|  |  |
| --- | --- |
|  | **Eötvös Loránd Tudományegyetem**  **Informatikai Kar**  **Algoritmusok és Alkalmazásaik Tanszék** |

Teremfoglaló rendszer

**Nagy Sára Tomecz Patrik**

mesteroktató Programtervező informatikus BSc

nappali tagozat

**Budapest, 2019**

Tartalomjegyzék

[1. Bevezetés 3](#_Toc529885656)

[2. Felhasználói dokumentáció 4](#_Toc529885657)

[2.1 A célközönség 4](#_Toc529885658)

[2.2 A hardveres és szoftveres követelmények 4](#_Toc529885659)

[2.3 A használati útmutató 5](#_Toc529885660)

[2.3.1 Látogatók számára elérhető felület 5](#_Toc529885661)

[2.3.2 Bejelentkezett felhasználók (tanárok) számára elérhető felület 13](#_Toc529885662)

[2.3.3 Adminisztrátorok számára elérhető felület 17](#_Toc529885663)

[3. Üzemeltetői dokumentáció 20](#_Toc529885664)

[3.1 Hardveres követelmények 20](#_Toc529885665)

[3.2 Szoftveres követelmények 21](#_Toc529885666)

[3.3 Telepítés 21](#_Toc529885667)

[4. Fejlesztői dokumentáció 22](#_Toc529885668)

[4.1 A fejlesztés menete 22](#_Toc529885669)

[4.2.1 A fejlesztői környezet 22](#_Toc529885670)

[4.2.2 Verziókezelés 22](#_Toc529885671)

[4.3 A felhasznált technológiák 23](#_Toc529885672)

[4.4 Az alkalmazás architektúrája 25](#_Toc529885673)

[4.4.1 A backend 25](#_Toc529885674)

[4.4.2 A frontend 27](#_Toc529885675)

[4.1 Tesztelés 27](#_Toc529885676)

[4.1.1 Backend 27](#_Toc529885677)

[5. Irodalomjegyzék 31](#_Toc529885678)

# 1. Bevezetés

Napjainkban nagy népszerűségnek örvend a webes alkalmazások fejlesztése.

A szakdolgozatom témája egy ilyen alkalmazás, egy teremfoglaló rendszer elkészítése. A rendszer megvalósítását modern, a tanulmányaim során megismert eszközökkel vittem véghez. Törekedtem arra, hogy a felület könnyen átlátható, felhasználóbarát legyen. A fejlesztés közben az alkalmazást folyamatosan teszteltem a hibák elkerülése végett.

A program célja, hogy a hallgatók számára megkönnyítse az egyes órákra, eseményekre vonatkozó foglalások közötti keresést, valamint, hogy segítse a tanárok munkáját a letisztult foglalási felület segítségével.

Az alkalmazás felületének létrehozása közben szem előtt tartottam azt is, hogy bárki számára könnyen kezelhető legyen. A vendégek megtekinthetik az aktuális foglalásokat, mind táblázat, mind naptár nézetben. Lehetőség nyílik a foglalások közötti keresésre is. A tanárok/felhasználók leadhatnak új tanóra-, esemény foglalásokat, szerkeszthetik saját foglalásaikat. Az adminisztrátor felvehet új épületeket, tantermeket, szemesztereket, tantárgyakat és foglalásokat, valamint lehetősége nyílik ezek szerkesztésére is, továbbá menedzselheti az éppen várakozó kéréseket.

# 2. Felhasználói dokumentáció

## 2.1 A célközönség

Az alkalmazás célközönsége elsősorban a felsőoktatásban tanuló hallgatók és tanárok. A cél, hogy könnyebbé, átláthatóbbá tegye a diákok számára a tanórákhoz, eseményekhez tartozó szükséges információk megszerzését, megkönnyítse az órarend összeállítását. Ezzel együtt a tanárok számára egyszerű felületet nyújt a foglalások leadására és azok szerkesztésére is.

## 2.2 A hardveres és szoftveres követelmények

A futó alkalmazás használatához az egyetlen követelmény egy webböngésző program.

Ajánlott webböngészők:

* Mozilla Firefox 62.0
* Google Chrome

A Mozilla Firefox rendszerkövetelményei Windowson:

* Operációs rendszerek:
  + Windows 7
  + Windows 8
  + Windows 10
* Hardveres követelmények:
  + Pentium 4-es vagy újabb processzor, amely támogatja az SSE2-t
  + 512MB memória / 2GB memória a 64-bites verzió számára
  + 200MB szabad tárhely

A rendszerkövetelmények Macen:

* Operációs rendszerek:
  + macOS 10.9
  + macOS 10.10
  + macOS 10.11
  + macOS 10.12
  + macOS 10.13
* Hardveres követelmények:
  + Intel x86 processzor
  + 512 MB memória
  + 200 MB szabad tárhely

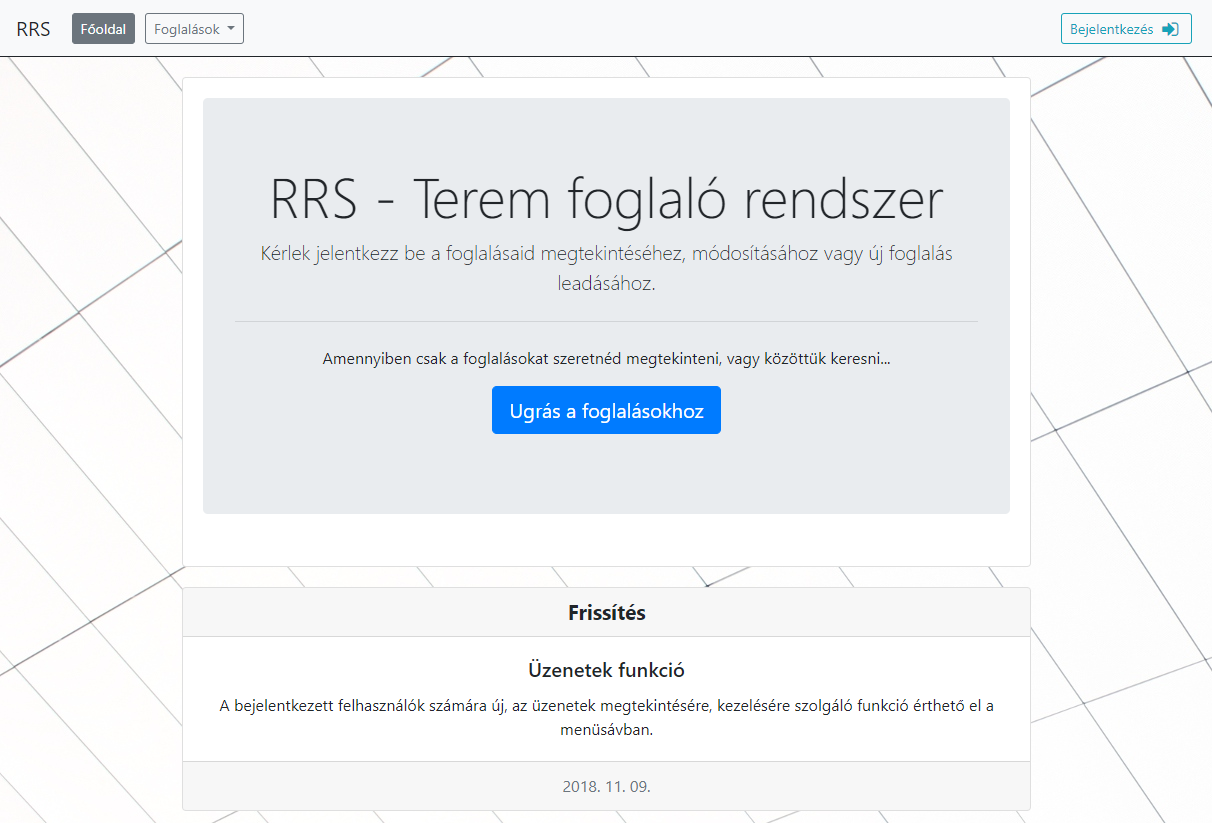
A Google Chrome követelményei hasonlóak.

## 2.3 A használati útmutató

### 2.3.1 Látogatók számára elérhető felület

#### 2.3.1.1 Főoldal

Az weboldal megnyitásakor a kezdőlap tárul a látogató szeme elé. A főoldalon értesülhetünk a hírekről, frissítésekről és a fontos tudnivalókról.



1. ábra A főoldal

#### 2.3.1.2 Navigáció

A felső állandó sávban található a menü, alatta pedig a weboldal tartalma. A menüsávban a szürke színnel történik az aktuális oldal jelölése. Kétféle menüpont található: sima és legördülő menü. A menü dinamikusan változik attól függően, hogy történt-e bejelentkezés.

A be nem jelentkezett felhasználók számára az alábbi menüsáv érhető el:



2. ábra A vendégek számára elérhető menüsáv

Bejelentkezés után attól függően, hogy milyen engedélyekkel rendelkezik az adott felhasználó, az alábbi két menü közül fogadja az egyik:



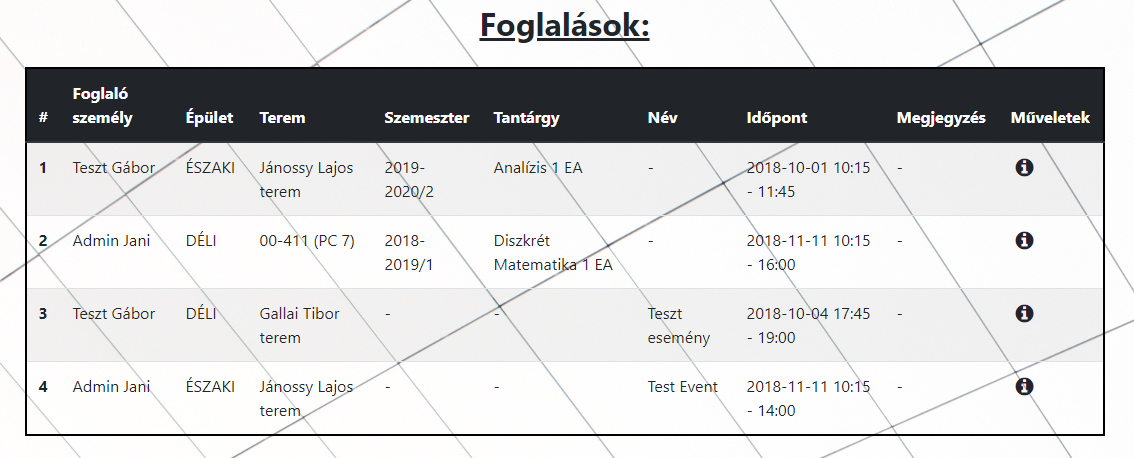
3. ábra A felhasználók számára elérhető menüsáv



4. ábra Az adminisztrátorok számára elérhető menüsáv.

#### 2.3.1.3 Foglalások

A foglalások menüpont alatt, mely egy legördülő menü, található minden, az elfogadott foglalásokkal kapcsolatos általános funkció. Az első menüpontra kattintva a foglalások táblázatos formában jeleníthetjük meg.

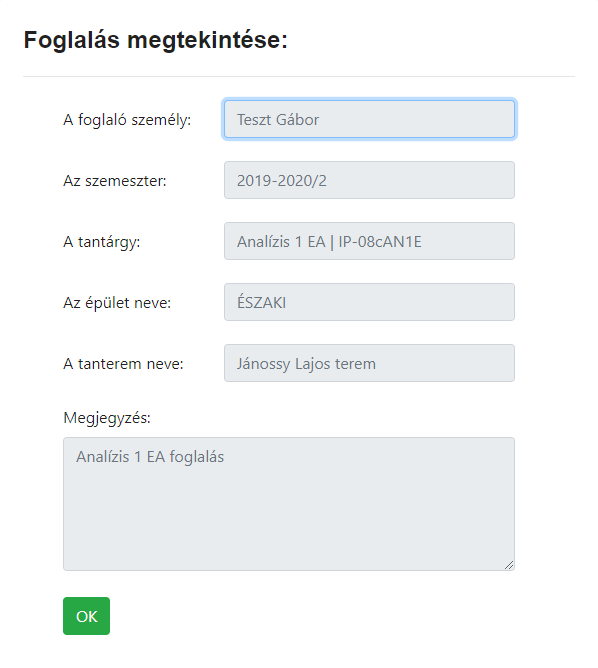


5. ábra Az elfogadott foglalások táblázatos nézetben

A táblázat egy sora egy adott foglalást jelöl és az alábbi információkat tartalmazza:

* A foglalás sorszáma. Fontos, hogy ez nem a foglalás azonosítója, ez csak a táblázaton belüli sorszám jelölésére szolgál.
* A foglaló személy neve
* Az épület neve
* A tanterem neve
* A szemeszter, ha tanórára vonatkozó foglalásról van szó
* A tantárgy neve, ha tanórára vonatkozó foglalásról van szó
* Az esemény neve, ha eseményre vonatkozó foglalásról van szó
* A foglalás időpontja
* A megjegyzés
* Az adott eseményre vonatkozó műveletek

A „Műveletek” oszlopban a „Részletek” gombra kattintva egy információs ablak jelenik meg, amely tartalmazza az adott foglalás részleteit. Ez az ablak némileg eltérhet attól függően, hogy tanórára vagy eseményre vonatkozó foglalást vizsgálunk.



