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

Teremfoglaló rendszer

**Nagy Sára Tomecz Patrik**

mesteroktató Programtervező informatikus BSc

nappali tagozat

**Budapest, 2018**

Tartalomjegyzék

[1. Bevezetés 3](#_Toc529445732)

[2. Felhasználói dokumentáció 4](#_Toc529445733)

[2.1 A megoldott probléma 4](#_Toc529445734)

[2.2 A célközönség 4](#_Toc529445735)

[2.3 A hardveres és szoftveres követelmények 4](#_Toc529445736)

[2.4 A használati útmutató 4](#_Toc529445737)

[3. Üzemeltetői dokumentáció 5](#_Toc529445738)

[4. Fejlesztői dokumentáció 6](#_Toc529445739)

[4.1 Tesztelés 6](#_Toc529445740)

[4.1.1 Backend 6](#_Toc529445741)

# 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 valósítottam meg. Törekedtem arra, hogy a felület könnyen átlátható, felhasználóbarát legyen. A fejlesztés közben az alkalmazást folyamatosan teszteltem a hibák elkerülése végett.

Az alkalmazás felülete letisztult, bárki számára könnyen kezelhető. 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.

Mivel napjainkban sokan tableteken, telefonokon használják ezen oldalakat, ezért nagy hangsúlyt fektettem ezen eszközök támogatására.

# 2. Felhasználói dokumentáció

## 2.1 A megoldott probléma

## 2.2 A célközönség

Az alkalmazás célközönsége elsősorban a felsőoktatásban tanuló hallgatók és tanárok. A cél, hogy könnyebbé, átláthatóbbá tegyek 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.

## 2.3 A hardveres és szoftveres követelmények

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 Mozilla Firefox rendszerkövetelményei 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.

## 2.4 A használati útmutató

# 3. Üzemeltetői dokumentáció

# 4. Fejlesztői dokumentáció

## 4.1 Tesztelés

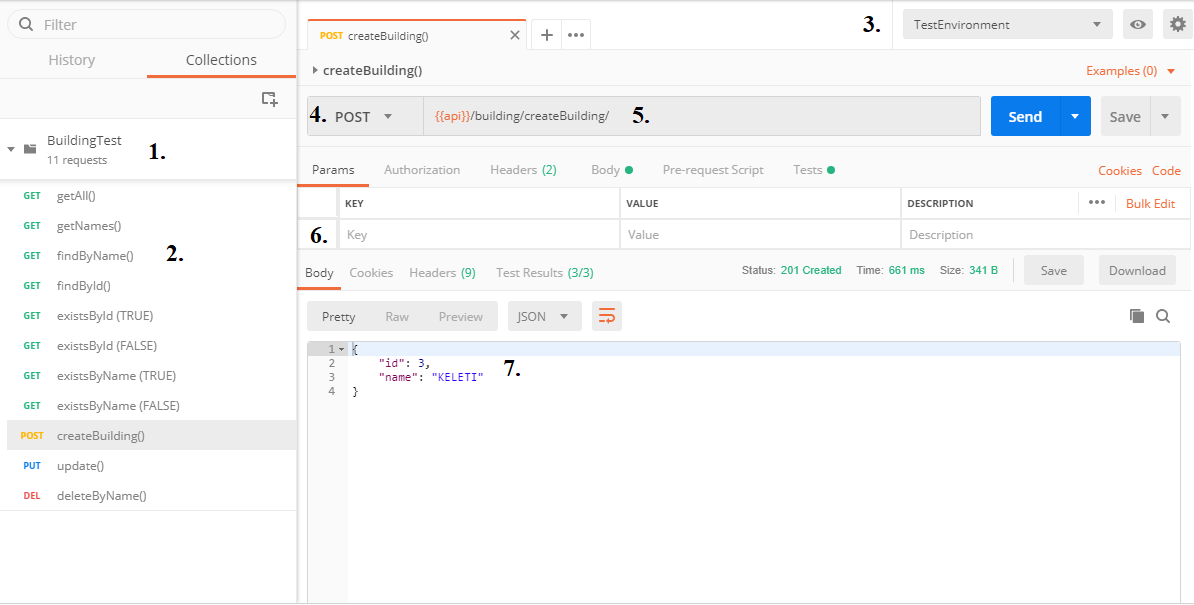
### 4.1.1 Backend

A backend tesztelése során külön kitértem a szerviz és kontroller osztályokra.

A szerviz osztályok tesztelését JUnit és Mockito keretrendszerek segítségével valósítottam meg. Ezek a tesztek automaták, minden egyes build esetén lefutnak. A tesztek 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 végpontok tesztelését a Postman nevezetű alkalmazás segítségével valósítottam meg. Azért esett erre az alkalmazásra a választásom, mert kezelőbarát felhasználói felülettel rendelkezik, automatizálható a tesztelés és széleskörű lehetőségeket biztosít.

