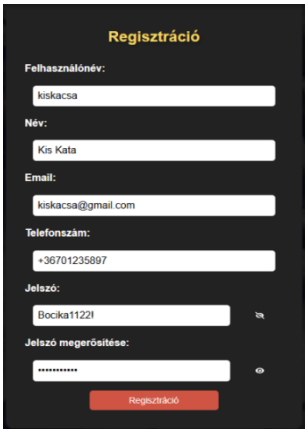
A Rólunk felületen bemutatjuk a csapatunk elképzelését a szabadulószobák világával kapcsolatban, hogy leendő ügyfeleink megismerhessenek bennünket és a céljainkat. A projekt során fontosnak éreztük, hogy egy élesben is helytálló termékcsomaggal készüljünk fel, így üzletileg is átgondoltuk, milyen lehet egy jó szabadulószoba honlapja.

Szobák

Kezdetnek 9 szobával számoltunk, amiből jelenleg mind foglalható, köszönhetően a rendszerünk teljes együttműködésének. Érdekesség, hogy korábban, más struktúrával dolgozva ezt nem tudtuk volna megtenni. Csempés megoldással dolgoztunk, így az ügyfelek számára teljes mértékben átlátható felületet hoztunk létre.

43. ábra – a szobák megjelenítése a weboldalon

Regisztráció



44.ábra – regisztrációs felület

Regisztrációs felületünkön egyszerűen és hatékonyan tudnak a felhasználók regisztrálni. A megfelelő adatok kitöltése után a gombra kattintva megerősítő e-mailt küldünk*.

A megerősítő e-mailben küldött linkre kattintás után a felhasználó beléphet, csapatot alakíthat, játszhat a versenyben.

Bejelentkezés

A bejelentkező felületünk nagyon hasonlít a fenti, regisztrációs felületre, viszont csak felhasználónevet és jelszót kér. Attól függően, hogy a felhasználó milyen role-lal rendelkezik, dobja be a megfelelő felületre a rendszer. Tehát, amennyiben a felhasználónk user role-lal rendelkezik, abban az esetben a felhasználóhoz dobja be a felület, amennyiben pedig kolléga, vagy admin role-lal, a lent bemutatásra kerülő adminisztrációs felületre.

Admin felület

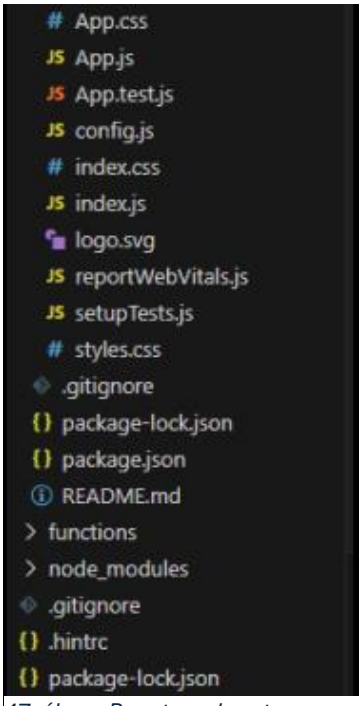
Az Admin felület leegyszerűsíti a szobák kezelőinek munkáját, egyszerűvé teszi az ügyfelek kezelését. Egyetlen kattintásra lekérhető az adatbázisból az összes felhasználó, a jogosultságaik, illetve a hozzájuk kapcsolódó csapat ID, amennyiben rendelkeznek ilyennel.

A versenyeredmények felvitele az admin feladata, aki ki tudja választani a szobát, a foglalás időpontját, és be tudja írni a csapat elért eredményét. Ez a booking tábla result rekordjába kerül a megfelelő sorba.

45. ábra – felhasználók lekérése az admin felületen

46. ábra – a versenytábla adatbeviteli felülete

React szerkezet



47. ábra – React szerkezet

A frontend felület létrehozásához React keretrendszert használtunk, mert ezzel látványos, dinamikus weboldalakat lehet létrehozni, emellett biztosítja az oldal stabilitását. A React keretein belül React Router DOM, React Icons, uuidv4, cryptoJS és Axios került feltelepítésre.

A React Router DOM egy npm-csomag, ami lehetővé teszi a dinamikus útválasztás megvalósítását egy webalkalmazásban, segíti az oldalak megjelenítését és segíti a felhasználókat a navigálásban. Ez egy teljeskörű kliens- és szerveroldali útválasztó könyvtár.

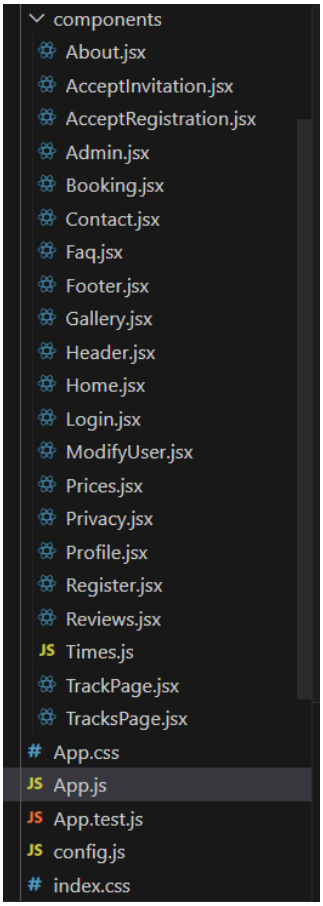
A React Icons egy olyan könyvtár, ami lényegében egy gyűjtemény népszerű ikoncsomagokból. Ezeket komponensként beépíthetjük a React alkalmazásunkba, anélkül, hogy képeket importálnánk.