6. ábra Tanórára vonatkozó foglalás megtekintése

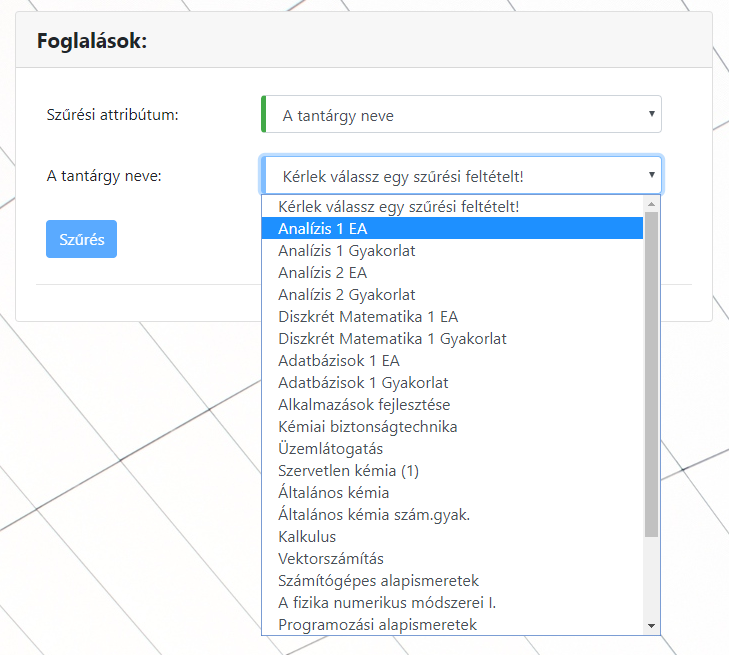


7. ábra Eseményre vonatkozó foglalás megtekintése

A második pontot választva a felhasználóbarát naptári nézethez navigálhatunk. Ahhoz, hogy a foglalások megjelenjenek először meg kell adnunk valamilyen szűrési feltételt az alábbiak közül:

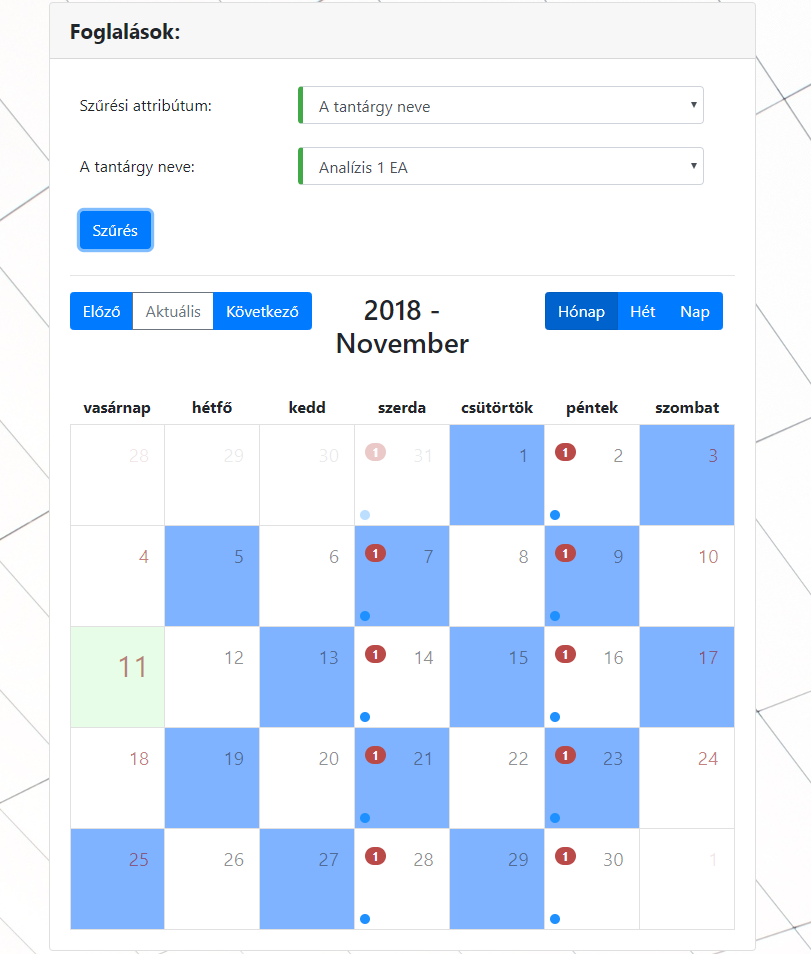
* A foglaló személy neve
* Az épület neve
* A tantárgy neve
* A szemeszter
* Az esemény neve

Abban az esetben, ha a szűrési attribútum kiválasztása sikeres, megjelenik egy második legördülő menü, ahol kiválaszthatjuk a kívánt értéket.



8. ábra A foglalásokra szűrése

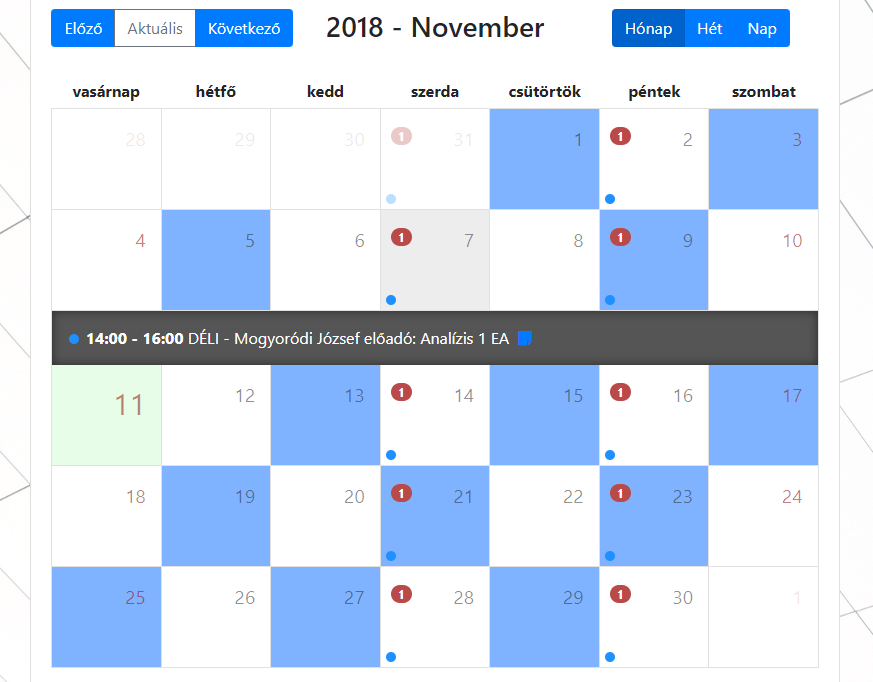
Az érték kiválasztása után, a „Szűrés” gomb aktiválódik, rá kattintva megtörténik a keresés. Ha a keresés sikeres volt, akkor lent megjelenik a Naptár nézet a megfelelő foglalásokat tartalmazva.



9. ábra Naptár (havi) nézet

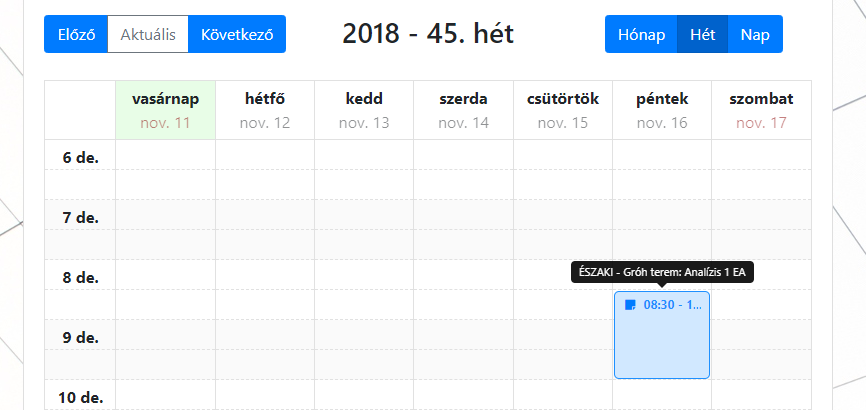
A naptár nézet több, a vendég számára hasznos, funkciót tartalmaz. A naptár lapozása az „Előző” és „Következő” gombok segítségével történik. Az „Aktuális” gombra kattintva a naptár aktuális naphoz tartozó oldala kerül betöltésre. A nézetek váltása a „Hónap”, „Hét” és „Nap” gombok segítségével valósítható meg.

A havi nézet esetén minden egyes napnál, ahol található valamilyen esemény, megjelenik egy piros háttérrel rendelkező ovális gömb. A gömbben található az adott napra vonatkozó foglalások darabszáma. A napra kattintva megtekinthetővé válik a foglalások kezdete, vége, az épület, a tanterem és a foglalás neve. A bővebb részletek megtekinthetők a foglalás neve melletti ikonra kattintva.



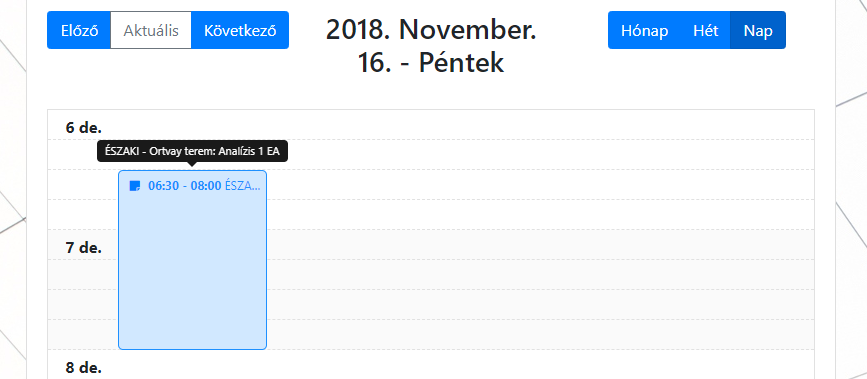
10. ábra A havi nézet

A heti nézetet megnyitva az összes, adott héthez tartozó, foglalás jelenik meg a felhasználó szeme előtt.



11. ábra A heti nézet

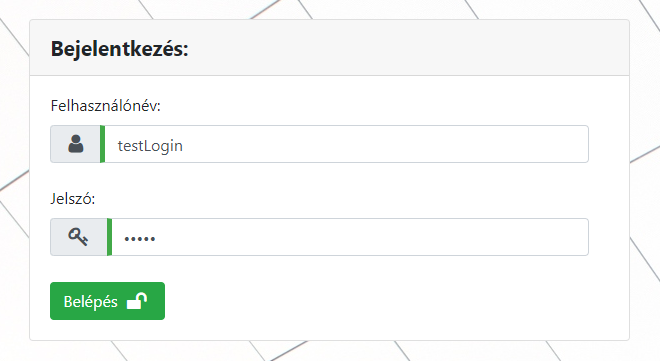
A „Nap” gombra kattintva az aktuális napra vonatkozó információk kerülnek megjelenítésre.



12. ábra A napi nézet

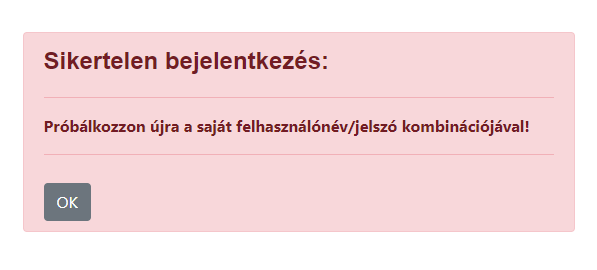
#### 2.3.1.4 Bejelentkezés

A menüsáv jobb szélén található a bejelentkezés gomb, mely a megfelelő űrlapra navigálja a látogatót.



13. ábra A bejelentkezés

Az űrlap kitöltése után a „Belépés” gombra kattintva kísérelheti meg a vendég a bejelentkezést. Sikeres bejelentkezés esetén a rendszer a főoldalra navigálja a felhasználót, azonban, ha a bejelentkezés sikertelen, az alábbi felugró ablak figyelmeztet:

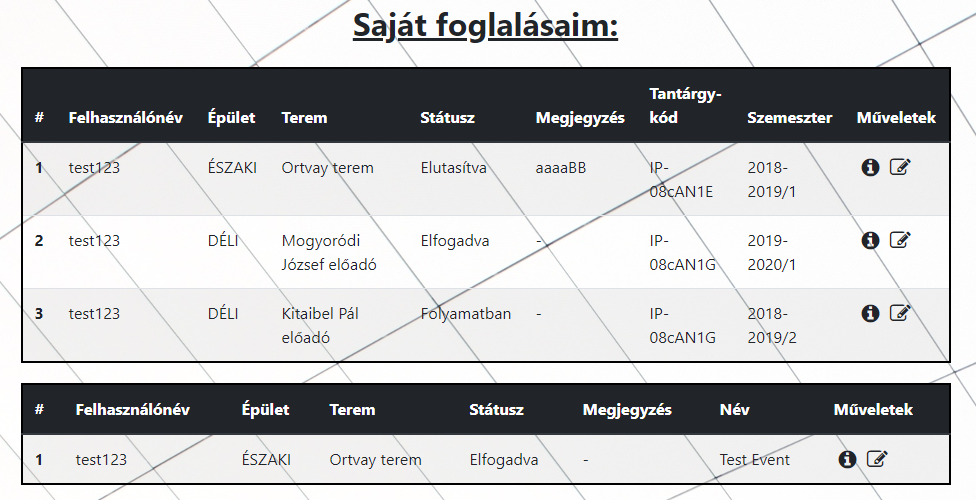


14. ábra Sikertelen bejelentkezés

### 2.3.2 Bejelentkezett felhasználók (tanárok) számára elérhető felület

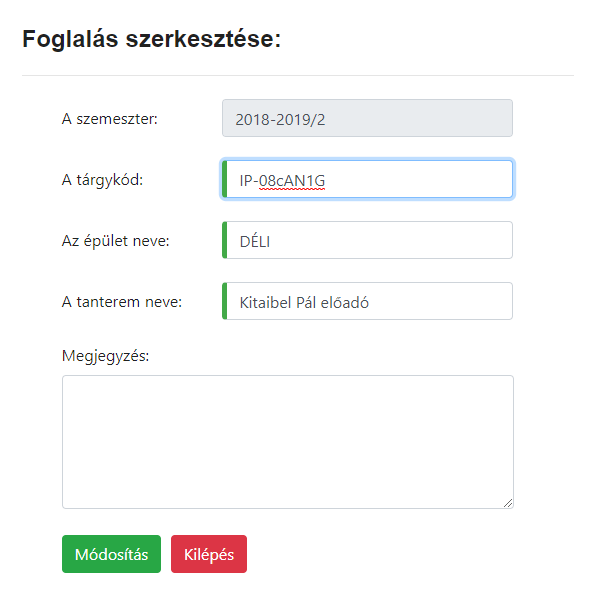
#### 2.3.2.1 Saját foglalásaim

A sikeres bejelentkezés után a főoldal tárul a felhasználó szeme elé, a menüsáv újabb, az alapvető elemeket kiegészítő funkciókkal bővül. A foglalásaim menüpont alatt kilistázhatjuk a saját foglalásainkat táblázatos formában státusztól függetlenül.



