győri szakképzési centrum

Jedlik Ányos
Gépipari és Informatikai
Technikum és Kollégium

9021 Győr, Szent István út 7.
+36 (96) 529-480
+36 (96) 529-448

OM: 203037/003
jedlik@jedlik.eu
www.jedlik.eu

Záródolgozat feladatkiírás

Tanulók nevei:	Bédi Alexandra, Molnár Máté
Képzés:	nappali
Szak:	5 0613 12 03 Szoftverfejlesztő és -tesztelő technikus

A záródolgozat címe:

„Agora” termelői és kézműves piactér webalkalmazás

Konzulens:	Sándor László
Beadási határidő:	2022. 04. 29

Győr, 2022. 04. 29.

Módos Gábor
igazgató

Konzultációs lap

	A konzultáció		Konzulens aláírása
	ideje	témája	
1.	2022.02.15.	Témaválasztás és specifikáció	
2.	2022.03.14.	Záródolgozat készültségi fokának értékelése	
3.	2022.04.17.	Dokumentáció véglegesítése	

Tulajdonosi nyilatkozat

Ez a dolgozat a saját munkánk eredménye. Dolgozatunk azon részeit, melyeket más szerzők munkájából vettük át, egyértelműen megjelöltük.

Ha kiderülne, hogy ez a nyilatkozat valótlan, tudomásul vesszük, hogy a szakmai vizsgabizottság a szakmai vizsgáról minket kizár és szakmai vizsgát csak új záródolgozat készítése után tehetünk.

Győr, 2022. április 29.

Tanulók aláírásai:

_____ _____

Bédi Alexandra Molnár Máté

Tartalom

1 Bevezetés

1.1 A szoftver célja

Az Agora weboldal online piacteret kínál kistermelőknek, kézműveseknek és kis üzleteknek. A szoftver célja az internetes vásárlást és eladást egyszerűvé tenni ezekben a közösségekben. A felhasználókat összekötni a közvetlen közelükben megtermelt környezetbarát és erkölcsösen beszerzett termékekkel. Sok esetben a jó minőségű és a környezetet a lehető legkisebb mértékben roncsoló árucikkeket megtalálni időigényes és nehéz feladat, a webáruház ebben kíván segítséget nyújtani. Az eladók nagyobb vásárlóbázishoz juthatnak, emberek nagyobb rétegét érhetik el, akik szívesen támogatják őket, míg a felhasználók felfedezhetik, mennyi mindent megtalálhatnak lakhelyük közelében.

Az oldalon keresztül reméljük, hogy tartós kapcsolatok is kialakulnak, lesznek, akik hétről hétre, évről évre az Agora piacterén szerzik be a számukra szükséges élelmiszereket, alapanyagokat, ajándékokat, ruhákat. Szeretnénk minél több felhasználót arra inspirálni, hogy belevágjon vállalkozásába, ha kezdetben, kicsiben is és egy támogató közösségre találjon. Az idő elteltével, akár termelési láncok is kialakulhatnak, például egy kistermelő barackjából lekvárt főzünk, és azt továbbadjuk a helyi cukrászdának, ahol 100%-ban lokálisan beszerzett alapanyagokból készített zserbót árusíthatnak, így teljesen felszámolva a szállítással járó káros anyag kibocsátást.

1.2 Célkitűzés

A tervezési fázisban célként egy olyan komplex webalkalmazás fejlesztését tűztük ki, amin keresztül a teljes vásárlási folyamat végbe vihető és több árus feltölthet termékeket. Ehhez szükséges volt a fiók kezelés felépítése, a regisztráció és bejelentkezés. A kosár, szállítási adatok megadása, rendeléslövetés funkciók a vásárlás lebonyolításához kellettek. Annak érdekében, hogy a felhasználók az igényeiknek leginkább megfelelő termékeket találják meg kerültek hozzáadásra az értékelések szöveges és numerikus alakban is.

Eladói oldalról is cél volt egyszerű, de szolgáltatásokban gazdag felületet nyújtani, például a termékek lehessenek akciósok, nyilvános és privát láthatósággal publikálhatók. A kistermelői és kézműves közönséget figyelemben tartva az alapanyagok és címkék megadása is központi szerepet kapott, hiszen, így tudják kiemelni termékeik legelőnyösebb oldalát, hogy azok helyben, környezetbarát módon készültek.

Emellett a közösség és kapcsolatépítés is szempont volt, a felhasználók kedvenc eladóikat követni tudják estleg kommunikálni velük a platformon keresztül.

