Programrendszerek fejlesztése gyakorlat
Kötelező program dokumentáció

Fejlesztés menete:

- Szükséges függőségek telepítése, mint pl. NodeJS
- Angular projekt létrehozása, `ng new` parancs segítségével

- **NodeJS + MongoDB**
- MongoDB organizáció és egy cluster létrehozása
- Node mappa létrehozása, `npm init`
  Szükséges modulok telepítése `npm install`:

```
"dependencies": {
    "bcryptjs": "^2.4.3",
    "body-parser": "^1.19.0",
    "cookie-parser": "^1.4.5",
    "cors": "^2.8.5",
    "express": "^4.17.1",
    "express-session": "^1.17.1",
    "mongoose": "^5.12.9",
    "nodemon": "^2.0.7",
    "passport": "^0.4.1",
    "passport-local": "^1.0.0"
},
```

- Csatlakoztatás a MongoDB adatbázishoz
- Backend elgondolása. Nem lett túlbonyolítva, létrehoztam hozzá az angularos mappában egy interface-t, hogy a későbbiekben könnyebb legyen használni. A node-os mappában pedig definiálok egy sémát a mongodb használatához

item.model.ts

shop-angular > src > app > models > TS

```
export interface Item {
    id: string;
    name: string;
    desc?: string;
    price: number;
}
```

```
const mongoose = require('mongoose');

var itemSchema = new mongoose.Schema({
    id: { type: String, unique: true, required: true, lowercase: true },
    name: { type: String, required: true },
    desc: { type: String },
    price: { type: Number, required: true }
}, { collection: 'item' });

mongoose.model('item', itemSchema);
```

- A felhasználóknak szintén létrehozok egy sémát:

```
var userSchema = new mongoose.Schema({
    username: { type: String, unique: true, required: true, lowercase: true },
    password: { type: String, required: true },
    email: { type: String, required: true },
    accessLevel: { type: String }
}, { collection: 'users' });
```

Az accesslevelt, minden felhasználónak automatikusan ’user’-re állítom. Illetve itt van definiálva az a függvény amelyik ellenőrzi, hogy a titkosított jelszó egyezik-e a belépéskor megadott jelszóval.

- REST végpontok megírása
  Ehhez a node-os mappában létrehoztam két file-t. Az egyik a user.js ami a felhasználók regisztrációját, bejelentkezését és autentikációját oldja meg. Az autentikáció passport segítségével lett megvalósítva. Itt van megoldva a login és logout is. POST metódussal lehetőség van a felhasználó adatainak megváltoztatására.
  A másik az item.js amelyben lekérhetők az eltárolt árucikkek (item), illetve lehet keresni árucikket id alapján és meg van valósítva a REST interface, azaz van GET, POST, PUT és DELETE végpont
  Server oldalon is ellenőrizve van, hogy csak beléptetett felhasználó fér az adatokhoz. Módosítani csak admin tud.

```
req.isAuthenticated() && req.session.passport.user.accessLevel == "admin"
```

- A CORS probléma whitelist-el lett megoldva.