15. ábra A saját foglalásaim

A bejelentkezett felhasználónak lehetősége nyílik a saját foglalásai szerkesztésére. Ehhez a „Műveletek” oszlopban a „Szerkesztés” gombra kell kattintani. Ekkor az alábbi felugró ablak jelenik meg:



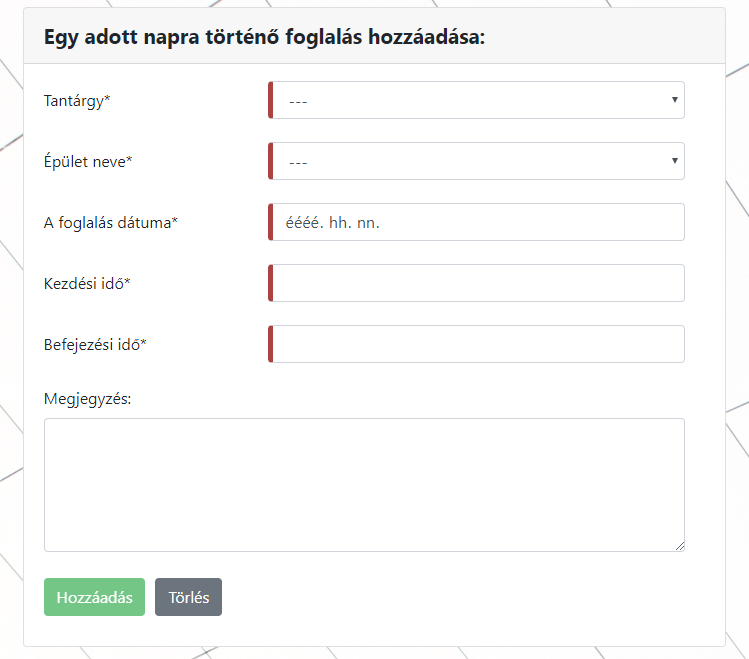
16. ábra Saját foglalás szerkesztése

#### 2.3.2.2 Űrlapok

Az űrlapok menüpont alatt három lehetőség közül választhat a felhasználó:

* Új egyszerű foglalás létrehozása
* Új szemeszterre vonatkozó foglalás létrehozása
* Új eseményre vonatkozó foglalás létrehozása

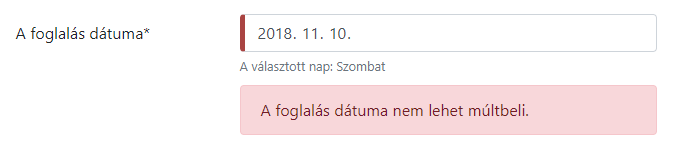
Mindhárom menüpont esetén egy-egy űrlap fogadja a felhasználót. Abban az esetben, ha a felhasználó az első menüpontot választja, azaz új egyszerű foglalást szeretne létrehozni, az alábbi űrlap jelenik meg:



17. ábra Egyszerű foglalás létrehozása

Az űrlap kitöltése során néhány adatot legördülő menük segítségével lehet kiválasztani, néhányat be kell vinni, és van, amit naptár segítségével kell kiválasztani. Ha ez sikeresen megtörtént, akkor a „Hozzáadás” gomb segítségével lehet rögzíteni a foglalást. A „Törlés” gombra kattintva az űrlap eredti állapotba állítható vissza. A szemeszterre és eseményre vonatkozó foglalások regisztrálása hasonlóképpen történik.

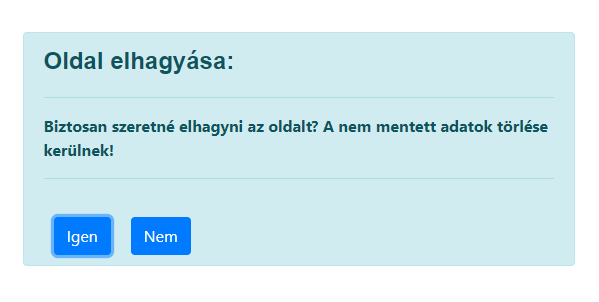
Minden egyes űrlap leadásához szükséges, hogy érvényes legyen. A mezők melletti szín jelöli az adott mező állapotát. Piros, ha a mező érvénytelen és zöld, ha érvényes. Abban az esetben, ha egy mező már érintett és mégis érvénytelen, akkor a mező alatt piros alapon hibaüzenet jelenik meg. Ennek a célja, hogy segítse a felhasználót a mező helyes kitöltésében.



18. ábra Egy érvénytelen mező

Az űrlap akkor számít érvényesnek, ha minden mező érvényes és egyetlen hibaüzenet sincs.

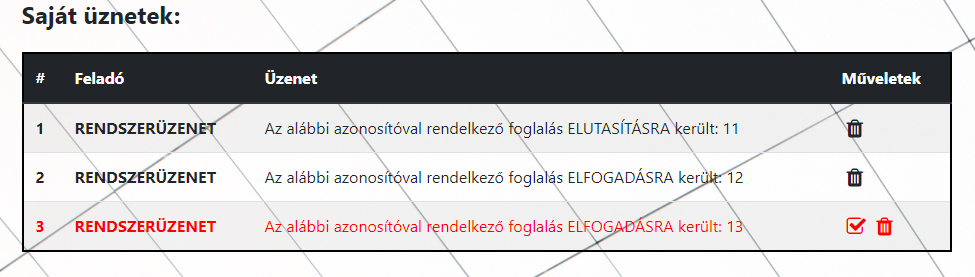
Abban az esetben, ha egy űrlap kitöltése már megkezdődött, de a felhasználó mégis megpróbál elnavigálni az oldalról, egy felugró ablak figyelmezteti arról, hogy a nem mentett adatok törlésre kerülnek.



19. ábra Oldal elhagyása figyelmeztetés

#### 2.3.2.3 Üzenetek

A felhasználónak lehetősége nyílik a saját üzenetei megtekintésére az „Üzenetek” menüpont alatt. A megjelenő táblázat minden sora egy-egy üzenetet jelöl. Az olvasatlan üzenetekhez tartozó sor piros, míg az olvasottakhoz tartozó fekete. Továbbá az oldal lehetőséget nyújt az üzenetek törlésére az „Üzenete törlése” gomb segítségével, valamint az olvasatlan üzenetek státuszának állítására a „Megjelölés olvasottként” gombra kattintva.



20. ábra Saját üzenetek

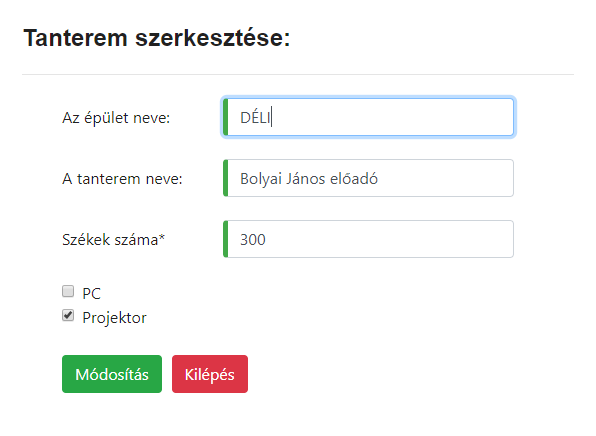
### 2.3.3 Adminisztrátorok számára elérhető felület

#### 2.3.3.1 Táblázatok

A „Táblázatok” menüpont alatt lehetősége nyílik az adminisztrátornak az adatbázisban található adatok táblázatos formában történő kilistázására. A legördülő menü az alábbi menüpontokat tartalmazza:

* Felhasználók
* Tantermek
* Félévek
* Tantárgyak
* Épületek
* Tantárgy foglalások
* Esemény foglalások

Minden egyes táblázat lehetővé teszi az adminisztrátor számára az adott sorhoz tartozó entitás szerkesztését. Ezt a „Műveletek” oszlopban a „Szerkesztés” gombra kattintva tehetjük meg. Ekkor egy, az adott entitáshoz tartozó, felugró ablak jelenik meg. A változtatásokat a „Módosítás” gombra kattintva lehet véglegesíteni.



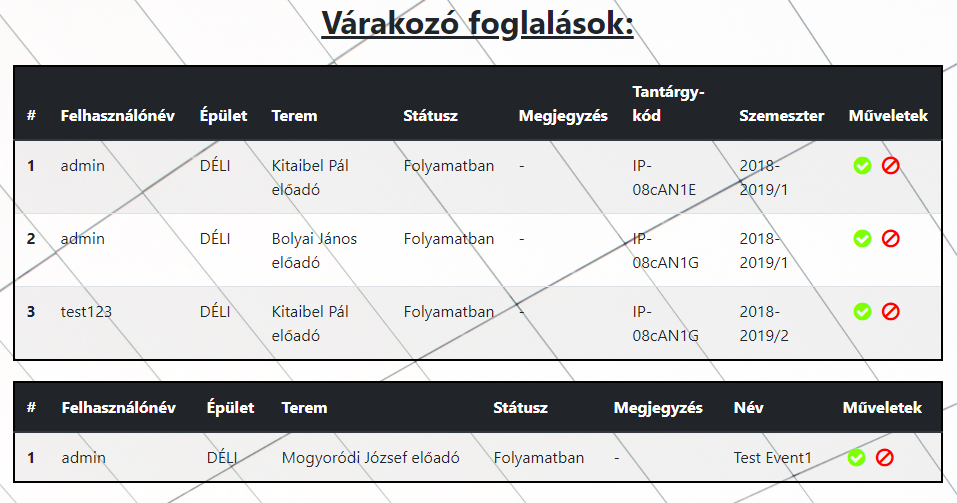
21. ábra A tanterem szerkesztéséhez tartozó felugró ablak

#### 2.3.3.2 Űrlapok

Az űrlapok menüpont bővült, újabb, kényelmesen használható funkciókat tartalmaz, lehetőséget nyújtva új szemeszter, felhasználó, tanterem, tantárgy vagy épület létrehozására, az érvényességre nagy hangsúlyt fektetve.

#### 2.3.3.3 Várakozó foglalások

A „Várakozó foglalások” menüpont célja, hogy az adminisztrátor könnyedén, grafikus felületen tudja kezelni az újonnan felvett foglalásokat. Az összes éppen „Folyamatban” státusszal rendelkező foglalás itt tekinthető meg. A foglalás státusza az „Elfogadás” gombra kattintva „Elfogadott” értékre módosul, az „Elutasítás” gomb hatására pedig „Elutasított” értéket vesz fel. Minden egyes módosítás azonnal frissül az adatbázisban, és mellékhatásként generál egy üzenetet a foglaló személy számára a státuszváltozásról.



22. ábra Várakozó foglalások

# 3. Üzemeltetői dokumentáció

## 3.1 Hardveres követelmények

A szoftvereknek megfelelő minimum és ajánlott hardveres követelmények az alábbi táblázatban tekinthetők meg:

|  |  |  |
| --- | --- | --- |
| Operációs rendszer | Minimum követelmény | Ajánlott követelmény |
| Windows x86 32-bit  Windows x86 64-bit | **Processzor:** 2 mag  **Memória:** 2 GB  **Merevlemez:** 10 GB szabad tárhely | **Processzor:** 4 mag vagy több  **Memória:** 8 GB vagy több  **Merevlemez:** 20 GB szabad tárhely |
| Linux x86 32-bit  Linux x86 64-bit | **Processzor:** 2 mag  **Memória:** 2 GB  **Merevlemez:** 10 GB szabad tárhely | **Processzor:** 4 mag vagy több  **Memória:** 8 GB vagy több  **Merevlemez:** 20 GB szabad tárhely |
| Mac OS X | **Processzor:** 2 mag  **Memória:** 2 GB  **Merevlemez:** 10 GB szabad tárhely | **Processzor:** 4 mag vagy több  **Memória:** 4 GB vagy több  **Merevlemez:** 20 GB szabad tárhely |
| Solaris x86 64-bit  Solaris Sparc 64-bit | **Processzor:** 2 mag  **Memória:** 2 GB  **Merevlemez:** 10 GB szabad tárhely | **Processzor:** 4 mag vagy több  **Memória:** 8 GB vagy több  **Merevlemez:** 20 GB szabad tárhely |

## 3.2 Szoftveres követelmények

Az alkalmazás tetszőleges környezetben üzemeltethető. A szükséges szoftverek listája a következő:

* Apache Maven 3.5.4+
* Apache Tomcat 8.0+
* Java 1.8+
* MySQL 8.0+
* NodeJs 8.12+ / Npm 6.4+

## 3.3 Telepítés

* Első lépésként indítsuk el a MySQL szervert.
* Az első indításkor szükség van az adatbázis/séma létrehozására (ajánlott a „roomreservationdb” név használata). Abban az esetben, ha más adatbázis név kerül megadásra, akkor a rrs-server\src\main\resources\application.properties fájlban az adatforrás URL-t ennek megfelelően kell beállítani.
* Az alkalmazás egyéb fontos beállításainak módosítására szintén az előbb említett application.properties fájlban van lehetőség.
* A gyökér könyvtárban található build.sh script futtatásával, vagy a mvn clean install parancs kiadásával tudjuk a backend és frontend fordítását elindítani.
* Ha ez megtörtént, és sikeres volt, akkor a rrs-server\deploy.sh futtatásával, vagy a mvn spring-boot:run parancs kiadásával tudjuk az alkalmazást elindítani az alapértelmezett 8080-as porton.
* Az alkalmazás első indítása létrehozza a megfelelő táblákat és a közöttük lévő kapcsolatokat.
* A mintaadatokkal történő feltöltéshez az testing\insert.sql fájl futtatása szükséges.

# 4. Fejlesztői dokumentáció

## 4.1 A feladat

A feladatom egy olyan teremfoglaló rendszer létrehozása volt, amelynek alkalmasnak kell lennie az esetleges jövőbeli továbbfejlesztésre, ezért a megfelelő működés mellett nagy figyelmet kellett fordítanom az adatszerkezetek helyes létrehozására is.

A különböző szerepkörök az alábbi funkciók használatára képesek:

* Vendég
  + Elfogadott foglalások táblázatos megjelenítése
  + Elfogadott foglalások kalendáriumban történő megjelenítése
* Felhasználó / Tanár
  + Saját foglalások megjelenítése táblázatban
  + Saját foglalások információinak megtekintése
  + Saját foglalások szerkesztése
  + Új egyszerű foglalás leadása
  + Új szemeszterre vonatkozó foglalás leadása
  + Új eseményre vonatkozó foglalás leadása
  + Saját üzenetek megtekintése / szerkesztése
* Adminisztrátor
  + Felhasználók táblázatos megjelenítése és szerkesztése
  + Tantermek táblázatos megjelenítése és szerkesztése
  + Félévek táblázatos megjelenítése és szerkesztése
  + Tantárgyak táblázatos megjelenítése és szerkesztése
  + Épületek táblázatos megjelenítése és szerkesztése
  + Tantárgy foglalások táblázatos megjelenítése és szerkesztése
  + Esemény foglalások táblázatos megjelenítése és szerkesztése
  + Új szemeszter felvétele
  + Új felhasználó (tanár) felvétele
  + Új tanterem felvétele
  + Új tantárgy felvétele
  + Új épület felvétele
  + Várakozó foglalások kezelése

## 4.2 A fejlesztés menete

### 4.2.1 A fejlesztői környezet

#### 4.2.1.1 Backend

A backend fejlesztése során a tanulmányaim közben már megismert NetBeans-re esett a választásom. A NetBeans egy integrált fejlesztői környezet, ami a Java nyelven alapul. A program grafikus fejlesztőfelületet kínál a különböző alkalmazások, Appletek vagy akár JavaBeanek elkészítéséhez, amelyek segítségével könnyebben, gyorsabban tudjuk fejleszteni saját programjainkat. Megszámlálhatatlan kényelmi funkcióval is rendelkezik, mint például a kódgenerálás, automatikus formázás és a kódkiegészítés.