1.3 Fejlesztői csapat

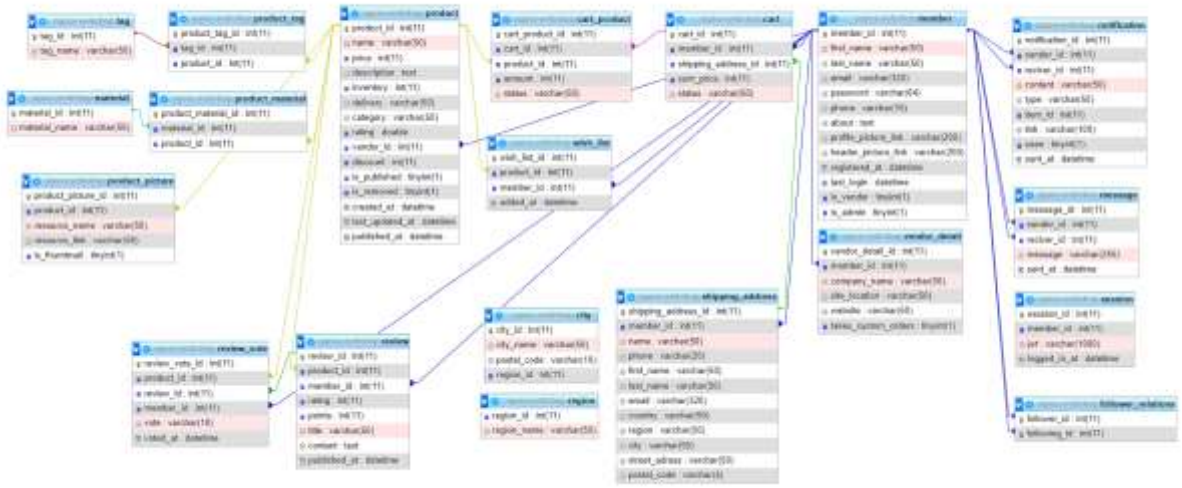
A fejlesztést hárman kezdtük meg, de egy két főből, Bédi Alexandrából és Molnár Mátéból álló csapatban fejeztük be. A munkamegosztás funkciók, működési folyamatok alapján történt a backend/frontend megkülönböztetés helyett, így nem volt szükséges hosszabb ideig a csapat másik tagjára várni, de ütközések sem alakultak ki a projectben.

2 Adatbázis

2.1 Adatmodell

A MySQL eredeti fejlesztői által készített, nyílt forráskódú MariaDB relációs adatbázist használja a weboldal az adatok tárolására és lekérdezésére.

Az adatmodell az oldal implementált vagy néhány esetben csak tervezett funkcióinak kiszolgálását biztosító adatbázis szerkezetét írja le.



Adatmodell táblái és kapcsolataik

2.2 Tárolt eljárások

A backend a legtöbb esetben egy tárolt eljárást hív meg az SQL szerveren, ennek köszönhetően a kód letisztultabb és elkerülhető a több kérés egyszerre hívásával járó biztonsági rés. Egy termék összes adatának lekéréséhez például öt SQL utasításra van szükség, az ezekre kapott adatokat egyetlen eljáráshívás adja vissza, így egyszerűbb feldolgozni azt.

Összetettebb tárolt eljárások is használatban vannak, ilyen például a termékek felvételét kezelő. A termékek alapanyagai, címkéi és a hozzájuk tartozó képekre mutató linkek mind tömbökben kerülnek tárolásra, így szerepelnek a http kommunikáció során küldött JSON állományokban is. Azonban a mySQL nem támogat tömb típust, az nem adható át egy eljárásnak. A tömb típusú adatokat karakterlánccá kellett átalakítani, melyekből az adatbázis ideiglenes táblákat tud létrehozni. Ezek a táblák már használhatók egy tömbhöz hasonlóan, végig lehet iterálni a benne szereplő adatokon.