Az uuidv4 metódussal értük el az egyedi azonosítók létrehozását regisztrációkor, ezzel biztosítva a felhasználók adatainak biztonságát. A metódus véletlenszerűen generálja az azonosítókat, rendkívül alacsonyra csökkentve az esélyét annak, hogy két különböző hívás ugyanazt az értéket adja vissza.

A cryptoJS egy másik Reactos könyvtár, ami kiterjedt kriptográfiai algoritmusokat kínál. Lehetővé teszi, hogy biztonságos kriptográfiai műveleteket végezzünk a böngészőnkben.

Az Axios a háttérkommunikációt segítő könyvtár. Többnyire http-kérések küldésére és REST-végpontokhoz használják. Leegyszerűsíti a kommunikációt a kliens és a szerver között.

Saját komponensek



48. ábra – saját komponensek

A fentebb bemutatott, feltelepített komponensek mellett az oldalunk saját komponenseket is tartalmaz, minden funkcióra külön-külön.

Ezekkel értük el, hogy az oldalunk funkcionalitása a terveknek megfelelő legyen.

A korábban leírt és bemutatott komponenseken túl tartalmazza például a Gyakran Ismételt Kérdések, Árak, Galéria, és a jelenleg hatályos törvények szerint nagyon fontos Adatvédelmi szabályzatot.

Fontos komponensek a helyes működés tekintetében az AcceptRegistration, mert ezen a komponensen keresztül, a backend által e-mailben kiküldött linken tudja megerősíteni a regisztrációját a felhasználó, illetve az AcceptInvitation, amivel egy csapat meghívását tudja elfogadni a játékos az előbb leírt módon.

Funkcionálisan fontosak még a ModifyUser és a Register fájlok, amik nevükből adódóan a regisztrációhoz és az adatmódosításhoz adnak támogatást a játékosnak.

A szobák kezelői számára létrehoztunk egy Admin komponenst is, ez segíti őket a napi munkavégzés során. Ide nem kizárólag ügyfél adatokat tudnak felvinni, hanem csapatot is tudnak regisztrálni, le tudják kérni a szobák használhatóságát, vagy éppen, ha például áramszünet van, karbantartásra tudják állítani a szoba elérhetőségét, ezzel elérve, hogy a játékosok ne tudjanak foglalni. Mindezeken felül ellenőrizni tudják a szobák aktuális foglaltságát, le tudják kérni a versenyeredményeket az adatbázisból, és telefonszámot nyerhetnek ki, ha a játékoscsapat késne, vagy egyéb okból keresnie kellene őket.

Tesztelés

Az elkészült weboldalunk funkcionalitását Python kóddal írt Selenium teszttel ellenőriztük. A Bejelentkezés oldalra írtuk meg a tesztet. A teszt során a kód képernyőfotókat készít az egyes lépések elvégzéséről, így láthatóvá válik a teszt sikeressége, illetve az oldal működése.

A teszthez több könyvtárat is importáltunk, hogy megfelelően tudjon működni. Ebbe tartozik maga a Selenium webdriver, ami a böngészőt vezérli, a Keys, ami billentyűzet eseményeket szimulál, és a By, amivel különböző elemeket kereshetünk az oldalon. Ezen kívül a Time könyvtárat importáltuk, hogy a kód futtatásakor látható legyen, mi történik a böngészőben:

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.by import By
import time
```

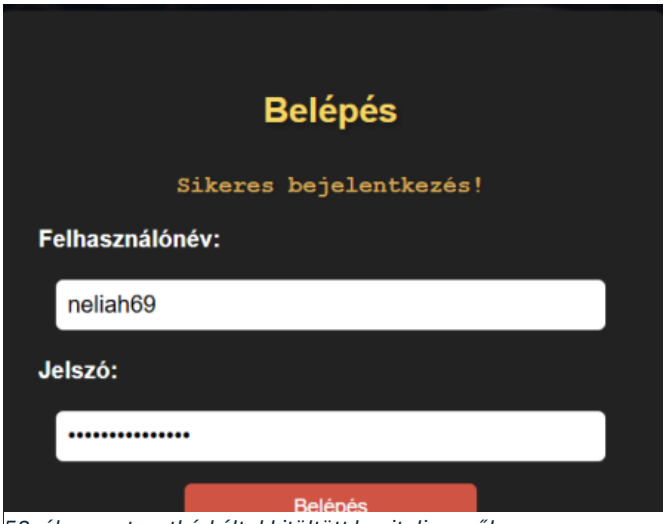
40. ábra – felhasznált python könyvtárak

A tesztkód megnyitja az oldalt (http://localhost:3000/).képernyőképet készít a főoldalról, majd elnavigál a Bejelentkezés gombra, és rákattint.

41. ábra – a tesztkód által megnyitott főoldal

51. ábra – a tesztkód által fotózott belépési oldal

Ez után a kitöltendő mezőket az előre megadott értékekkel kitölti, és belép az oldalra:



52. ábra – a tesztkód által kitöltött beviteli mezők

Források

https://learn.microsoft.com/hu-hu/dotnet/csharp/fundamentals/tutorials/oop

https://learn.microsoft.com/hu-hu/shows/entity-framework-core-for-beginners/

https://www.apachefriends.org/faq_windows.html

https://www.geeksforgeeks.org/reactjs-router/

https://tutorials.eu/data-transfer-objects-in-c-sharp/

https://learn.microsoft.com/en-us/dotnet/architecture/maui/mvvm

https://www.selenium.dev/

https://www.python.org/

Ábrajegyzék