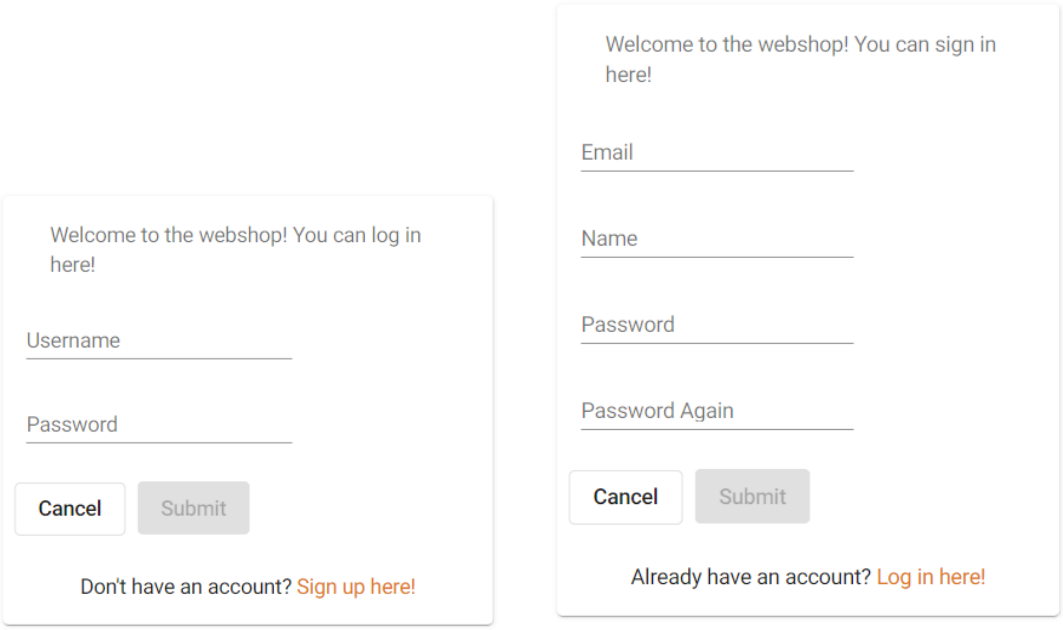
- **Frontend: Angular**
- Auth komponens: login és regisztráció, autentikáció. AuthGuard létrehozása, auth komponensre irányít, ha nem megfelelő oldalra navigálunk.
- Error komponens: csak egy sima 404 szöveget ír ki, ha nem létező oldalra próbálunk navigálni
- Home komponens: a főoldalunk. Kilistázza az adatbázisban található árucikkeket, amiket hozzá tudunk adni a kosarunkhoz.
- Cart komponens: a kosarunkban található árucikkek listázása. Módosíthatunk a kosarunk tartalmán vásárlás előtt.
- Transactions komponens: kilistázza az eddigi tranzakciókat
- Servicek megírása:
- authService a felhasználókezelésre – NodeJS szerverrel való kapcsolat
- itemService: árucikkek és tranzakciók kezelése – árucikket a NodeJS szervertől és a Java Springes szerevertől is le tudunk kérni; tranzakciók – Java Spring

- **Java Spring szerver + PostgreSQL:**
- Spring initializr
- Bean osztályok létrehozása: Item és Transaction
- Tranaction dátum mezője `LocalDateTime` amit Angulárban könnyű kezelni egy date pipe segítségével, viszont fejlesztés közben Postman-ben ügyelni kell hogy megfelelő formátumú sztringet adjunk meg, amit a Spring le tud kezelni.
- Ezekhez Repository interface amely bővíti a JpaRepository-t
- Service interface, CRUD műveletek definiálása
- Service interface implementálása
- Végül Controller osztályok, a REST végpontok kezelésére
- CORS problémára megoldás: `@CrossOrigin(origins = "*")`