Az alapanyagokat a táblába beszúró eljárás bemeneti paraméterként fogadja a karakterlánccá alakított tömböt, annak hosszát és a termék egyedi azonosítóját, melyhez az anyagok társításra kerülnek. Az ideiglenes tábla létrehozását és feltöltését követően egy

WHILE ciklus megy végig azon. Ha olyan alapanyaghoz ér, ami még nem szerepel az adatbázisban, az hozzáadásra kerül a `materials` táblához, ha már szerepel, akkor csak a több a többhöz kapcsolatot fenntartó táblába lesznek beillesztve a megfelelő azonosítók.

```
BEGIN
    DECLARE x INT DEFAULT 1;
    CREATE TEMPORARY TABLE materials(i INT, name VARCHAR(50));
    SET @sql = CONCAT('INSERT INTO materials(i, name) VALUES ', mats);
    PREPARE stmt FROM @sql;
    EXECUTE stmt;
    DEALLOCATE PREPARE stmt;
    WHILE (x < (length + 1)) DO
        SET @currMatName = (SELECT name FROM materials WHERE materials.i = x);
        IF (SELECT EXISTS (SELECT * from material WHERE material.material_name
= @currMatName)) THEN
            INSERT INTO product_material(material_id, product_id) VALUES(
(SELECT material_id FROM material WHERE material_name = @currMatName),
prodId);
        ELSE
            INSERT INTO material(material_name) VALUES(@currMatName);
            INSERT INTO product_material(material_id, product_id) VALUES(
(SELECT material_id FROM material WHERE material_name = @currMatName),
prodId);
        END IF;
        SET x = x + 1;
    END WHILE;
END
```

Alapanyagok beszúrását végző tárolt eljárás

Ehhez hasonló a többi tömbként tárolt adat beszúrását végző eljárás is, melyeket a terméket hozzáadó tárolt eljárás hív meg.

Mikor egy eladó új publikus láthatóságú terméket tesz közzé az őt követő felhasználók számára értesítések lesznek kiküldve, így nem csak a termék adatait kell az adatbázisnak fogadnia. Ilyenkor egy eljárás megkeresi az eladót követő felhasználókat és egy ciklussal végig iterál rajtuk, mentve az értesítésben a termék egyedi azonosítóját, amivel a felhasználónak az új árucikkre mutató link képezhető.

```
BEGIN
    DECLARE n INT DEFAULT 0;
    DECLARE i INT DEFAULT 0;
    SELECT COUNT(*) FROM follower_relations WHERE
follower_relations.following_id = userId INTO n;
    WHILE i<n DO
        CALL insertNotification( userId, (SELECT
follower_relations.follower_id FROM follower_relations WHERE
follower_relations.following_id = userId LIMIT i,1), 'product', prodId);
        SET i = i + 1;
    END WHILE;
END
```

Értesítéseket beszúró ciklus

Három megkülönbözetett típusú értesítés létezik a programban. Az `insertNotification` eljárás hívható *„product”* paraméter helyett *„order-tracking”* vagy *„order-arrived”* bemenettel is, így egyszerűn beállítva az értesítés szöveges tartalmát, valamint a hozzá tartozó linket.

2.3 Tesztelés

Annak érdekében, hogy könnyen tesztelni lehessen az adatbázist és a szoftver többi részét is, készült egy random tesztadatokat generáló konzolos alkalmazás. A .NET-ben Entity Framework használatával létrehozott szoftverben csak a kívánt adatok mennyiségét kell megadni és azokat hozzá is adja az adatbázishoz.

Az *adatbázis* mappában megtalálható egy csak a futáshoz szükséges adatokat tartalmazó kiexportált SQL állomány, valamint egy tesztadatokkal feltöltött verzió is.

3 Backend

3.1 Keretrendszer és könyvtárak

A Node.js aszinkron futású, esemény vezérelt platformra épülő Express keretrendszerben került fejlesztésre a http kéréseket kiszolgáló backend. A mysql nevű Node.js modul segítségével történik a kommunikáció a MySQL (MariaDB) adatbázissal, míg a Multer csomag nyújtotta *middleware* a fájlfeltöltést kezeli.

3.2 Útválasztás

A backend alkalmazás egy szervert indít a 3080-as porton, ami beérkező kéréseket vár meghatározott végpontokon. Az útvonalak definiálása egy Express objektum függvényeivel lehetségesek. A különböző típusú http kérések mindegyikéhez tartozik egy-egy metódus, mely annak fogadására szolgál. Ezek a függvények egy a végpont címét meghatározó karakterláncot várnak, ebben kettőspontot követően akár egy paraméter is megadható. Emellett visszahívó függvényt szükséges megadni, mely az adott végpont hívásakor fut le.

```
const express = require('express');
const shopController = require('../controllers/shop');

const router = express.Router();

router.get('/product/:id', shopController.getProduct);
router.get('/review/:id', shopController.getReviewById);
router.get('/products', shopController.getAllProducts);

module.exports = router;
```

Részlet a shop.js állományból

Az alkalmazásban külön fájlokban funkcióijuk és elérhetőségük alapján kerültek csoportosításra a végpontokat meghatározó metódusok, hogy egyszerűbben áttekinthetők legyenek. A példában látható sorok adott végpontra küldött, GET metódussal hívott http kéréseket kötnek össze a `shopController` néven beimportált állományban szereplő visszahívó függvényekkel.

