

Programrendszerek fejlesztés projekt munka

Gáspár Tamás

April 25, 2021

Contents

1	Linkek	1
2	A fejlesztés menete	2
2.1	Alapozás	2
2.2	Felhasználói felület létrehozása	2
2.3	Authentikáció	3
2.4	Felhasználókezelés	3
2.5	Termékkezelés admin felületről	3
2.6	Termék komponens, images endpoint	4
2.7	JavaEE szerver fejlesztése	4
3	API pontok	4
3.1	NodeJS szerver	4
3.1.1	/status	4
3.1.2	/login	5
3.1.3	/logout	5
3.1.4	/authenticate	5
3.1.5	/user	5
3.1.6	/product	6
3.1.7	/images	7
3.2	JavaEE szerver	7
3.2.1	/purchase	7
3.2.2	/productTotal	8
4	Bemutató	8

1 Linkek

Az alábbiak az alkalmazáshoz tartozó legfontosabb linkek.

- A projekt GitHub repozitóriuma
- Az alkalmazás Heroku-n.

- A JavaEE szerver Heroku-n.

A NodeJS szerver adatbázisa a MongoDB Atlas-ban van hosztolva. A JavaEE szerver PostgreSQL adatbázisa az ElephantSQL ingyenes csomagjával van kitelepítve.

2 A fejlesztés menete

Itt részletesebben leírom, hogy egyes commitok mit adtak hozzá, vagy változtattak meg. Minden esetben megadok egy linket is a commitra, hogy ez ellenőrizhető legyen. A commitokban ezt a dokumentációt is megváltoztatom, de ezt nem írom itt le. Az olyan commitok nem kerülnek ide, ahol csak a dokumentációt frissítettem.

2.1 Alapozás

A commit teljes szövege: *NodeJS és angular hozzáadása. NodeJS kezdeti konfigurálás, Angular alap frontend.*

Ez a commit itt található.

Létrehoztam a projekt struktúráját, majd az Angularos és a NodeJS projekteket. Eldöntöttem, hogy az angular alkalmazást a NodeJS szerver fogja hosztolni, és kidolgoztam ennek a módját. Letöltöttem a szükséges *npm* csomagokat, és megírtam a szerver vázát. Az angular alkalmazáshoz egyszerű frontendet készítettem, ebbe fognak majd a komponensek kerülni.

2.2 Felhasználói felület létrehozása

A commit teljes szövege: *Angular frontend fejlesztése*

Ez a commit itt található.

Megalkottam a felhasználói felületet az angularban. A főkomponens tartalmazza a menüsört, és a bejelentkezési állapotot, és ezen belül jelennek meg az egyes komponensek. A hozzáadott komponensek:

- Login: bejelentkezés
- Register: regisztráció
- Home: főoldal
- Products: termékek
- About: információ

2.3 Authentikáció

A commit teljes szövege: *Felhasználókezelés megvalósítása*

Ez a commit itt található.

Kliens és szerveroldalon is implementáltam a felhasználókezelést. Ez szerver oldalon azt jelenti, hogy létrehoztam a login, logout és user endpointokat, és az felhasználó adatbázis sémáját. Angular esetén funkcionalitás került a login és register komponensek mögé, és bekerült egy guard, ami ellenőrzi a bejelentkezést.

Még itt elég sok a hiba, a későbbiekben ezen jelentősen javítok. Pl: bejelentkezés ellenőrzése service-el, nem pedig a local storage-al, vagy a loading indicator mutatása amíg a bejelentkezés tart.

2.4 Felhasználókezelés

A commit teljes szövege: *Backend: felhasználói adatok frissítése user endpointon. UI: sweetalert, material, felhasználói felületadatmódosításra*

Ez a commit itt található.

Szerver oldalon bővítettem a `/user` endpointot, hogy lehessen felhasználói jelszót változtatni, és felhasználót törölni. Kliens oldalon ehhez egy felületet hoztam létre, ahol a bejelentkezett felhasználó módosíthatja a jelszavát vagy törölheti magát.

Szépítettem az oldal kinézetét, a Google material komponensek használatával, és az alap JavaScript alert dialógus helyett egy alert könyvtárra tértem át.

2.5 Termékkezelés admin felületről

A commit teljes szövege: *Backend: product endpoint, Frontend: admin felület*

Ez a commit itt található.

Szerver oldalon létrehoztam a `/product` endpointot, a termékek módosítására és lekérésére. Ehhez egy admin felületet is létrehoztam. Ezt a felületet csak bejelentkezett admin érheti el.