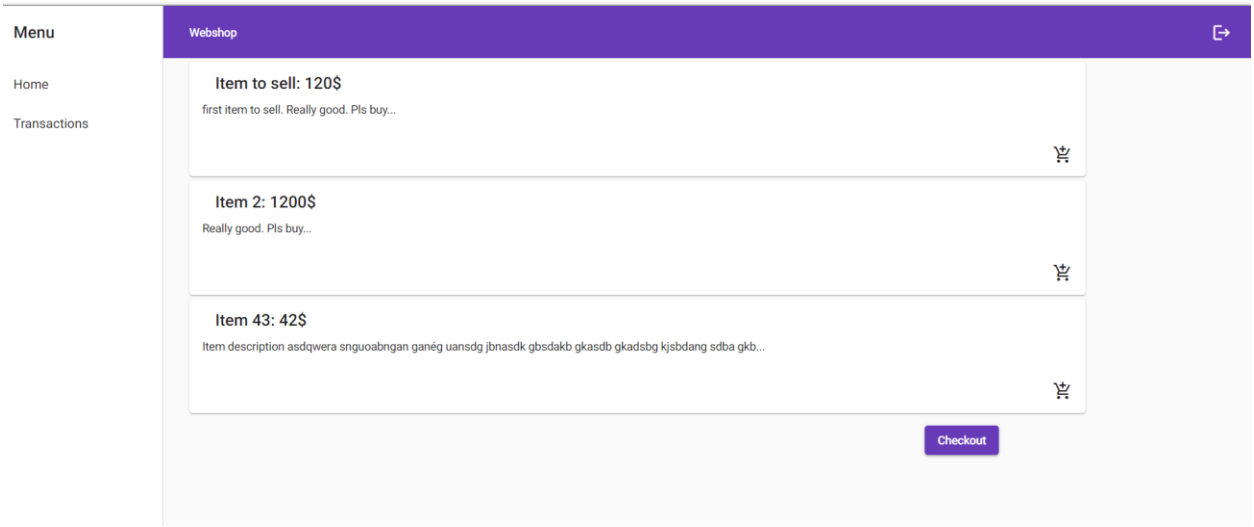
- **Hostolás:**
- Java Spring szerver Heroku-ra van deploy-olva
  Az adatbázistáblákat és szekvenciákat a pgAdmin program segítségével hoztam létre a Heroku által készített PostgreSQL adatbázisban.
  A szerver elérhető a https://guarded-ridge-09602.herokuapp.com linken
  `/items` és `/transactions` útvonalakkal lekérhetők az adatbázisban tárolt árucikkek és tranzakciók
- A NodeJS szerver Heroku-ra van deploy-olva
- NodeJS szerver hostolja a lebuildelt Angular appot is
  Azért ezt a megoldást választottam mert így kisebb az esély a CORS hibára. Viszont ennek van pár nemvárt mellékhatása.
- Az oldal frissítésekor azt a NodeJS szerver által visszaadott üzenetet kapjuk, miszerint „A kert eroforras nem talalhato”. Ez azért van mert a NodeJS egy nem definiált utvonalat próbál értelmezni és a middle-ware lánc utolsó eleme fut le. Ilyenkor ki kell törölnünk a böngészőben a az URL-ből az utvonalakat.
- Egyéb probléma, hogy első alkalommal mikor belépünk az oldalra és listázzuk a tranzakciókat a Tranzakciók oldalon, elég sokat kell várni mire válaszol a szerver. Ennek nem tudom mi lehet az oka, talán hogy a Heroku egy Amerikai szervert készített a `heroku create` során.
- Az alkalmazás linkje: https://peaceful-temple-90106.herokuapp.com