3.3 Útvonalkezelők

A végpontokra érkező kéréseket feldolgozó visszahívó függvények az útvonalkezelők, melyek lényegében kliens-szerver összekötő köztes szoftverek. Paraméterként egy kérés, illetve válasz objektumot kapnak. A legtöbb esetben az authentikációt követően az adatbázissal lépnek kapcsolatba és JSON formátumban egy választ küldenek vissza, mely a szükséges adatokat tartalmazza. Egy ilyen, egy értékelést az adatbázisból kiolvasó útvonalkezelő függvény, így néz ki:

```
exports.getReviewById = (req, res, next) => {
  conn = connectDb()
  if (!Number(req.params.id)) {
    res.status(400);
    return res.json({ 'message': 'ID must be a number' });
  }

  let id = Number(req.params.id);

  conn.query('CALL selectReview(?, ?)',
    [id, 0], (err, rows, fields) => {
      if (err) {
        if (err.code = 'ECONNREFUSED') {
          res.status(503);
          return res.json({ 'message': 'Connection refused by database
server.' })
        }
        return res.json({ 'message': err })
      } else {
        res.json(rows[0]);
      }
    }
  );
  conn.end();
};
```

Egy értékelést továbbító útvonalkezelő függvény

A URL-ben az id változó helyén szereplő rész átadásra kerül a függvénynek, ami ellenőrzi, hogy az valóban szám-e. Ha így van, a mysql csomag segítségével kérést intéz az adatbázis felé. A kérés egy tárolt eljárást hív, aminek felparaméterezése biztonságos módon történik az SQL injekciós támadás elkerülése érdekében. Amennyiben a kért mezők megérkeznek azok JSON formátumban visszaküldésre kerülnek, hiba esetén is szabványos http válasz lesz továbbítva.

Egy termék hozzáadásakor, a már említett okokból karakterlánccá kell alakítani tömböket olyan módon, hogy azt a tárolt eljárás fel tudja dolgozni. Ezt a backenden végzi a következő függvény:

```
function stringifyArray(array) {
  let strArray = '';
  for (let index = 0; index < array.length; index++) {
    const element = array[index];
    strArray += ("(" + (index + 1) + ",\'" + element + "\'");
    if (index == (array.length - 1)) {
      strArray += ")";
    }
    else {
      strArray += "),";
    }
  }
  return strArray;
};
```

A tömböt átalakító függvény

3.4 Tesztelés

A backend alkalmazás tesztelése a Thunder Client nevű VS Code kiegészítővel történt, a lefutatott kérések egy kollekcióban mentésre kerültek és az *Agora_API_tesztek.js* állomány importálása után újra lefuttathatók.

4 API dokumentáció

4.1 Áttekintés

Ez a dokumentáció az Agora webshop által használt JSON alapú API használatát mutatja be. A kéréseket a fejlesztési fázisban egy a 3080-as porton futó Node.js webszerver fogadta, így a példákban a végpontok „http://localhost:3080/” kezdetűek.

4.2 Authentikáció

Az erőforrások közzétételéhez és eléréséhez a legtöbb esetben egy hozzáférési azonosítóra (token) van szükség. A sikeres bejelentkezést követően egy JWT-t tartalmazó objektum érkezik válaszként, mellyel a további kérések azonosíthatók.