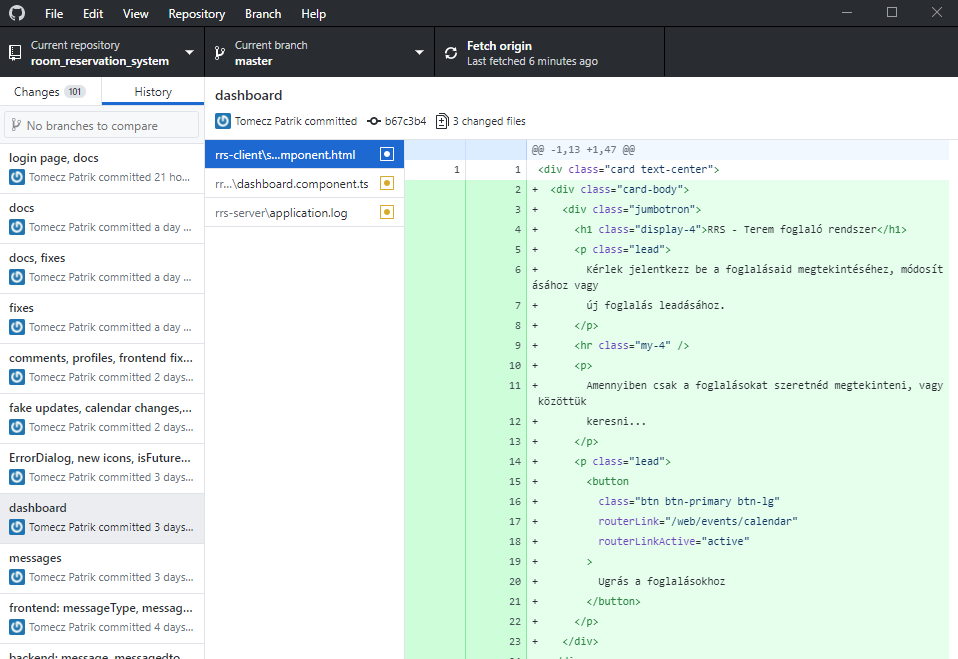
#### 4.2.1.2 Frontend

A megjelenítési réteg kódolása során a Visual Studio Code szerkesztő mellett döntöttem. A Visual Studio Code, melyet gyakran csak VS Code-nak rövidítenek, egy ingyenes, nyílt forráskódú kódszerkesztő, melyet a Microsoft fejleszt Windows, Linux és OS X operációs rendszerekhez. Támogatja a hibakeresőket, valamint beépített Git támogatással rendelkezik, továbbá képes az intelligens kódkezelésre az IntelliSense segítségével. Ezen felül testre szabható, így a felhasználók megváltoztathatják a kinézetet, a szerkesztő gyorsbillentyű-kiosztását, az alapértelmezett beállításokat és még sok egyebet. A program megnyitása után a kiegészítők menüpont alatt lehetőség nyílik saját kiegészítők telepítésére, ki- és bekapcsolására.

### 4.2.2 Verziókezelés

A fejlesztés kezdetétől a program forráskódját a [www.github.com](http://www.github.com) oldalon tároltam. A célom az volt, hogy bármilyen, a forráskódban történt módosítás visszakereshető, ellenőrizhető, visszaállítható legyen. A gitnek köszönhetően követni tudtam minden egyes változást, a projektet több számítógépről is el tudtam érni, valamint mindig volt egy biztonsági másolatom, ha esetlegesen valamely hardver meghibásodott volna.

Kezdetben a gitet parancssori felületen keresztül használtam, majd kis idő elteltével áttértem a GitHub Desktop alkalmazásra. Ezen program segítségével, grafikus felületen könnyedén végrehajtható volt minden, amire a verziókezelés során szükségem volt.



23. ábra A GitHub Desktop alkalmazás

## 4.3 A felhasznált technológiák

|  |  |
| --- | --- |
| Technológia | Leírás/Indoklás |
| Java Spring Boot | A Spring egy nyílt forráskódú, inversion of controlt megvalósító Java alkalmazás keretrendszer.  A kontroll megfordítása (angolul inversion of control, röviden IoC) főleg objektumorientált programozási nyelvekben használt technika a komponensek összeillesztésére, konfigurálására és kezelésére. A technika lényege, hogy a komponenskezelést (pl. létrehozást, példányosítást, paraméterezést, megszüntetést, metódus hívást) kiemeljük a programkódból, és általában egy külső keretrendszerre bízzuk, mint pl. a Spring. Célja a modularitás növelése és bővíthetővé tétele.  A Spring keretrendszer magját képező szolgáltatásokat főként Java alkalmazás fejlesztésére használják a programozók. Ugyanakkor a Java EE platformra is elérhetők a bővítményei, amelyek web-alkalmazás fejlesztését segítik elő. |
| Maven | Az Apache Maven (röviden Maven) egy szoftver, amelyet szoftverprojektek menedzselésére és a build folyamat automatizálására lehet használni.  A Maven bevezeti a POM, azaz a Projekt Objektummodell (angolul: Project Object Model) fogalmát. Egy POM egy buildelendő projektet ír le és annak függőségeit. Az egyes lépéseket céloknak, angolul goal-oknak nevezik. Vannak előre definiált célok a tipikus feladatokra, mint például a kód fordítása és csomagolása, de a felhasználónak lehetősége van saját célokat is definiálni a projektspecifikus lépések végrehajtására.  A Maven hálózatképes, tehát szükség esetén dinamikusan is le tud tölteni komponenseket. Repository névvel illetik a különböző hosztok fájlrendszereinek azon mappáit, ahol a letölthető komponensek találhatók. A Maven nem csak a repository-kból való letöltést támogatja, hanem a készült szoftvercsomag feltöltését is. |
| Hibernate | A Hibernate egy objektum-relációs leképezést (ORM) megvalósító programkönyvtár Java platformra. Segítségével osztályokat és a relációs adatbázisok tábláit tudjuk egymásba leképezni, az adatbázisban lévő rekordokat objektumokként kezelhetjük, és az objektumainkat egyszerűen tárolhatjuk állapotmegőrző módon adattáblákban.  A Hibernate használható önálló Java alkalmazásokban is, ugyanakkor tipikusan Java EE környezetben alkalmazzák. |
| JPA | A Java Persistence API, vagy röviden JPA, egy keretrendszer a Java programozási nyelvhez, melynek fő feladata a relációs adatok kezelése a Java-t használó alkalmazásokban. |
| MySQL | Az adatbázis motor szerepére a MySQL-t választottam, mivel ez az egyik legelterjedtebb adatbázis-kezelő napjainkban. Telepítése könnyű, Java-s környezetben könnyen használható a MySQL Conncetor segítségével. Az adatok vizsgálatára, módosítására és ellenőrzésére biztosít egy MySQL Workbench névvel ellátott programot. |
| Angular 6 | Az Angular egy TypeScript alapú, nyílt forráskódú frontend keretrendszer, melyet a Google Angular csapata fejleszt. Segítségével egyszerű módon bármilyen komplex webalkalmazás létrehozása lehetséges. |
| Bootstrap 4 | A Bootstrap egy ingyenes, nyílt forráskódú frontend keretrendszer, melyet weboldalak, webalkalmazások tervezésére használnak. Megannyi előre megírt HTML és CSS alapú komponenst tartalmaz, többek között űrlapokat, gombokat, navigációs elemeket, táblázatokat és csúszkákat. |

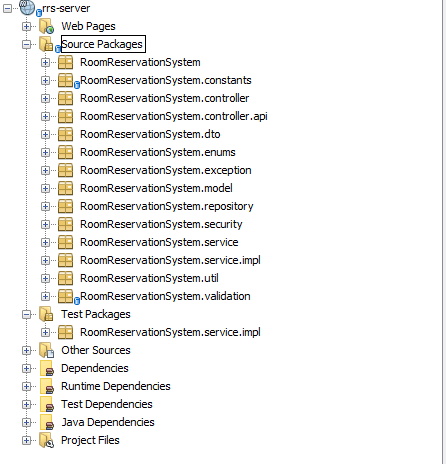
## 4.4 Az alkalmazás architektúrája

### 4.4.1 A backend

#### 4.4.1.1 A szerver felépítése

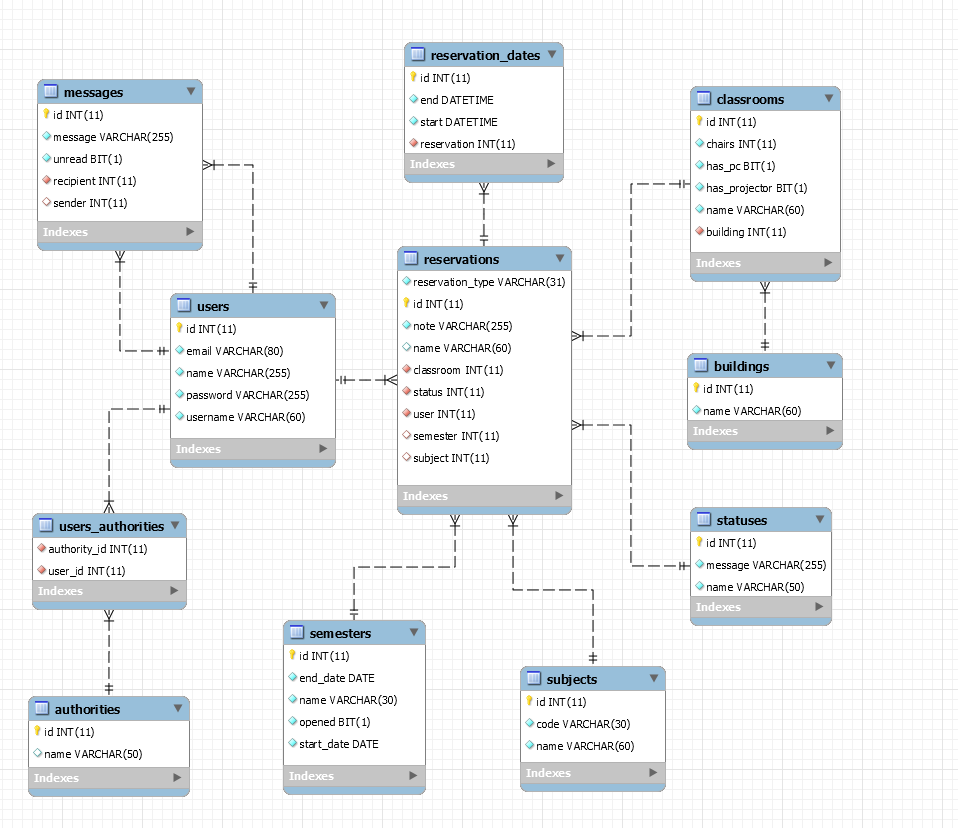
A fontosabb csomagok és leírásuk:

|  |  |
| --- | --- |
| Név | Leírás |
| constants | A konstansokat tartalmazó csomag. |
| controller  controller.api | Az entitásokhoz tartozó vezérlő osztályokat tartalmazó csomag. A vezérlő dolgozza fel a felhasználói adatbevitelt, függvényhívásokká képezi le azokat. Ezek fogják előidézni az adatok módosítását, törlését, vagy a nézetek megváltozását. |
| dto | Az adatátviteli (DTO, azaz Data Transfer Object) osztályokat tartalmazó csomag. Ezen objektumok célja a folyamatok közötti adatközvetítés. |
| enums | A felsorolási típusokat tartalmazó csomag. Ezek olyan típusok, melyeknek megengedett értékei fix konstansokból állnak. |
| exception | A kivételeket tartalmazó csomag. |
| model | Az entitásokat tartalmazó csomag. Az entitások azon objektumok, amelyeknek tulajdonságait gyűjtjük és tároljuk az adatbázisban. |
| repository | Az entitásokhoz tartozik egy-egy repository interfész. Ezen interfészek a JpaRepository interfészt terjesztik ki, és az adott entitásra vonatkozó adatbázis műveletekért felelősek. |
| security | A végpontok védettségét és a JWT (JSON Web Token) azonosítást megvalósító, konfiguráló osztályokat tartalmazó csomag. |
| service | A szervíz réteghez tartozó interfészeket tartalmazó csomag. |
| service.impl | A szervíz réteghez tartozó interfészeket implementáló osztályokat tartalmazó csomag. Feladata a repository és kontroller osztályok közötti kapcsolat megteremtése, az adatok megfelelő átalakítása. |
| util | Az szükséges segédosztályokat tartalmazó csomag. |
| validation | A beérkező objektumok ellenőrzéséért felelős osztályokat tartalmazó csomag. |



24. ábra A backend szerkezete

#### 4.4.1.2 Az adatbázis szerkezete



25. ábra Az adatbázis szerkezeti diagramja

#### 4.1.1.3 Az entitások

Az entitások gyakorlatilag az adatbázistábláknak felelnek meg, hiszen a JPA az annotációk segítségével az osztályokat SQL utasításokká alakítja át, melyek elkészítik az adatbázis sémákat. Az entitások létrehozása során, a fejlesztés megkönnyítése végett, segítségül hívtam a Lombok java könyvtárat, mellyel automatikusan generáltam le az egyes getter, setter, equals és hashcode metódusokat, valamint néhány osztály konstruktorát. Minden egyes entitás rendelkezik a @Table annotációval, mellyel a generált adatbázistábla nevét lehet megadni. Továbbá, mivel az adattagok attribútumoknak felelnek meg, ezért a @Column annotáció segítségével történt meg az egyes oszlopok neveinek megadása. Sok esetben szükség volt az entitások közötti kapcsolatok megadására, melyek az alábbiak lehetnek:

* @OneToOne: Egy az egyhez kapcsolat. Nagyon ritka, a szakdolgozatomban ezt a kapcsolat típust nem alkalmaztam.
* @OneToMany: Egy a sokhoz kapcsolat. Például egy épülethez több tanterem tartozik.
* @ManyToOne: Sok az egyhez kapcsolat. Az előző példa másik oldala, egy adott tanteremhez csak egy épület tartozhat.
* @ManyToMany: Sok a sokhoz kapcsolat. Például egy felhasználó több engedéllyel is rendelkezhet, de egy engedély akár több felhasználóhoz is tartozhat.