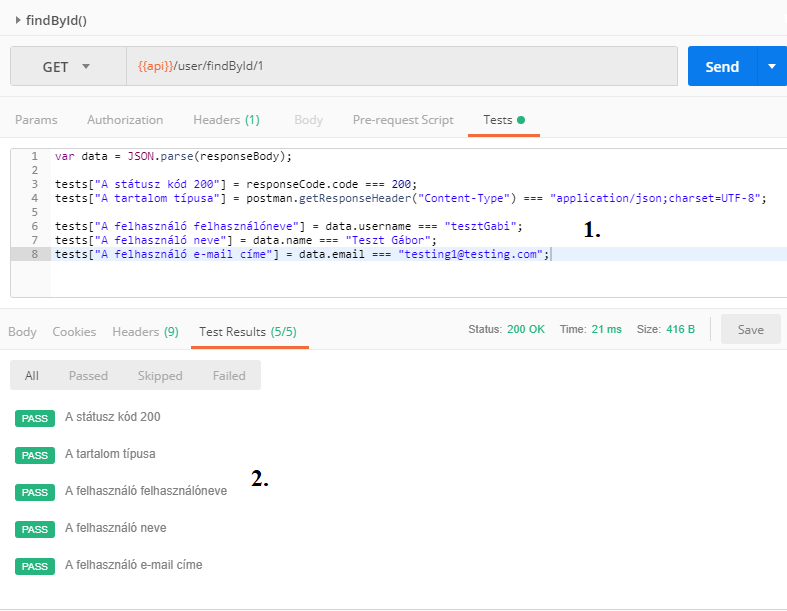
Az alkalmazás használata:



1. ábra A Postman használata

1. A megfelelő gyűjtemény kiválasztása. Ezen gyűjteményekbe szolgálnak a kérések csoportosítására, könnyen importálhatók, exportálhatok.
2. A gyűjteményen belül kiválasztjuk a megfelelő kérést.
3. Annak érdekében, hogy a kéréseket, tesztelést könnyebben, dinamikusabban tudjuk végrehajtani a Postman-ben használhatunk változókat, amelyeket a környezetekben definiálhatunk. A környezet minden kérés futtatása előtt kiválasztható.
4. A kérés típusának kiválasztása
5. A kéréshez tartozó URL megadása
6. A kérés paramétereinek megadása (opcionális). A paraméterek kulcs-érték párok, ezek segítségével generálódik a megfelelő URL.
7. A kérés elküldése
8. A válasz, melynek formátuma beállítható (pl.: JSON)

Tesztelés a Postmanben:



2. ábra Tesztelés a Postmanben

1. 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.
2. A kérés futtatása után a definiált tesztek automatikusan lefutnak és grafikusan is megjelennek az eredmények.

A tesztek futtatásának több módja is van:

1. A tesztek egyéni futtatása a Postman alkalmazásból
2. Egy adott gyűjtemény futtatása a Postman alkalmazásból. Ebben az esetben minden egyes a gyűjteményben található kérés és a hozzájuk tartozó tesztek is lefutnak.
3. Script segítségével történő futtatás
   1. Az eredmények szöveges formátumban történő mentéséhez a „text\_tests.sh” script futtatása szükséges
   2. Ha HTML formátumban szeretnénk a teszteket megkapni, akkor a „html\_test.sh” scriptre van szükségünk

Fontos, hogy a tesztek nem csak a kérések sikeres végrehajtása, a státuszkódot, hanem a válasz törzsét is ellenőrzik. Ebből kifolyólag a tesztesetek akkor lehetnek sikeresek, ha azok a minta adatbázis állapotra futnak le. Ezt az állapotot a „TEST\_DB.sql” fájl futtatásával érhetjük el.

A tesztek eredményei a mintaadatbázison HTML formátumban:

Az épületekhez tartozó végpontok teszteseteinek futási eredménye:

|  |  |
| --- | --- |
| Azonosító | Eredmény |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |
| 7 |  |
| 8 |  |
| 9 |  |
| 10 |  |
| 11 |  |

A tantermekhez tartozó végpontok teszteseteinek futási eredménye:

|  |  |
| --- | --- |
| Azonosító | Eredmény |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |
| 7 |  |
| 8 |  |
| 9 |  |
| 10 |  |
| 11 |  |
| 12 |  |
| 13 |  |
| 14 |  |
| 15 |  |
| 16 |  |
| 17 |  |
| 18 |  |
| 19 |  |
| 20 |  |

A szemeszterekhez tartozó végpontok teszteseteinek futási eredménye:

|  |  |
| --- | --- |
| Azonosító | Eredmény |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |
| 7 |  |
| 8 |  |
| 9 |  |
| 10 |  |
| 11 |  |

A tantárgyakhoz tartozó végpontok teszteseteinek futási eredménye:

|  |  |
| --- | --- |
| Azonosító | Eredmény |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |
| 7 |  |
| 8 |  |
| 9 |  |
| 10 |  |
| 11 |  |

A felhasználókhoz tartozó végpontok teszteseteinek futási eredménye:

|  |  |
| --- | --- |
| Azonosító | Eredmény |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |
| 7 |  |
| 8 |  |
| 9 |  |
| 10 |  |
| 11 |  |
| 12 |  |
| 13 |  |
| 14 |  |
| 15 |  |
| 16 |  |

A tanórára vonatkozó foglalásokhoz tartozó végpontok teszteseteinek futási eredménye:

|  |  |
| --- | --- |
| Azonosító | Eredmény |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |
| 7 |  |
| 8 |  |
| 9 |  |
| 10 |  |
| 11 |  |
| 12 |  |
| 13 |  |
| 14 |  |
| 15 |  |
| 16 |  |
| 17 |  |
| 18 |  |
| 19 |  |
| 20 |  |

Az eseményekre vonatkozó foglalásokhoz tartozó végpontok teszteseteinek futási eredménye:

|  |  |
| --- | --- |
| Azonosító | Eredmény |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |
| 7 |  |
| 8 |  |
| 9 |  |
| 10 |  |
| 11 |  |
| 12 |  |
| 13 |  |
| 14 |  |
| 15 |  |
| 16 |  |
| 17 |  |
| 18 |  |
| 19 |  |
| 20 |  |
| 21 |  |