A bejelentkezés a következő végpontra küldött kéréssel lehetséges:

POST http://localhost:3080/user/login

Content-Type: application/json
Accept: application/json
Accept-Charset: utf-8

```
{
    "userEmail":"nagyerzsi@mail.com",
    "userPassword":"12345"
}
```

Amire válasz:

Status: 200 OK
content-type: application/json; charset=utf-8


```
{
  "status": 1,
  "data": [
    {
      "member_id": 26,
      "first_name": "Erzsébet",
      "last_name": "Nagy",
      "email": "nagyerzsi@mail.com",
      "password": "827ccb0eea8a706c4c34a16891f84e7b",
      "phone": null,
      "about": null,
      "profile_picture_link": null,
      "header_picture_link": null,
      "registered_at": "0000-00-00 00:00:00",
```

```
      "last_login": null,
      "is_vendor": 0,
      "is_admin": 0
    }
  ],
  "token":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJkYXRhIjpbeyJtZW1iZXJfaWQiOjI2L
CJmaXJzdF9uYW1lIjoiRXJ6c8OpYmV0IiwibGFzdF9uYW1lIjoiTmFneSIsImVtYWlsIjoi
bmFneWVyenNpQG1haWwuY29tIiwicGFzc3dvcmQiOiI4MjdjY2IwZWVhOGE3MDZjNGMzNGE
xNjg5MWY4NGU3YiIsInBob25lIjpudWxsLCJhYm91dCI6bnVsbCwicHJvZmlsZV9waWN0dX
JlX2xpbmsiOm51bGwsImhlYWRlcl9waWN0dXJlX2xpbmsiOm51bGwsInJlZ2lzdGVyZWRfY
XQiOiIwMDAwLTAwLTAwIDAwOjAwOjAwIiwibGFzdF9sb2dpbiI6bnVsbCwiaXNfdmVuZG9y
IjowLCJpc19hZG1pbiI6MH1dLCJpYXQiOjE2NDc4ODU4Mjl9.0iuoN33qYUmgM-
_xIBfUZKMbeD7LoOx5BMbdtobfRvM"
}
```


A sikeres azonosítást követően a HTTP fejléc „Authorization” mezőjébe helyezzük el a kulcsot. A kérések a következő formátumban várják majd:

Authorization: Bearer {{token}}

4.3 Erőforrások

Az API fő célja, hogy különböző erőforrásokat tegyen elérhetővé a frontend számára. A következőkben ismertetésre kerül pontosan, hogyan szükséges egy kérést felépíteni és milyen válasz várható arra.

4.3.1 Felhasználók

4.3.1.1 Felhasználó lekérdezése

Egy felhasználó adatait lekérdező kérés így néz ki:

```
GET http://localhost:3080/api/user/{{userId}}
```

A válaszobjektum a következőket tartalmazza:


```
{
  "userId": 21,
  "firstName": "Erzsébet",
  "lastName": "Nagy",
  "about": "könnyebb tehát keresztül Nagyon inkább italért. eljön.
vagyok? tehát utolsó eddig a",
  "profileImgUrl": "assets/def-pfp2.png",
  "headerImgUrl": "assets/default_assets/def-bg3.png",
  "registeredAt": "2022-01-10T23:00:00.000Z",
```

```
  "companyName": "PeaceOfMind Csoport",
  "siteLocation": "Rápcakapi",
  "website": null,
  "takesCustomOrders": 0,
  "followers": 2,
  "following": 2,
  "email": "evanagy784@mail.com",
  "phone": "+36 10 767 3558",
  "isVendor": 1
}
```


Mező	**Típus**	**Leírás**
userId	number	A felhasználó egyedi azonosítója.
email	string	A felhasználó email címe.
phone	string	A felhasználó telefonszáma. Lehet null.
firstName	string	A felhasználó keresztneve.
lastName	string	A felhasználó vezeték neve.
about	string	A felhasználó rövid bemutatkozása. Lehet null.
profileImgUrl	string	A felhasználó profilképére mutató link.
headerImgUrl	string	A felhasználó borítóképére mutató link.
registredAt	date	A profil létrehozásának dátuma.
followers	number	A felhasználót követő fiókok száma.
following	number	A felhasználó által követett fiókok száma.
isVendor	bool	A felhasználó akkor eladó, ha igaz. A dupla vonal utáni mezők csak abban az estben léteznek, ha a felhasználó eladó.
companyName	string	Az eladó által képviselt cég neve.
siteLocation	string	Az eladó által képviselt cég, amennyiben létezik, telephelye.
website	string	Az eladó vagy az általa képviselt cég weboldalára mulató link.
takesCustomOrders	bool	Az eladó fogad egyedi rendeléseket vagy sem.