Az alkalmazás

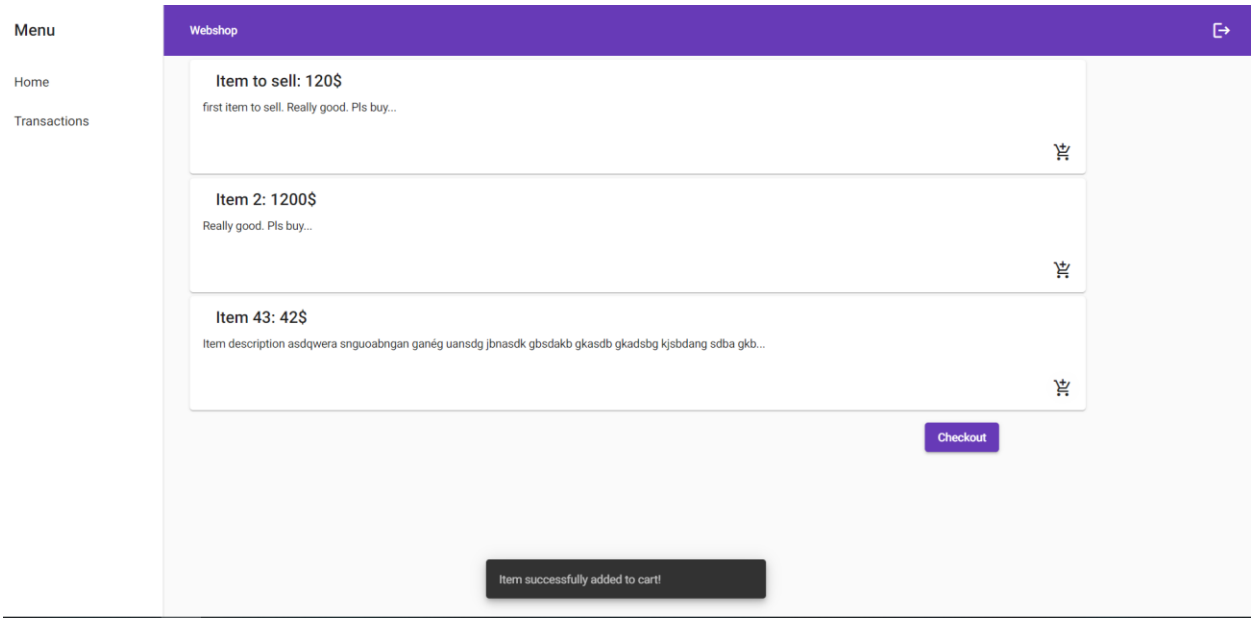
Belépéskor Login vagy Regisztráció:



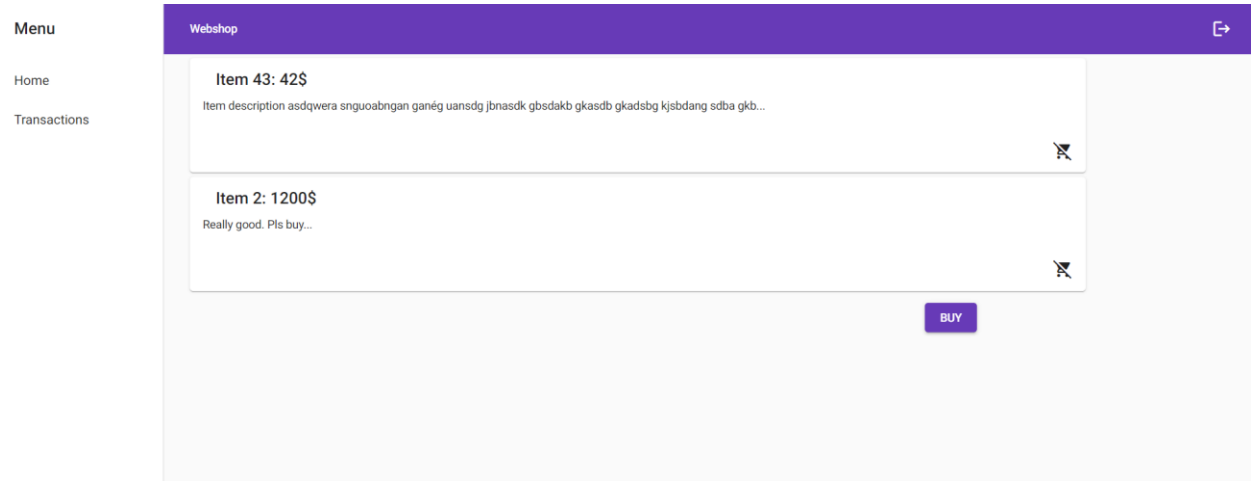
Belépéshez használható a: `szaboz` felhasználónév és a: `PRF2021` jelszó.