Az egyes entitások és rövid jellemzésük:

|  |  |
| --- | --- |
| Név | Jellemzés |
| BaseEntity | Az entitásokhoz tartozó absztrakt ősosztály. Szinte minden entitás belőle származik le. Egyetlen adattaggal rendelkezik, az azonosítóval (**id**), ennek köszönhetően minden entitás egyértelműen azonosítható lesz. |
| Authority | Egy egyszerű entitás, amely két adattaggal rendelkezik. Az első az engedély neve (**name**). Mivel azt szeretnénk, hogy egy adott engedély csak egyszer szerepeljen az adatbázisban, ezért ez egyedi. A második adattag pedig egy lista (**userList**), mely azokat a felhasználókat tartalmazza, akik rendelkeznek az adott engedéllyel. |
| User | A felhasználó entitás célja a felhasználók megfelelő tárolása. Az entitás rendelkezik felhasználónév (**username**) és e-mail cím (**email**) attribútumokkal. Fontos, hogy ezek külön-külön is egyediek legyenek, ezért itt nem engedünk meg duplikátumokat. Az osztály tartalmazza még a felhasználó jelszavát (**password**), teljes nevét (**name**),továbbá a hozzá tartozó engedélyek (**authorityList**), valamint foglalások (**reservationList**) listáját. |
| Message | Az üzenet entitás az egyszerű üzenetek tárolásához szükséges attribútumokat modellezi. Tárolja az üzenet feladóját (**sender**), a címzettet (**recipient**), az üzenet státuszát (**unread**) továbbá az üzenet szövegét (**message**). Mivel a cél az volt, hogy a rendszer is képes legyen üzenetküldésre, és ne csak felhasználók között történhessen meg, ezért a küldő lehet üres, ekkor a rendszer automatikusan azt feltételezi, hogy rendszerüzenetről van szó. |
| Building | Az épület entitás. Két attribútummal rendelkezik, az első az épület neve (**name**), a második pedig a tantermek listája (**classroomList**). Fontos, hogy nem szeretnénk két azonos névvel rendelkező épületet, ezért ez az attribútum egyedi. |
| Classroom | A tantermek tárolásához szükséges tulajdonságokat modellező entitás. Tárolja a tanterem nevét (**name**), az épületet, amiben a tanterem található (**building**), valamint a foglalások listáját (**reservationList**). Egyéb speciálisabb attribútumokkal is rendelkezik, mint például az adott terem befogadó képessége (**chairs**), vagy, hogy rendelkezésre áll-e számítógép (**hasPc**), projektor (**hasProjector**). Egy tantermet a neve nem azonosít egyértelműen, hiszen két különböző épület tartalmazhat azonos nevű termeket, viszont, ha a tanterem nevét az őt tartalmazó épülethez rendeljük, akkor ennek már egyértelműnek kell lennie. |
| Subject | Minden egyes tantárgy rendelkezik tárgykóddal (**code**), névvel (**name**), valamint a rá vonatkozó foglalások listájával (**reservationList**). Több tantárgy is rendelkezhet azonos névvel, de a tárgykódnak egyéninek kell lennie. |
| Semester | A szemeszter entitás. Tartalmazza a szemeszter nevét (**name**), amely egy egyedi érték, a szemeszter kezdetének (**startDate)** és végének (**endDate**) dátumát, a szemeszter státuszát (**opened**), valamint a szemeszterre vonatkozó foglalások listáját (**reservationList**). |
| Status | A státusz egy egyszerű entitás mindössze három adattaggal. Minden státuszhoz tartozik egy név (**name**), egy üzenet (**message**) és egy lista azokról a foglalásokról, amelyek a státusszal rendelkeznek (**reservationList**). A státusz neve egyedi. |
| Reservation | Ez a foglalásokhoz tartozó absztrakt ősosztály. Alapvetően azon attribútumokat tartalmazza, amelyek mindkét típusú foglalásban előfordulnak: a foglaló felhasználó (**user**), a tanterem (**classroom**), a foglalás státusza (**status**), a lefoglalt időpontok (**dateList**), valamint a foglaláshoz tartozó megjegyzés (**note**).  A leszármazás esetén az úgynevezett („Single Table”) stratégiát alkalmaztam, melynek lényege, hogy mind a tanórákra, mind az eseményekre vonatkozó foglalások tárolása egy közös táblában történik. Ehhez szükség volt egy úgynevezett megkülönböztető oszlop megadására, amelybe a foglalás típusától függően egyedi érték kerül. Azon attribútum, amellyel az adott foglalás nem rendelkezik (például az esemény foglaláshoz nem tartozik tantárgy) null értéket vesz fel. |
| ClassReservation | A tanórákra vonatkozó foglalások tárolásához szükséges attribútumokat modellező entitás. Az ősosztályból örökölt tulajdonságokon túl, rendelkezik tantárggyal (**subject**) és szemeszterrel (**semester**). |
| EventReservation | Az eseményekre vonatkozó foglalásokhoz tartozó entitás. Az alapvető tulajdonságokon kívül egyedül névvel (**name**) rendelkezik. Ez az érték egyedi. |
| ReservationDate | A foglalásokhoz tartozó időpontokat tároló entitás. Három mezővel rendelkezik. A foglalás (**reservation**) megadja, hogy melyik foglalásra vonatkozik. A kezdés (**start**) és befejezés (**end**) értelemszerűen a foglalás kezdésének és befejezésének pontos idejét jelölik. |

#### 4.1.1.4 A szervíz és repository interfészek/osztályok

Minden egyes entitáshoz tartozik egy repository interfész, amely a JpaRepository interfészt valósítja meg. Ezen interfészek kapcsolódnak az adatbázishoz és bonyolítják le a megfelelő műveleteket. Bár a JPA támogatja az SQL utasítás szöveg formátumban történő megadását, sokkal kényelmesebb a konvencionális név használata, amelyből az SQL parancs generálódik. Bár a támogatott kulcsszavak listája nagyon hosszú, mégis szeretnék közülük néhány, a szakdolgozatomban is előfordulót kiemelni.

* And: Segítségével a logikai feltételek összekapcsolhatók. Például findByNameAndChairsBetween esetén a név és székek száma alapján történik a keresés.
* Between: A „Between” kulcsszó segítségével adott intervallumon belül tudunk keresni. Például findByStartDateBetween esetén a paraméterben megadott két dátum közötti „startDate” attribútummal rendelkező entitásokat kapjuk eredményül.
* LessThan: A „LessThan” kulcsszó használatával a paraméterben megadott értéknél kisebb értékkel rendelkező entitásokat kapjuk válaszul. Például findByChairsLessThan.
* GreaterThan: A „GreaterThan” kulcsszó használatával az előbb említett működés ellentétjét érjük el. Például findByChairsGreaterThan visszaadja a paraméterben átadott értéknél több székkel rendelkező tantermeket egy listában.

Ahogyan minden entitáshoz tartozik egy repository interfész, úgy egy szervíz interfész és annak a megvalósítása is. A szervíz osztályok feladata, hogy átalakítsa az objektumokat az adatbázis / kontrollerek számára, ezért egyfajta kapcsolatot teremt a repositoryk és kontrollerek között. Amikor befut egy kérés a kontrollerhez, az meghívja a megfelelő szerviz osztály megfelelő metódusát, amely az adatbázisból lekérdezett adatokat módosítja, átalakítja a megjelenítési formátumra, majd visszaadja a kontrollernek, ami eljuttatja a frontend számára.

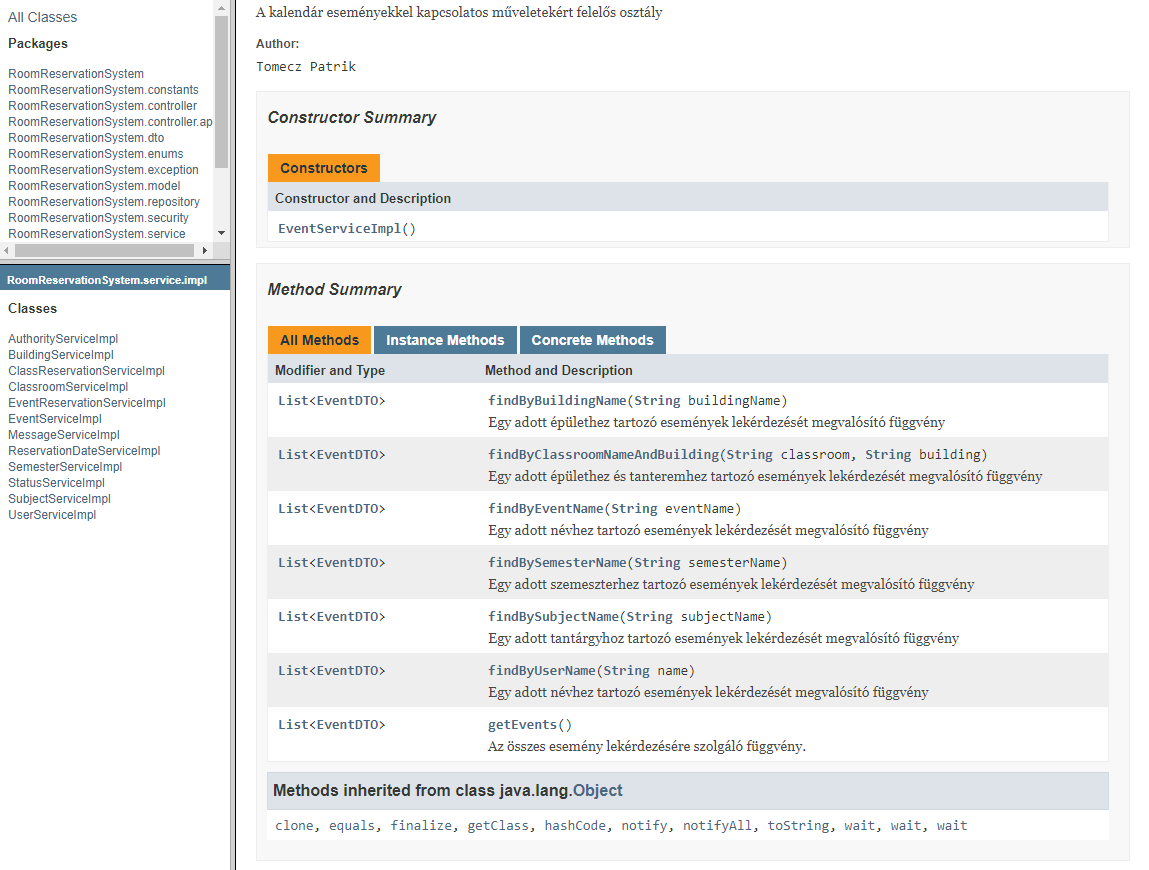
#### 4.1.1.5 A kontroller osztályok

A kontroller osztályok feladata a frontendtől érkező kérések kezelése. A végpontok eléréséhez gyakran hitelesítés szükséges, ennek hiányában, vagy ha a felhasználó nem rendelkezik a megfelelő engedélyekkel, a végpont nem érhető el. Minden egyes kérés, esetén a sikeres hitelesítés után, az aktuális kontroller meghívja az entitáshoz tartozó szervíz osztály megfelelő metódusát. Abban az esetben, ha nem történt semmilyen hiba, akkor a visszatérési értéket egy válaszentitásba csomagolja, és OK státusszal elküldi a frontend számára. Hiba esetén a hibaüzenetet 404-es státuszkóddal juttatja el a megjelenítési réteghez.

#### 4.1.1.6 Javadoc

A Javadoc (vagy hivatalosan JavaDoc) egy dokumentáció generátor melyet a Sun Microsystems készített a Java programozási nyelv számára, ezzel lehetővé téve a HTML formátumú API dokumentáció generálását a forráskódból.

A fejlesztés során törekedtem arra, hogy a frissen megírt kódot jól, konvencionálisan kikommentáljam. Ebben nagy segítség volt a NetBeans IDE, hiszen pár gomb lenyomásával automatikusan legenerálja az adott osztályhoz, függvényhez tartozó komment vázlatot, ezt csak megfelelően ki kell tölteni. A build folyamat során a megfelelő formátumú kommentekből létrejön egy különálló, jól olvasható, HTML formátumú dokumentáció, ami a rendszer megértését, felhasználhatóságát javítja. Ez a dokumentáció az rrs-server\target\apidocs mappában található.



26. ábra A forráskód alapján generált dokumentáció

### 4.4.2 A frontend

#### 4.4.2.1 A frontend felépítése

A megjelenítési réteg felépítését a már említett angular keretrendszer segítségével vittem véghez.

##### 4.4.2.1.1 Modules

Az alkalmazást a fejlesztés során modulokra bontottam, így az összetartozó komponensek, oldalak és szerviz osztályok egy helyre kerültek. Ezek gyökérkönyvtáron belül a „modules” mappában találhatók.

A modulok felépítése:

* components: A komponenseket tartalmazó mappa. Ide kerül az összes, az adott modul számára létrehozott komponens.
* pages: Az adott modulhoz tartozó oldalak könyvtára. Ezek a már kész, megjelenítésre szánt komponensekből felépített lapok.
* <modulnév>.module.ts: Maga a modul, tartalmazza az importokat, exportokat, deklarációkat és szolgáltatókat.
* <modulnév>-routing.module.ts: A modulhoz tartozó útvonalakért felelős osztály.

##### BuildingsModule:

Az épületekhez tartozó modul. A benne található komponensek felelnek az épületek táblázatban történő megjelenítéséért („building-table”), új épület hozzáadásáért („add-building-form”) és a már meglévő épületek adatainak módosításáért („edit-building-form”). Két oldalt tartalmaz ez a modul, az első megjeleníti az épületeket, a második segítségével új épületet lehet felvenni.

##### ClassroomsModule:

A tantermekhez kapcsolódó modul. Három komponenst tartalmaz, amelyek rendre a tanterem létrehozásáért („add-classroom-form”), a tantermek táblázatban történő megjelenítéséért („classroom-table”) és szerkesztéséért felelősek („edit-classroom-form”). A két oldal ezekre a komponensekre épít, így az alkalmazás lehetőséget biztosít az adminisztrátor számára a tantermek megtekintésére és hozzáadására.

##### EventsModule:

Ez a modul felel az összes eseményért és azok táblázatos, valamint naptári nézetben történő megjelenítéséért. A „calendar-header” komponens felel a naptár nézetben megjelenő fejlécért, amely segítségével a felhasználó könnyebben tájékozódhat az aktuális nézetről, válthat a nézetek között, valamint lapozhatja a naptárat. Ezt a komponenst használja fel a „calendar-full”, amely már a kész kalendáriumot tartalmazza. Ahhoz, hogy az oldal látogatói számára kényelmesebb legyen a keresés, létrehoztam egy „filter-events-form” névvel ellátott komponenst, amely szintén a naptárhoz tartozik, és segítségével a felhasználó a megadott tulajdonságok alapján szűrheti az eseményeket. A modul tartalmaz egy „calendar-table” komponenst is, amely az események táblázatos megjelenítéséért felelős. Az oldalak értelemszerűen az események táblázatos és naptári reprezentációjáért felelősek.