4.3.1.2 Felhasználó lekérdezése röviden

Szükség lehet csak a felhasználó adatainak egy részét röviden lekérdezni, ebben az esteben hívható a következő kérés is:

```
GET http://localhost:3080/api/user/{{userId}}/short
```

A válasz ekkor például így nézhet ki:

```
{
  "userId": 21,
  "firstName": "Erzsébet",
  "lastName": "Nagy",
  "about": "könnyebb tehát keresztül Nagyon inkább italért. eljön.
vagyok? tehát utolsó eddig a",
  "profileImgUrl": "assets/def-pfp2.png",
  "takesCustomOrders": 0
}
```

A mezők tartalma megegyezik a több adatot kiszolgáló kéréssel kapcsolatban leírtakkal.

4.3.1.3 Felhasználó szerkesztése

Regisztrált felhasználó adatait szerkeszteni a következő végpontra intézett kéréssel lehet:

```
PUT http://localhost:3080/api/user
```

```
Content-Type: application/json
Accept: application/json
Accept-Charset: utf-8
Authorization: Bearer {{token}}
```

```
{
  "userId": 21,
  "firstName": "Erzsébet",
  "lastName": "Nagy",
  "about": "könnyebb tehát keresztül Nagyon inkább italért. eljön.
vagyok? tehát utolsó eddiga",
  "profileImgUrl": "assets/def-pfp2.png",
  "headerImgUrl": "assets/default_assets/def-bg3.png",
  "companyName": "PeaceOfMind Csoport",
  "siteLocation": "Rápcakapi",
  "website": null,
  "takesCustomOrders": 0,
```

```
  "phone": "+36 10 767 3558",
  "isVendor":1
}
```

A JSON objektumnak kötelezően tartalmaznia kell a felhasználó egyedi azonosítóját, ez alapján választja ki, melyik fiók adatai módosulnak. Ha egy nem kötelező mező megadását kihagyja, az értéke null lesz. Fontos megjegyezni ahhoz, hogy ne szűnjön meg egy fiók eladói státusza, minden változtatáskor meg kell adni annak értékét. Egy eladó adatainak szerkesztésekor a táblázatban dupla vonal után felsorolt paraméterek is bevihetők. Abban az esetben, ha egy vásárlónál kerülnek megadásra, a kérés hiba nélkül lefut, de mivel a mezők nem léteznek változás nem fog történni.

Paraméter	**Típus**	**Kötelező?**	**Leírás**
userId	number	igen	A felhasználó azonosítója.
firstName	string	igen	A felhasználó keresztneve.
lastName	string	igen	A felhasználó vezeték neve.
about	string	nem	A felhasználó rövid bemutatkozása. Lehet null.
profileImgUrl	string	igen	A felhasználó profilképére mutató link.
headerImgUrl	string	igen	A felhasználó borítóképére mutató link.
phone	string	nem	A felhasználó telefonszáma.
isVendor	bool	nem	A felhasználó eladó vagy sem.
companyName	string	nem	Az eladó által képviselt cég neve.
siteLocation	string	nem	Az eladó által képviselt cég, amennyiben létezik, telephelye.
website	string	nem	Az eladó vagy az általa képviselt cég weboldalára mulató link.
takesCustomOrders	bool	nem	Az eladó fogad egyedi rendeléseket vagy sem.

A visszatérő objektum felépítése azonos egy felhasználó lekérdezésekor kapott struktúrával, a frissített adatokról ad visszajelzést.

Status: 200 OK


```
{
  "userId": 21,
  "firstName": "Erzsébet",
  "lastName": "Nagy",
  "about": "könnyebb tehát keresztül Nagyon inkább italért. eljön.
vagyok? tehát utolsó eddiga",
  "profileImgUrl": "assets/def-pfp2.png",
  "headerImgUrl": "assets/default_assets/def-bg3.png",
  "registeredAt": "2022-01-10T23:00:00.000Z",
  "companyName": "PeaceOfMind Csoport",
  "siteLocation": "Rápcakapi",
  "website": null,
  "takesCustomOrders": 0,
  "followers": 3,
  "following": 2,
  "email": "evanagy784@mail.com",
  "phone": "+36 10 767 3558",
  "isVendor": 1
}
```


4.3.2 Termékek

4.3.2.1 Termék lekérdezése

Egy terméket lekérdező kérés így néz ki:

```
GET http://localhost:3080/api/product/{{productId}}
```

A válasz objektum egy termék adatait tartalmazza. Az értékelések rájuk tett szavazatok alapján jelennek meg sorrendben, a legtöbb ponttal rendelkező az első.