Baloldalt látható egy responzív menüsáv a kezdőoldallal és a tranzakciós oldallal. Jobb felső sarok: kijelentkezés. Középen pedig láthatóak az adatbázisban tárolt elemek. A kis kosárra kattintva egy árucikket hozzáadhatunk a kosarunkhoz:

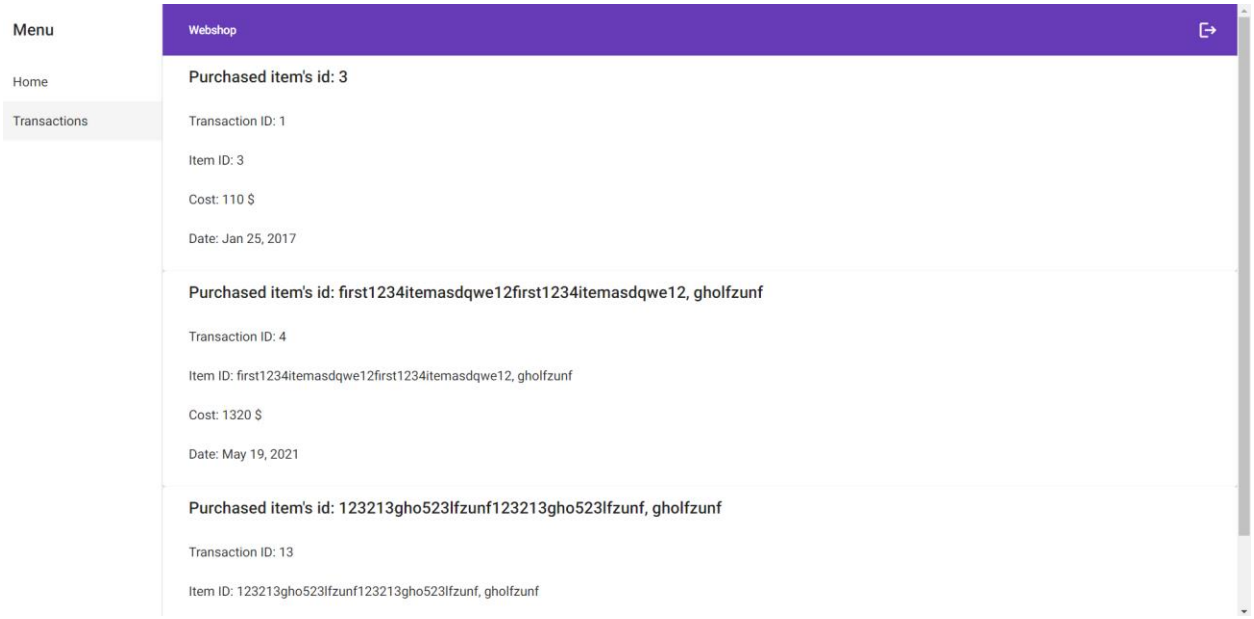


A következő lépés a kosár megtekintése: Checkout gomb



Láthatóak a kosárban szereplő cikkek. A kis ikonra kattintva törölhető egy cikk a kosárból. Buy gomb megnyomásával kapunk egy felugró üzenetet és visszatérünk a kezdőlapra. Továbbá létrejön az adatbázisban (Java Spring + PostgreSQL) egy tranzakció a vásárolt cikekk azonosítójainak listájával, az vásárolt áruk értékének az összegével és egy dátummal.

A tranzakciós menüpont:



A Java Spring + PostgreSQL adatbázisban nem lett megvalósítva az árucikk hozzáadása, amennyiben még nem szerepelt tranzakcióban. Viszont van hozzá REST végpont, illetve az Angularban az itemService-nek van egy `getSoldItems` metódusa.

A hostolt erőforrások linkjei:

Java Spring + PostrgreSQL: https://guarded-ridge-09602.herokuapp.com

- Elérhető útvonalak: /items , /transactions

NodeJS + MongoDB + Angular: https://peaceful-temple-90106.herokuapp.com

- Elérhető útvonalak: / , /items , /users