##### MessagesModule:

Ez talán az egyik legegyszerűbb modul. Egy komponenst és a hozzátartozó oldalt tartalmazza. A komponens („message-table”) segítségével a bejelentkezett felhasználóhoz tartozó üzeneteket tudjuk megjeleníteni táblázatos formában. A komponens támogatja az üzenetek „olvasott” státuszra állítását, valamint a törlést.

##### ReservationsModule:

A foglalásokhoz tartozó modul, mely talán a legösszetettebb az összes közül. Több komponenst is tartalmaz, amelyek az egyes foglalások létrehozásáért, különböző megjelenítéséért és szerkesztéséért felelősek. Az „add-event-reservation-form” az eseményre vonatkozó foglalás, az „add-semester-reservation-form” egy adott szemeszterre történő tanórára vonatkozó foglalás, míg az „add-simple-reservation-form” egy adott időpontra történő tanórára vonatkozó foglalás létrehozásáért felelős űrlapot tartalmazza. A foglalások táblázatos reprezentációjáért rendre a „class-reservation-table” és az „event-reservation-table” komponensek illetékesek. Mivel az alkalmazás támogatja az egyes foglalásokra vonatkozó részletes információk megtekintését, ezért szükséges ezen komponensek („observe-class-reservation-form” és „observe-event-reservation-form”) létrehozására is. Fontos célkitűzés volt, hogy a tanároknak és az adminisztrátoroknak lehetősége legyen a foglalások szerkesztésére. Ezen funkciók megvalósítása az „edit-class-reservation-form” és „edit-event-reservation-form” feladata. Több oldal is van, amely az előbb említett komponensekkel dolgozik, de nem mind került felhasználásra, hiszen néhány funkciót a felugró ablakok segítségével lehet elérni.

##### SemestersModule:

A szemeszterekhez tartozó modul. A komponensek felelnek a táblázatban történő megjelenítéséért („semester-table”), az új szemeszter felvételéért („add-semester-form”) és a már meglévő szemeszter adatainak megváltoztatásáért („edit-semester-form”). Két oldalt tartalmaz, amelyek a már említett komponensekre épülnek. Az első oldal megjeleníti az szemesztereket, a második segítségével új szemesztert lehet felvenni.

##### SubjectsModule:

A tantárgyakért felelős modul. Az alapvető komponenseket tartalmazza, akárcsak az előző modul. A komponensek segítségével lehetőség nyílik a tantárgyak megtekintésére („subjects-table”), felvételére („add-subject-form”) és szerkesztésére („edit-subject-form”). A lapoknak hála ezek a komponensek megfelelően megjelennek a felhasználó előtt.

##### UsersModule:

A felhasználókhoz tartozó modul. A komponenseknek hála lehetőség nyílik új felhasználó felvételére („add-user-form”), a felhasználók szerkesztésére („edit-user-form”), a felhasználók megjelenítésére („user-table”) és a bejelentkezésre is („login-form”). Az egyes oldalak feladata az előbb említett funkciók reprezentációja a látogató felé.

##### 4.4.2.1.2 Shared

Ebben a könyvtárban a megosztott tartalmak találhatók. A tartalom típustól függően alkönyvtárakba lett csoportosítva.

##### Components:

Ebbe a könyvtárba azon komponensek kerültek, amelyek több modul által kerültek, vagy kerülhetnek a jövőben felhasználásra. Ilyen komponensek például a dialógusok, vagy a főoldal.

##### Config:

A konfigurációs állományokat tartalmazó alkönyvtár.

##### Directives:

A direktívákat tartalmazó mappa. Ezen osztályok segítségével valósult meg az űrlapok hitelesítése. A direktíváknak számos fajtája van. Található közöttük olyan, amelynek „Cross Field Validation” a feladata, azaz egy adott űrlapon belül kettő vagy több mező alapján történő validáció. Ilyen például a „confirm-email.directive” vagy a „confirm-password.directive”, melyek rendre ez e-mail cím és jelszó egyezését vizsgálják. De vannak úgynevezett aszinkron validációért felelős direktívák is, ezek a backend segítségével ellenőrzik az adatok helyességét. Többek között ilyen a „taken-event-name”, amely azt vizsgálja, hogy van-e az adatbázisban a megadott névvel rendelkező foglalás, ha igen, akkor jelzi a hibát.

##### Enums:

A felsorolási típusokat tartalmazó könyvtár.

##### Guards:

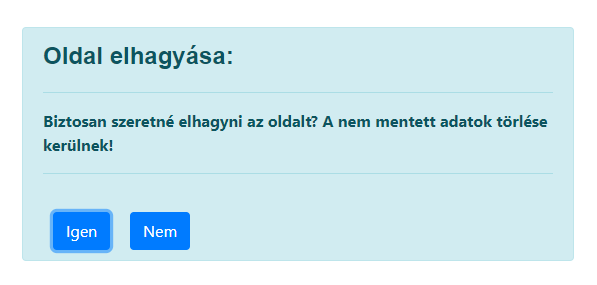
A guard-ok olyan interfészek, melyek segítenek a router számára eldönteni, hogy megtörténjen-e a navigáció a kért útvonalra. A döntéshozatal egy igaz/hamis érték alapján történik. Összesen öt típusa van a guard-oknak, melyek az alábbiak:

* CanActivate
* CanDeactivate
* CanActivateChild
* CanLoad
* Resolve

Mivel a szakdolgozatomban csak az első két guard jelenik meg, ezért ezeket szeretném részletezni.

CanDeactivateGuard:

Ez a guard a CanDeactivate<T> interfészt implementálja és az űrlapokat tartalmazó lapokhoz történt a hozzárendelése. Feladata, hogy a felhasználótól megerősítést kérjen abban az esetben, ha úgy szeretne elnavigálni az oldalról, hogy az űrlap kitöltését már megkezdte. Ekkor a rendszer egy felugró ablakban figyelmezteti a felhasználót.



27. ábra Az oldal elhagyását megerősítő dialógus

RoleGuard:

A RoleGuard a CanActivate interfész implementálásával jött létre. Feladata, hogy az egyes útvonalak elérését csak abban az esetben engedélyezze, ha az adott illető rendelkezik a megfelelő jogosultsággal. A guard minden egyes útvonal esetén tisztában van az útvonal eléréséhez szükséges engedéllyel, illetve képes az adott felhasználó engedélyeinek lekérdezésére, így egy könnyed ellenőrzés után mindig a megfelelő oldalra történik a navigáció.

##### Models:

Az entitásokat tartalmazó könyvtár.

##### Modules:

Az egyéb modulokat tartalmazó könyvtár. Ide tartozik a material és a pipe modul.

##### Pipes:

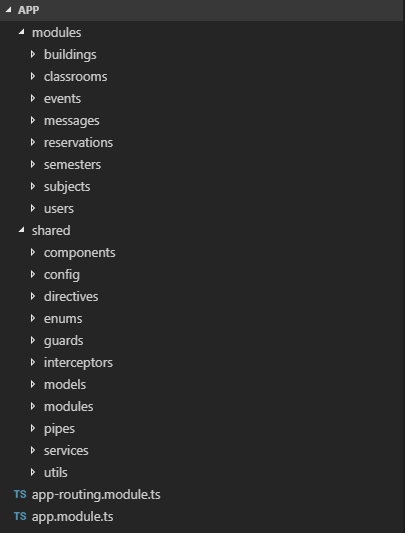
A csöveket tartalmazó mappa. A csövek a pipe modulhoz tartoznak, és segítségükkel történik az adatok formázása a HTML kódban.

##### Services:

A szerviz osztályok helye. Ide tartoznak az api szerviz osztályok, de itt található a dialógusokhoz használt osztály is.

##### Utils:

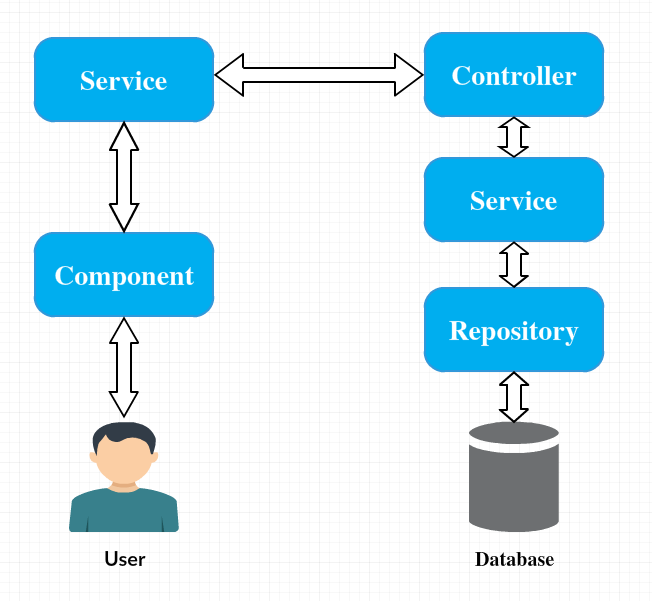
A segédosztályokat tartalmazó alkönyvtár. Többek között ezen osztályok segítségével történik meg például a backendtől érkező üzenetek formázása vagy dátumok generálása egy adott intervallumon belül.



28. ábra A frontend szerkezete

## 4.5 Az alkalmazás működése

1. A felhasználó interakcióba lép a komponenssel.
2. A komponens megfelelően értelmezi a felhasználói input-ot, majd meghívja a szerviz osztály megfelelő metódusát.
3. A szerviz osztály metódusa http kérést kezdeményez. A kérés típusa lehet POST, GET, DELETE és PUT, attól függően, hogy az adatokat bevinni, lekérdezni, törölni vagy frissíteni szeretnénk.
4. A http kérés beérkezik a backend megfelelő kontroller osztályához, majd meghívja a szerviz osztály megfelelő metódusát.
5. A szerviz osztály a repository segítségével felveszi a kapcsolatot az adatbázissal és végrehajtja a megfelelő műveletet.



29. ábra Az alkalmazás működése

A fentebb leírt működés szemléltetése egy konkrét példán keresztül:

1. A felhasználó megfelelően kitölti a foglalás leadásáért felelős űrlapot, majd rálép a hozzáadás gombra.
2. Az űrlap meghívja a foglaláshoz tartozó szerviz osztály azon metódusát, ami a foglalás létrehozásáért felel.
3. Ez a metódus kezdeményezi a megfelelő http (post) kérést.
4. A http kérés befut a foglalásokhoz tartozó kontroller osztályhoz. Amennyiben nem lép fel semmilyen hiba, az objektumot átadja a szerviz osztály megfelelő metódusának.
5. A metódus létrehozza a foglalás entitást (konverziók segítségével), majd a repository segítségével elmenti az adatbázisba.

## 4.6 Az azonosítás

A felhasználó a bejelentkezés gombra kattintva elküldi a felhasználónév/jelszó párosát a backendnek. A backend értelmezi a kérést, majd ellenőrzi az adatok helyességét. Amennyiben helyes adatokkal kísérelte meg a vendég a bejelentkezést, úgy egy tokent kap válaszul, amelyben több kulcsfontosságú információ szerepel, mint például az engedélyek listája és a token lejáratának dátuma. Ezen információk eltárolódnak a Local Storage-ben, hogy bármikor hozzájuk lehessen férni, hiszen minden egyes kérés esetén a token csatolásra kerül. A felületen az engedélyek ismeretében új funkciók jelennek meg a felhasználó számára.

## 4.5 Tesztelés

### 4.5.1 Backend

A backend tesztelése során külön kitértem a szerviz és kontroller osztályok vizsgálatára.

#### 4.5.1.1 A szerviz osztályok

A szerviz osztályok tesztelését JUnit és Mockito keretrendszerek segítségével valósítottam meg.

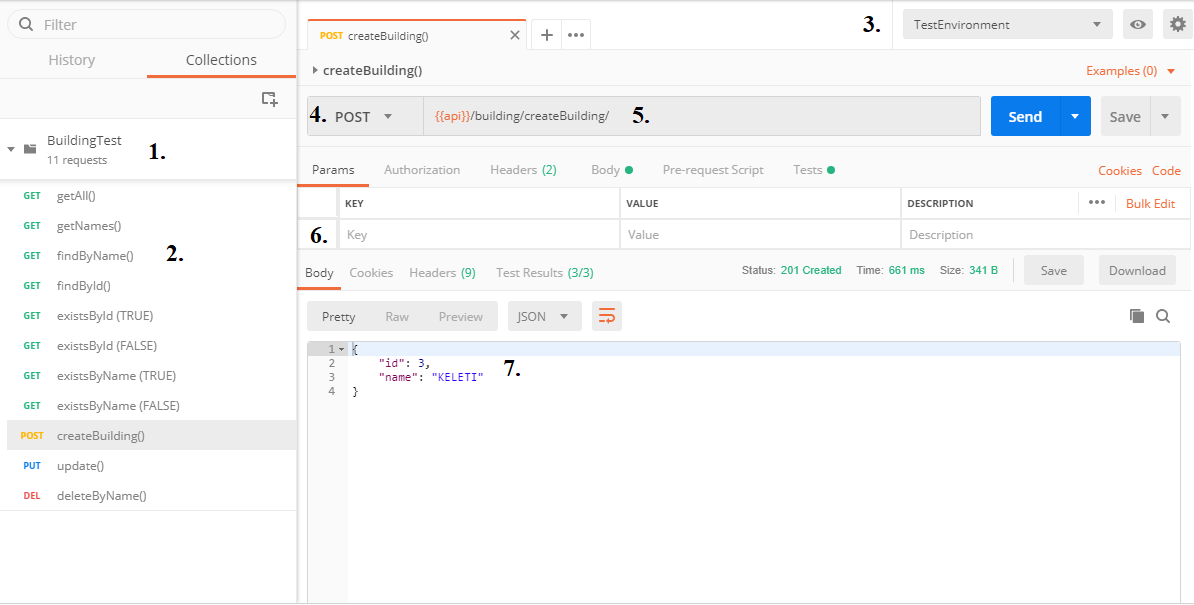
JUnit egy egységteszt keretrendszer Java programozási nyelvhez. A JUnit teszteket gyakran a build folyamat részeként szokták beépíteni. Pl. napi build-ek esetén ezek a tesztek is lefutnak. A release akkor hibátlan, ha az összes teszt hibátlanul lefut.

A tesztosztályok felépítése általában azonos. Az osztályok elején találhatóak a Mockito számára fontos adattagok a megfelelő annotációkkal ellátva, ezeket követik a tesztekhez szükséges (osztályszintű) segédváltozók, majd végül az egyes függvények tesztjei. Egy függvényhez legalább egy, de akár több teszt is tartozhat a tartalmától, kivételektől függően. A tesztfüggvények az esetek nagy részében először a repositorykat mókolják a Mockito segítségével, majd megtörténik valamely, a szerviz osztályba tartozó függvény meghívása, legvégül pedig ennek a függvényhívásnak az eredményét teszteljük a JUnit segítségével.