Status: 200 OK
content-type: application/json; charset=utf-8


```
{
  "productId": 10,
  "name": " Ementáli Sajt",
  "price": 7400,
```

```
  "description": "darabig közé jog tudom szó áldja azon kopog, nem
vagyok ",
  "inventory": 16,
  "delivery": "Azonnal szállítható",
  "category": "Tejtermék",
  "rating": 2.75,
  "sellerId": 24,
  "discount": null,
  "isPublic": 1,
  "createdAt": "2022-01-03T23:00:00.000Z",
  "lastUpdatedAt": null,
  "publishedAt": "2022-01-03T23:00:00.000Z",
  "sellerFirstName": "Rozália",
  "sellerLastName": "Papp",
  "materials": [
    "tehéntej"
  ],
  "tags": [
    "Bio",
    "Vegán"
  ],
  "imgUrl": [
    "assets/product-pictures/cheese6.jpg",
    "assets/product-pictures/cheese3.jpg",
    "assets/product-pictures/cheese4.jpg"
  ],
  "reviews": [
    {
      "reviewId": 18,
      "userId": 20,
      "rating": 3,
      "title": "rendetlenséget! ajtót! ",
      "content": "szeret. hozzá jól ezzel tudom jól érdekében hiszem,
életforma.",
      "publishedAt": "2022-01-03T23:00:00.000Z",
      "userFirstName": "Ildikó",
      "userLastName": "Biró"
    },
    {
```

```
      "reviewId": 41,
      "userId": 7,
      "rating": 3,
      "title": "elég vagyok ",
      "content": "ember együtt a lehetőség szép bumról, lesz szép áldja
hiszem.",
      "publishedAt": "2022-01-03T23:00:00.000Z",
      "userFirstName": "Melinda",
      "userLastName": "László"
    }
  ]
}
```


Mező	**Típus**	**Leírás**
productId	number	A termék egyedi azonosítója.
name	string	A termék megnevezése.
price	number	A termék ára forintban.
description	string	A termék rövid leírása.
inventory	number	Jelenleg raktárban lévő termékek darabszáma. Értéke lehet null.
delivery	string	A termék elérhetőségét jelöli. Lehet "Azonnal szállítható" vagy "Megrendelésre készül".
category	string	A termék kategóriája.
rating	number	Értékelés, a vásárlói vélemények átlaga, tört szám 1 és 5 között.
sellerId	number	Az eladó egyedi azonosítója.
discount	number	Akció, százalékban értendő, 1 és 99 közötti szám. Értéke lehet null.
isPublic	bool	A termék láthatóságát jelöli.
createdAt	date	A termék létrehozásának dátuma.
lastUpdatedAt	date	A termék legutóbbi frissítésének dátuma. Ha nem lett frissítve, akkor null.
sellerFirstName	string	A termék eladójának keresztneve.
sellerLastName	string	A termék eladójának vezetékneve.
materials	string array	A termék anyagának/összetevőinek felsorolása.
tags	string array	A terméket leíró címkék felsorolása.
imgUrl	string array	A termékhez tartozó képekre mutató linkek.
review	object array	A termékről írt felhasználói vélemények.

4.3.2.2 Az összes termék lekérdezése

Minden publikusan közzé tett termék elérése a hozzáadás sorrendjében, így lehetséges:

```
GET http://localhost:3080/api/products
```

A válasz a termékeket röviden leíró objektumok sora:


```
[
  {
    "productId": 1,
    "name": "Arany nyaklánc",
    "price": 6990,
    "sellerId": 1,
    "discount": null,
    "isPublic": 1,
    "sellerFirstName": "Zita",
    "sellerLastName": "Boros",
    "imgUrl": "assets/item1.jpg"
  },
  {
    "productId": 2,
    "name": "Ásvány nyakék",
    "price": 4990,
    "sellerId": 2,
    "discount": null,
    "isPublic": 1,
    "sellerFirstName": "Aranka",
    "sellerLastName": "Németh",
    "imgUrl": "assets/item2.jpg"
  },
  {
    "productId": 3,
    "name": "Gyönygy medál",
    "price": 12990,
    "sellerId": 2,
    "discount": null,
    "isPublic": 1,
    "sellerFirstName": "Aranka",
    "sellerLastName": "Németh",
    "imgUrl": "assets/item4.jpg"
```

```
  }
]
```

Ahol egy objektum felépítése a következő:

Mező	**Típus**	**Leírás**
productId	number	A termék egyedi azonosítója.
name	string	A termék megnevezése.
price	number	A termék ára forintban.
sellerId	number	Az eladó egyedi azonosítója.
discount	number	Akció, százalékban értendő, 1 és 99 közötti szám. Értéke lehet null.
isPublic	bool	A termék láthatóságát jelöli.
imgUrl	string array	A termékhez tartozó index képre mutató link.