Itt lehetőség van új termék létrehozására, meglévő frissítésére, és törlésére. Létrehozás és frissítés úgy történik, hogy fel lehet tölteni egy JSON fájlt, ami a termék adatait tartalmazza. Egy ilyen példa fájlt beraktam a *docs* mappába is, *imaginaryPiece.json* néven.

2.6 Termék komponens, images endpoint

A commit teljes szövege: *Backend: images endpoint, Frontend: termék komponens*

Ez a commit itt található.

A szerverhez hozzáadtam a */images* endpointot, amitől el lehet kérni az egyes termékekhez tartozó képeket. Angularban megcsináltam az egy terméket mutató komponenset. Ez a komponens elkéri a szervertől a termék képét, majd megjeleníti.

2.7 JavaEE szerver fejlesztése

A commit teljes szövege: *JavaEE szerver: vásárlások kezelése*

Ez a commit itt található.

Elkészítettem a *Spring Boot* alapú java szerveret. Ez kezeli és tárolja a felhasználók vásárlásait. A frontendhez hozzáadtam a vásárlási előzményeket. Az admin komponenset kibővítettem egy résszel, ahol a termékek bevételei lehet lekérni.

3 API pontok

Itt dokumentálom, hogy milyen API endpointokat használ az alkalmazás.

3.1 NodeJS szerver

Mindegyik endpoint a */api* mögött van. Például:

http://localhost:3080/api/status

Törekedtem arra, hogy az *API* biztonságos legyen. Ahol csak lehet beállítottam, hogy csak bejelentkezett kientől fogadjon kérést a szerver. Ahol egy felhasználó adatait módosítjuk, ott szükség van az adott felhasználó felhasználónevére és jelszavára. Komoly következményekkel járó hívások, pl termékek hozzáadása vagy törlése esetén admin felhasználónév és jelszó kell.

3.1.1 /status

Ez csak GET-re válaszol, és egyszerűen visszaküld egy kis szöveget. Arra használható, hogy megnézzük hogy működik-e a szerver. Az endpointok a nodeJS mappa endpoints almappájában vannak definiálva, kivéve a */status*, mert az annyira egyszerű.

3.1.2 /login

Ide csak POST-olni lehet, és meg kell adni a felhasználónevet és jelszót. Ha ezek jók, akkor a bejelentkeztetés megtörténik.

Az adatokat JSON-ban várja:

```
{
  "username": "valaki",
  "password": "valaki_jelszava"
}
```

A választ JSONben küldi:

```
{
  "message": "Pl Sikeres bejelentkezés",
  "isAdmin": "false"
}
```

Az angular nézi az *isAdmin* értékét, hogy tudja, hogy admin jelentkezett-e be.

3.1.3 /logout

A */login* párja, ami kijelentkezteti a felhasználót. Csak POST-ot fogad, nem vár semmilyen adatot és akkor sem dob hibát, ha nem volt senki bejelentkezve.

3.1.4 /authenticate

A */login*-hoz hasonlóan működik, viszont nem jelentkezteti be a felhasználót, csak megmondja, hogy a kapott adatok érvényesek-e. A *JavaEE* szerver ezt hívja meg, hogy ellenőrizze a kapott felhasználóneveket és jelszavakat.

Csak *POST*-ot fogad, a kérés törzsében meg kell adni a felhasználónevet és a jelszót (pont mint a */login* esetén).

Eredménye egy *JSON*, ami a következőképpen néz ki:

```
{
  "message": "valid",
  "isAdmin": "false"
}
```

3.1.5 /user

Felhasználókezelő endpoint, ami POST-ot, PUT-ot és DELETE-et támogat.

POST esetén regisztráció történik. JSON-ban meg kell adni a felhasználónevet és jelszót (úgy, mint a */login* esetén).

PUT esetén a jelszó frissíthető, itt meg kell adni a felhasználónevet és **jelenlegi** jelszót, majd az új jelszót:

```
{
  "username": "valaki",
  "password": "valaki_jelszava",
  "newPassword": "valaki_uj_jelszava"
}
```

Csak akkor lesz változtatás, ha a felhasználó be van jelentkezve, és a jelenlegi jelszava egyezik.

DELETE-tel felhasználó törlés kell. PUT-hoz hasonlóan itt is küldeni kell a felhasználónevet és a jelszót, csak akkor fog végrehajtódni, ha be vagyunk jelentkezve és a jelszó helyes.