#### 4.5.1.2 A kontroller osztályok

A végpontok tesztelését a Postman nevezetű alkalmazás segítségével valósítottam meg. Azért esett erre a programra a választásom, mert kezelőbarát felhasználói felülettel rendelkezik, automatizálható a tesztelés és széleskörű lehetőségeket biztosít.

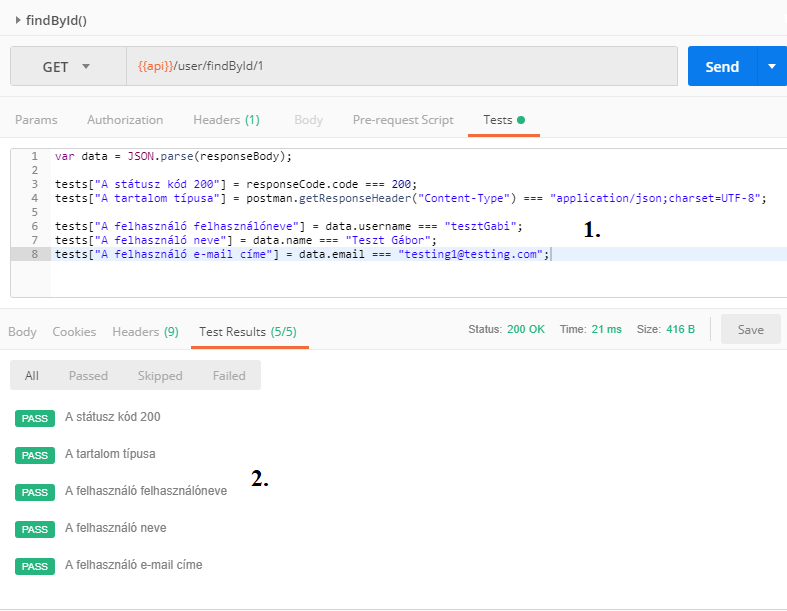
Az alkalmazás használata:



30. ábra A Postman használata

1. A megfelelő gyűjtemény kiválasztása. Ezen gyűjtemények szolgálnak a kérések csoportosítására, könyvtárba szervezésére. Könnyen kezelhetőek, továbbá a Postman lehetőséget biztosít az importálásra, exportálásra és duplikálásra is.
2. A kollekció kiválasztása után megjelennek a hozzá tartozó kérések.
3. Annak érdekében, hogy a kéréseket és tesztelést könnyebben, dinamikusabban tudjuk végrehajtani a Postman-ben használhatunk olyan változókat, amelyeket a környezetekben definiálhatunk. Ezen változókkal megadható az api végpont, de akár a hitelesítéshez használt token is, így ezen adatok módosulása esetén, csak egy helyen kell az adatokat frissíteni. A környezet minden kérés futtatása előtt kiválasztható, importálható, exportálható.
4. Az alkalmazás lehetőséget nyújt a kérés típusának kiválasztására. A felhasználó a legördülő menüre kattintva jelölheti ki a számára megfelelőt.
5. A kéréshez tartozó URL megadására az ötös számmal jelölt mező szolgál. A mintaképen látható, hogy az URL részét képezi az „{{api}}” szöveg. Ez egy a 3-as pontban említett változók közül.
6. A kérés paramétereinek megadására szolgáló felület (opcionális). A paraméterek kulcs-érték párok, ezek segítségével generálódik a megfelelő URL.
7. A kéréshez tartozó válasz a 7-es számmal jelölt mezőben jelenik meg. A mező tartalmának formátuma beállítható.

Tesztelés a Postman-ben:



31. ábra Tesztelés a Postman-ben

A Postman segítségével Javascript nyelven írt teszteket tudunk készíteni. A tesztek során nem csak a válasz objektum attribútumainak vizsgálatát tudjuk végrehajtani, hanem akár a válasz státuszát, tartalmának típusát is ellenőrizhetjük.

A mintaképen az 1-es számmal jelzett szövegdobozban megtekinthető egy minta teszteset. A tesztelés során először elmentjük egy változóba a válasz JSON objektumot, majd megtörténik a tesztek definiálása. Mint látható ezen tesztek a válaszhoz tartozó státuszkódót, a válasz típusát és kódolását, valamint az adatot ellenőrzik.

A kérés futtatása után a definiált tesztek automatikusan lefutnak, és az eredmények grafikusan is megjelennek a 2-es számmal jelzett téglalapban.

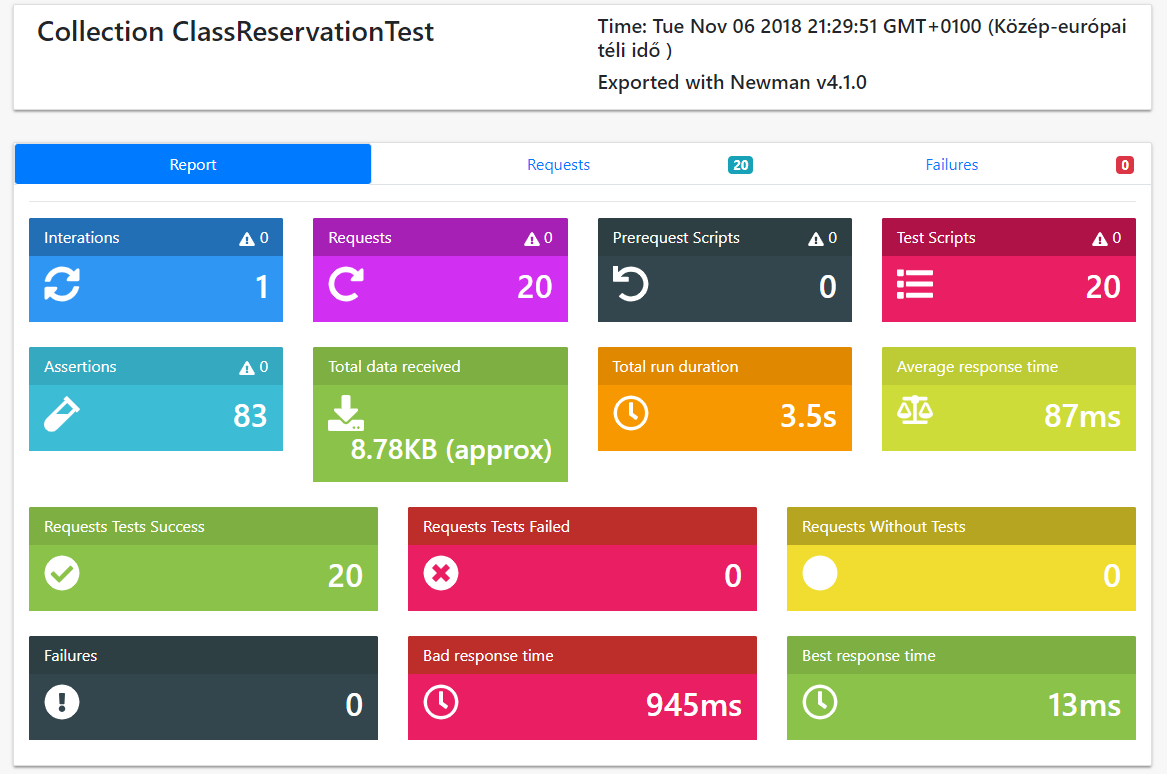
A tesztek futtatásának több módja is van, melyek az alábbiak:

* Egy adott teszt futtatása a Postman-ből.
* Egy adott gyűjtemény futtatása a Postman alkalmazásból. Ebben az esetben minden egyes, a gyűjteményben található kérés és a hozzájuk tartozó tesztek is lefutnak.
* Script segítségével történő futtatás a testing\postman mappából.
  + Az eredmények szöveges formátumban történő mentéséhez a „text\_tests.sh” script futtatása szükséges. Az eredmények a results\text könyvtárba kerülnek.
  + Ha HTML formátumban szeretnénk a teszteket megkapni, akkor a „html\_test.sh” scriptre van szükségünk, ekkor az eredmények a results\html könyvtárban fognak megjelenni.

Fontos, hogy a tesztek nem csak a kérések sikeres végrehajtását és a státuszkódot, hanem a válaszban érkező adatokat is ellenőrzik. Ebből kifolyólag a tesztesetek akkor lehetnek sikeresek, ha azok a „minta adatbázis” állapotra futnak le. Ezt az állapotot a testing\insert.sql fájl futtatásával állíthatjuk elő.

A teszteredmények áttekintésére a HTML riport ajánlott. A riport megnyitása után a html oldal betöltésre kerül a böngészőnkben. Minden riport tartalmazza a kollekció nevét és a teszt futtatásának idejét a fejlécben. A fejléc alatt három fül található, ezek sorban a „Report”, a „Requests” és a „Failures” nevet kapták.

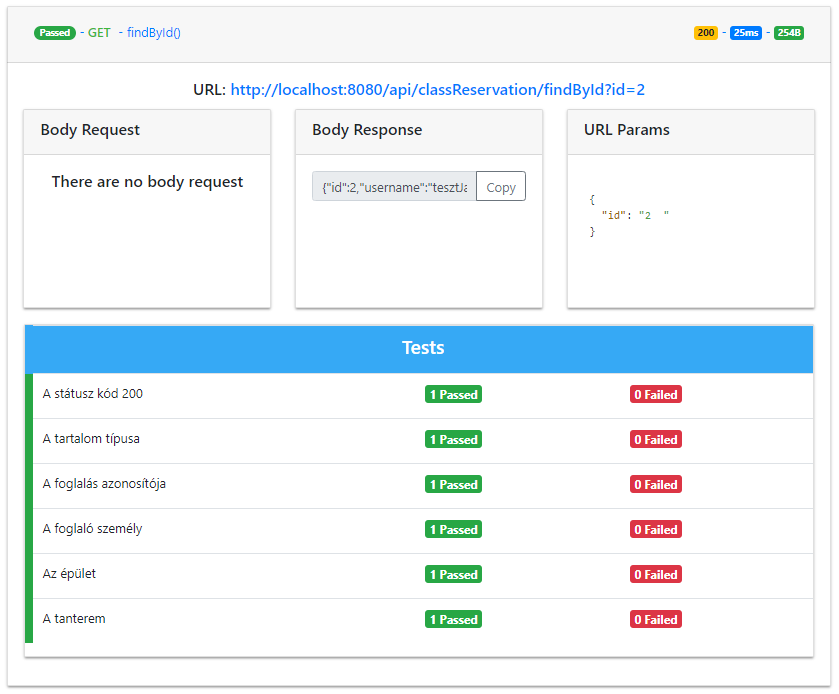
A „Report” fülön egy összefoglaló statisztika található. Itt megtekinthető többek között az iterációk, a kérések és az érvényesítések száma, valamint, hogy hány teszt futott le sikeresen, és hány bukott el.



32. ábra HTML riport összefoglalás

A „Requests” fülön a teszt által végrehajtott kéréseket lehet megvizsgálni. A kéréseket kategorizálhatjuk az alapján, hogy sikeresen, sikertelenül vagy tesztek nélkül kerültek végrehajtásra, de van lehetőség a név alapján történő keresésre is.

Egy adott kérés részletes adatait a kérésre kattintva tekinthetjük meg. Ekkor megjelenik a kéréshez tartozó URL, a kérés és a válasz törzse, valamint a paraméterek. Ezen információk alatt a tesztek táblázatos megjelenítése látható, amely magába foglalja a teszt nevét és az eredményét.



33. ábra HTML riport részletes nézet

### 4.5.3 Frontend

A megjelenítési réteg fejlesztése során törekedtem a folyamatos tesztelésre. Minden komponenst kitettem a rendeltetés szerű használat mellett extrém eseteknek is.

#### 4.5.3.1 Kézi tesztek

A kézi tesztek során a felület funkcióinak vizsgálatára helyeztem a hangsúlyt. Gyakorlatilag a kód ismerete nélkül, felhasználói szemszögből teszteltem az egyes komponenseket, űrlapokat. A tesztelés során törekedtem arra, hogy a komponenseket, űrlapokat, oldalakat a lehető legszéleskörűbben tegyem próbára. Az eredményeket listákban foglaltam össze.

##### 4.5.3.1.1 Űrlapok alapvető funkciói

* Mi történik, ha a felhasználó egy mezőt helyesen tölt ki?
  + A mező mellett található indikátor élénk zöld színűre változik.
* Mi történik, ha a felhasználó nem tölt ki egy kötelező mezőt?
  + A rendszer közli, hogy a mezőt ki kell tölteni.
* Mi történik, ha a felhasználó nem tölt ki minden kötelező mezőt?
  + Az űrlap nem lesz érvényes, a hozzáadás gomb nem válik kattinthatóvá.
* Mi történik, ha a felhasználó egy mezőt helytelenül tölt ki?
  + A mező mellett található indikátor élénk piros színűre változik. A rendszer jelzi a hibát.

##### 4.5.3.1.2 Bejelentkezés

* Mi történik, ha a felhasználó nem adja meg a felhasználónevét?
  + A bejelentkezés gomb nem aktív, amíg az űrlap nincs megfelelően kitöltve. A felhasználó nem tudja megkísérelni a bejelentkezést.
* Mi történik, ha a felhasználó nem adja meg a jelszavát?
  + Az előző esetben leírtaknak megfelelően a bejelentkezés nem lehetséges addig, amíg az űrlap nincs megfelelően kitöltve.
* Mi történik, ha a felhasználó helytelen felhasználónév/jelszó kombinációt ad meg?
  + A bejelentkezés sikertelen lesz, az űrlap tartalma törlődik. A felhasználó egy felugró ablakban értesül a sikertelen kísérletről.
* Mi történik, ha a felhasználó helyes adatokkal próbál bejelentkezni?
  + A bejelentkezés sikeres lesz, a felhasználót a rendszer a főoldalra navigálja.