3.1.6 /product

Ezen az endpointon kérhetőek le, vagy módosíthatóak a termékek. Mindegyik híváshoz be kell, hogy jelentkezünk. Azokat hívásokat, amik a termékek adatbázisát megváltoztatják, csak admin hajthatja végre.

GET: Ez lekéri az összes terméket, amiket egy JSON fájlban küld vissza.

POST (admin): Új termék hozható létre vele. Várja a hitelesítő adatokat, és a termék adatait, a következő formában:

```
{
  "username": "valaki_admin",
  "password": "valaki_jelszava",
  "name": "termek neve",
  "description": "termek leirasa",
  "price": "termek ara",
  "imgPath": "utvonal a termék képere"
}
```

PUT (admin): POST-hoz hasonló, de itt a terméknek már léteznie kell, és az adatai frissítve lesznek. Pont ugyanolyan formában várja az inputot, mint a POST.

DELETE (admin): Termék törlése. Az inputot a következő formában várja:

```
{
  "username": "valaki_admin",
  "password": "valaki_jelszava",
  "name": "torlendo termék neve",
}
```

Az alkalmazás biztosít az admin felületen opciót *JSON* feltöltésére, amivel egy új terméket lehet megadni vagy egy meglévőt frissíteni. Egy ilyen példa *JSON*-t beraktam a *docs* mappába *imaginaryPiece.json* néven.

3.1.7 /images

Ez csak *GET*-et fogad, és lehetőség van vele lekérni a termékekhez tartozó képeket. Az *URL*-ben egy *imageName* nevű paramétert kell megadni, ami megmondja a szervernek, hogy melyik képet kell visszaküldeni. Az angular ezt a termék *imgPath* attribútumából állítja elő.

Például így lehet a 'Gyalog' nevű termék képét elkérni (debug módban):

http://localhost:3080/api/images?imageName=pawn.png

Csak bejelentkezett kientől fogad kéréseket. A válasz maga a kép lesz (*Content-Type: image/jpeg*).

3.2 JavaEE szerver

A NodeJS szerverhez hasonlóan itt is a */api* mögött vannak az endpointok. Itt is törekedtem a biztonságossá tételre, felhasználónevek és jelszavak kérésével.

3.2.1 /purchase

Vásárlásokat kezelő endpoint. **POST** esetén elment egy vásárlást. A következő formájú *JSON*-t várja:

```
{
  "username": "valaki",
  "password": "valaki_jelszava",
  "productName": "termek_neve",
  "price": "termek_ara",
  "dateTime": "vasarlas_idopontja"
}
```

A termék ára itt még szövegesen kell hogy legyen, pl *1000 Ft*. Ezt majd a szerver alakítja számmá. A *dateTime* attribútum a vásárlás időpontja (*UNIX* timestamp).

GET krést is fogad, ezzel egy felhasználó vásárlásait lehet lekérni. Meg kell adni a felhasználónevet és jelszót. Például:

http://localhost:8080/api/purchase?username=valaki&password=valaki_jelszava

A hívás eredménye egy *JSON* tömb, amiben a vásárlások vannak felsorolva.

3.2.2 /productTotal

Ez az endpoint csak **GET**-et fogad, és csak admin által hívható. A következő formában várja a paramétereiket:

```
http://localhost:8080/api/productTotal?username=admin  
      &password=admin_jelszava&productName=termek_neve
```

JSON tömbben visszaadja az összes vásárlást, ahol az adott terméket vették meg. A teljes bevétel és az utolsó vásárlás meghatározását már az angular végzi.

4 Bemutató

Ahogy a követelményekben megvan, lehetőség van a belépni a következő felhasználóval:

- Username: szaboz
- Jelszó: PRF2021

Ez a felhasználó egyben admin is, ezért az admin felületet is ki lehet próbálni vele. Lehet regisztrálni persze más felhasználót is, az nem lesz admin.

Az első *HTTP* kérés a NodeJS és a JavaEE szerverek felé beletelik egy kis időbe, mivel a Heroku "altatja" az alkalmazást, ha egy ideje nem használták. Van azonban töltést jelző elem, ezért látszik, hogy mikor vár éppen válaszra.

A bal felső sarokban lévő menüvel választhatunk komponenst. Mindenhova viszonylag jól kinéző felhasználói felületet készítettem, ezért a komponensek használata (pl: vásárlás, jelszó csere) elég egyszerű.