##### 4.5.3.1.3 Új egyszerű foglalás létrehozása

* Mi történik, ha a felhasználó nem választ ki tantárgyat?
  + A rendszer közli, hogy a tantárgyat meg kell adni
* Mi történik, ha a felhasználó nem létező tantárgyat választ ki?
  + Ez alapvetően nem lehetséges, csak akkor, ha a felhasználó az alapértelmezett értéket választja ki, ekkor a rendszer közli, hogy ilyen tantárgy nem létezik.
* Mi történik, ha a felhasználó nem létező épületet választ ki?
  + Ez alapvetően nem lehetséges, csak akkor, ha a felhasználó az alapértelmezett értéket választja ki, ekkor a rendszer közli, hogy nem létezik ilyen épület.
* Mi történik, ha a felhasználó nem létező tantermet választ ki?
  + Ez alapvetően nem lehetséges, csak akkor, ha a felhasználó az alapértelmezett értéket választja ki, ekkor a rendszer közli, hogy nem létezik ilyen tanterem.
* Mi történik, ha a foglalás dátuma múltbeli?
  + A rendszer közli, hogy a dátum nem lehet múltbeli.
* Mi történik, ha a kezdési idő nem megfelelő hosszú?
  + A rendszer közli, hogy a kezdési időnek pontosan öt karakterből kell állnia.
* Mi történik, ha a kezdési idő nem megfelelő formátumú?
  + A rendszer közli, hogy a kezdési időt ÓÓ:PP formátumban kell megadni.
* Mi történik, ha a befejezési idő nem megfelelő hosszú?
  + A rendszer közli, hogy a befejezési időnek pontosan öt karakterből kell állnia.
* Mi történik, ha a befejezési idő nem megfelelő formátumú?
  + A rendszer közli, hogy a befejezési időt ÓÓ:PP formátumban kell megadni.

##### 4.5.3.1.4 Új szemeszterre vonatkozó foglalás létrehozása

* Mi történik, ha a felhasználó nem választ ki szemesztert?
  + A rendszer közli, hogy a szemesztert meg kell adni
* Mi történik, ha a felhasználó nem létező szemesztert választ ki?
  + Ez alapvetően nem lehetséges, csak akkor, ha a felhasználó az alapértelmezett értéket választja ki, ekkor a rendszer közli, hogy ilyen szemeszter nem létezik.
* Mi történik, ha a felhasználó nem választ ki tantárgyat?
  + A rendszer közli, hogy a tantárgyat meg kell adni
* Mi történik, ha a felhasználó nem létező tantárgyat választ ki?
  + Ez alapvetően nem lehetséges, csak akkor, ha a felhasználó az alapértelmezett értéket választja ki, ekkor a rendszer közli, hogy ilyen tantárgy nem létezik.
* Mi történik, ha a felhasználó nem létező épületet választ ki?
  + Ez alapvetően nem lehetséges, csak akkor, ha a felhasználó az alapértelmezett értéket választja ki, ekkor a rendszer közli, hogy ilyen épület nem létezik.
* Mi történik, ha a felhasználó nem létező tantermet választ ki?
  + Ez alapvetően nem lehetséges, csak akkor, ha a felhasználó az alapértelmezett értéket választja ki, ekkor a rendszer közli, hogy ilyen tanterem nem létezik.
* Mi történik, ha a felhasználó nem választ ki napot?
  + A rendszer automatikusan kiválasztja az első értéket (Hétfő), ezért ez nem okoz problémát.
* Mi történik, ha a kezdési idő nem megfelelő hosszú?
  + A rendszer közli, hogy a kezdési időnek pontosan öt karakterből kell állnia.
* Mi történik, ha a kezdési idő nem megfelelő formátumú?
  + A rendszer közli, hogy a kezdési időt ÓÓ:PP formátumban kell megadni.
* Mi történik, ha a befejezési idő nem megfelelő hosszú?
  + A rendszer közli, hogy a befejezési időnek pontosan öt karakterből kell állnia.
* Mi történik, ha a befejezési idő nem megfelelő formátumú?
  + A rendszer közli, hogy a befejezési időt ÓÓ:PP formátumban kell megadni.

##### 4.5.3.1.5 Új eseményre vonatkozó foglalás létrehozása

* Mi történik, ha a felhasználó olyan nevet ad meg, ami már foglalt?
  + A rendszer közli, hogy ezzel a névvel már létezik foglalás.
* Mi történik, ha a felhasználó nem választ ki tantárgyat?
  + A rendszer közli, hogy a tantárgyat meg kell adni
* Mi történik, ha a felhasználó nem létező tantárgyat választ ki?
  + Ez alapvetően nem lehetséges, csak akkor, ha a felhasználó az alapértelmezett értéket választja ki, ekkor a rendszer közli, hogy ilyen tantárgy nem létezik.
* Mi történik, ha a felhasználó nem létező épületet választ ki?
  + Ez alapvetően nem lehetséges, csak akkor, ha a felhasználó az alapértelmezett értéket választja ki, ekkor a rendszer közli, hogy ilyen épület nem létezik.
* Mi történik, ha a felhasználó nem létező tantermet választ ki?
  + Ez alapvetően nem lehetséges, csak akkor, ha a felhasználó az alapértelmezett értéket választja ki, ekkor a rendszer közli, hogy ilyen tanterem nem létezik.
* Mi történik, ha a foglalás dátuma múltbeli?
  + A rendszer közli, hogy a dátum nem lehet múltbeli.
* Mi történik, ha a kezdési idő nem megfelelő hosszú?
  + A rendszer közli, hogy a kezdési időnek pontosan öt karakterből kell állnia.
* Mi történik, ha a kezdési idő nem megfelelő formátumú?
  + A rendszer közli, hogy a kezdési időt ÓÓ:PP formátumban kell megadni.
* Mi történik, ha a befejezési idő nem megfelelő hosszú?
  + A rendszer közli, hogy a befejezési időnek pontosan öt karakterből kell állnia.
* Mi történik, ha a befejezési idő nem megfelelő formátumú?
  + A rendszer közli, hogy a befejezési időt ÓÓ:PP formátumban kell megadni.

##### 4.5.3.1.6 Saját tanórára vonatkozó foglalás szerkesztése

* Mi történik, ha a felhasználó nem adja meg a tárgykódot?
  + A rendszer közli, hogy a tárgykódot meg kell adni.
* Mi történik, ha a felhasználó túl rövid tárgykódot ad meg?
  + A rendszer közli, hogy a tárgykód túl rövid.
* Mi történik, ha a felhasználó túl hosszú tárgykódot ad meg?
  + A rendszer nem engedélyezi több karakter bevitelét.
* Mi történik, ha a felhasználó nem létező tárgykódot ad meg?
  + A módosítás után a rendszer hibaüzenet formájában figyelmezteti a felhasználót, a módosítások nem lépnek érvénybe.
* Mi történik, ha a felhasználó nem adja meg az épületet?
  + A rendszer közli, hogy az épületet meg kell adni.
* Mi történik, ha a felhasználó túl rövid épületnevet ad meg?
  + A rendszer közli, hogy az épület neve túl rövid.
* Mi történik, ha a felhasználó túl hosszú épületnevet ad meg?
  + A rendszer nem engedélyezi több karakter bevitelét.
* Mi történik, ha a felhasználó nem létező épületet ad meg?
  + A módosítás után a rendszer hibaüzenet formájában figyelmezteti felhasználót, a módosítások nem lépnek érvénybe.
* Mi történik, ha a felhasználó nem adja meg a tantermet?
  + A rendszer közli, hogy a tantermet meg kell adni.
* Mi történik, ha a felhasználó túl rövid tanterem nevet ad meg?
  + A rendszer közli, hogy a tanterem neve túl rövid.
* Mi történik, ha a felhasználó túl hosszú tanterem nevet ad meg?
  + A rendszer nem engedélyezi több karakter bevitelét.
* Mi történik, ha a felhasználó nem létező tantermet ad meg?
  + A módosítás után a rendszer hibaüzenet formájában figyelmezteti felhasználót, hogy a megadott épülethez nem tartozik ezzel a névvel rendelkező tanterem. A módosítások nem lépnek érvénybe.

##### 4.5.3.1.7 Saját eseményre vonatkozó foglalás szerkesztése

* Mi történik, ha a felhasználó nem adja meg az épületet?
  + A rendszer közli, hogy az épületet meg kell adni.
* Mi történik, ha a felhasználó túl rövid épületnevet ad meg?
  + A rendszer közli, hogy az épület neve túl rövid.
* Mi történik, ha a felhasználó túl hosszú épületnevet ad meg?
  + A rendszer nem engedélyezi több karakter bevitelét.
* Mi történik, ha a felhasználó nem létező épületet ad meg?
  + A módosítás után a rendszer hibaüzenet formájában figyelmezteti felhasználót, a módosítások nem lépnek érvénybe.
* Mi történik, ha a felhasználó nem adja meg a tantermet?
  + A rendszer közli, hogy a tantermet meg kell adni.
* Mi történik, ha a felhasználó túl rövid tanterem nevet ad meg?
  + A rendszer közli, hogy a tanterem neve túl rövid.
* Mi történik, ha a felhasználó túl hosszú tanterem nevet ad meg?
  + A rendszer nem engedélyezi több karakter bevitelét.
* Mi történik, ha a felhasználó nem létező tantermet ad meg?
  + A módosítás után a rendszer hibaüzenet formájában figyelmezteti felhasználót, hogy a megadott épülethez nem tartozik ezzel a névvel rendelkező tanterem. A módosítások nem lépnek érvénybe.
* Mi történik, ha a felhasználó nem adja meg a kezdési időt?
  + A rendszer közli, hogy a kezdési időt meg kell adni.
* Mi történik, ha a felhasználó túl rövid kezdési időt ad meg?
  + A rendszer közli, hogy a kezdési idő formátuma nem megfelelő.
* Mi történik, ha a felhasználó túl hosszú kezdési időt ad meg?
  + A rendszer nem engedélyezi több karakter bevitelét.
* Mi történik, ha a felhasználó nem megfelelő formátumú kezdési időt ad meg?
  + A rendszer közli, hogy a kezdési idő formátuma nem megfelelő.
* Mi történik, ha a felhasználó nem adja meg a befejezési időt?
  + A rendszer közli, hogy a befejezési időt meg kell adni.
* Mi történik, ha a felhasználó túl rövid befejezési időt ad meg?
  + A rendszer közli, hogy a befejezési idő formátuma nem megfelelő.
* Mi történik, ha a felhasználó túl hosszú befejezési időt ad meg?
  + A rendszer nem engedélyezi több karakter bevitelét.
* Mi történik, ha a felhasználó nem megfelelő formátumú befejezési időt ad meg?
  + A rendszer közli, hogy a befejezési idő formátuma nem megfelelő.

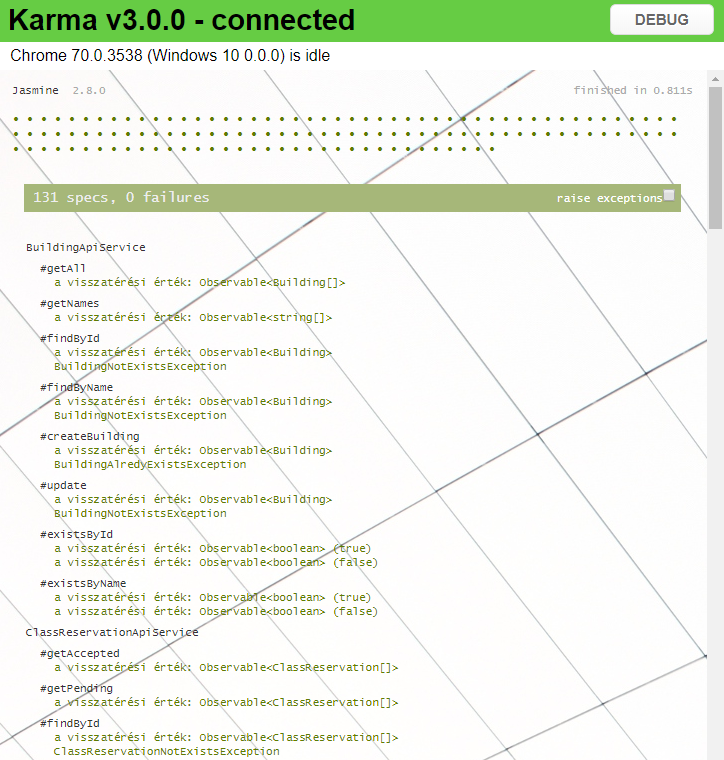
#### 4.5.3.2 Jasmine

A frontend fejlesztése során nagy hangsúlyt fektettem a szerviz osztályok tesztelésére. Minden egyes szerviz osztályhoz tartozik egy tesztfájl, amely a kiterjesztéséről ismerhető fel (spec.ts). Ezen szerviz osztályokban törekedtem az Angular beépített tesztkönyvtárainak használatára, valamint arra, hogy minden egyes végpontot próbára tegyek a lehetséges hibák kiküszöbölése végett.

Mint minden külső függőségnél, a HTTP backend esetén is szükség van a mókolásra, hogy a tesztek során képesek legyünk a távoli szerver működésének szimulálására. Ehhez a @angular/common/http/testing könyvtár hatalmas segítséget nyújtott. A tesztek felépítése, végrehajtása azonos. Először az alkalmazás végrehajtja a http kéréseket, majd a tesztek segítségével vizsgáljuk, hogy egyes kérések megtörténtek-e vagy sem. Továbbá megerősíthetjük, hogy az egyes értékek megfelelnek-e az elvárásainknak, végül biztosíthatjuk a válaszokat. A tesztesetek végén lehetőségünk van annak ellenőrzésére, hogy az alkalmazás nem hajtott végre váratlan kéréseket.

A fentebb említett tesztfájlok a src\shared\services\api könyvtárban találhatók. Futtatásukhoz használjuk a rrs-client\test.sh scriptet, vagy adjuk ki az „npm run ng test” parancsot. Bármelyik opciót is választjuk a tesztek rövid időn belül lefutnak, és az eredmények automatikusan megjelennek az alapértelmezett böngészőnk egy új ablakában.

A megjelenő riport jól áttekinthető, könnyen értelmezhető. Megkönnyíti a tesztelést, hiszen nem csak az adott osztályhoz tartozó teszteseteket tudjuk egy könnyed kattintás segítségével újra futtatni, hanem tetszőlegesen is kiválaszthatunk egyetlen tesztet, amit újra végre szeretnénk hajtani. Időközben az „ng test” parancs figyeli a változtatásokat, és amint valamilyen apró módosítást hajtunk végre a kódban, a tesztek újra lefutnak, a böngésző pedig frissül.



34. ábra A HTML teszt riport

# 5. Irodalomjegyzék

* <https://hu.wikipedia.org/wiki/NetBeans>
* <https://hu.wikipedia.org/wiki/Visual_Studio_Code>
* https://hu.wikipedia.org/wiki/Spring\_keretrendszer
* <https://hu.wikipedia.org/wiki/A_kontroll_megford%C3%ADt%C3%A1sa>
* <https://hu.wikipedia.org/wiki/Apache_Maven>
* <https://hu.wikipedia.org/wiki/Hibernate>
* <https://hu.wikipedia.org/wiki/Java_Persistence_API>
* <https://hu.wikipedia.org/wiki/MySQL>
* <https://en.wikipedia.org/wiki/Angular_(application_platform)#Version_6>
* <https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework)>
* <https://hu.wikipedia.org/wiki/Adat%C3%A1tviteli_objektum>
* <https://en.wikipedia.org/wiki/Javadoc>
* https://github.com/MarcosEllys/awesome-newman-html-template