4.3.2.3 Egy eladó termékeinek lekérdezése

Ha egyetlen árus portékájára vagyunk kíváncsiak, azt a következő kéréssel érhetjük el:

```
GET http://localhost:3080/api/products/seller/{{userId}}
```

A válasz az összes termék lekérdezésekor kapott objektumokkal egyező felépítésű.

```
[
  {
    "productId": 5,
    "name": "Friss Ementáli Sajt",
    "price": 3610,
    "sellerId": 21,
    "discount": null,
    "sellerFirstName": "Erzsébet",
    "sellerLastName": "Nagy",
    "imgUrl": "assets/product-pictures/cheese1.jpg"
  },
  {
    "productId": 14,
    "name": "Product-02",
    "price": 1500,
    "sellerId": 21,
    "discount": null,
    "sellerFirstName": "Erzsébet",
```

```
    "sellerLastName": "Nagy",
    "imgUrl": "assets/default_assets/def-prod.png"
  },
  {
    "productId": 15,
    "name": "Teljes kiőrlésű kenyér",
    "price": 450,
    "sellerId": 21,
    "discount": null,
    "sellerFirstName": "Erzsébet",
    "sellerLastName": "Nagy",
    "imgUrl": "assets/default_assets/def-prod.png"
  }
]
```


4.3.2.4 Saját termékek lekérdezése

Egy eladó termékeinek lekérdezésétől annyiban tér el, hogy visszaadja a privát láthatóságú elemeket is. A válasz ugyanúgy a termékeket röviden leíró objektumok sora. A következő végponton érhető el:

```
GET http://localhost:3080/api/my-products/{{userId}}
```

4.3.2.5 Új termék hozzáadása

Új termék felviteléhez szükséges megadni egy megfelelő felépítésű JSON állományt a kérés testében, valamint az eladónak azonosítania kell magát a fejlécben elhelyezett kulccsal. Erre egy példa:

```
POST http://localhost:3080/api/product
```

Content-Type: application/json
Accept: application/json
Accept-Charset: utf-8
Authorization: Bearer {{token}}


```
{
  "name": "Teljes kiőrlésű kenyér",
  "price": 500,
  "discount": 20,
  "inventory": 23,
  "delivery": "Azonnal szállítható",
  "category": "Pékáru",
  "sellerId": 21,
```

```
  "materials": [ "liszt", "élesztő" ],
  "tags": [ "Kézzel készült", "Bio" ],
  "imgUrl": [ "assets/default_assets/def-prod.png" ],
  "description": "A legfinomabb friss, teljes kiőrlésű kenyér a
régióban.",
  "isPublic": true
}
```

Paraméter	Típus	Kötelező?	Leírás
name	string	igen	A termék megnevezése.
price	number	igen	A termék ára forintban.
discount	number	nem	A termékre vonatkozó akció százalékban. (1 és 99 közötti szám.)
inventory	number	nem (Ha, megrendelésre készül.)	Jelenleg raktárban lévő termékek darabszáma.
delivery	string	igen	A termék elérhetőségét jelöli. Lehet "Azonnal szállítható" vagy "Megrendelésre készül".
category	string	igen	A termék kategóriája.
sellerId	number	igen	Az eladó egyedi azonosítója.
materials	string array	nem	A termék anyagának/összetevőinek felsorolása.
tags	string array	nem	A terméket leíró címkék felsorolása.
imgUrl	string array	igen	A termékhez tartozó képekre mutató linkek.
description	string	nem	A termék rövid leírása.
isPublic	bool	igen	A termék láthatósága

A válaszobjektum felépítése azonos egy termék lekérdezése után visszatérővel, többek között kiderül belőle a beillesztett elem egyedi azonosítója.

```
{
  "productId": 43,
  "name": "Teljes kiőrlésű kenyér",
  "price": 500,
  "description": "A legfinomabb friss, teljes kiőrlésű kenyér a
régióban.",
  "inventory": 23,
  "delivery": "Azonnal szállítható",
  "category": "Pékáru",
  "rating": null,
  "sellerId": 21,
  "discount": 20,
  "isPublic": 1,
  "createdAt": "2022-03-29T17:43:26.000Z",
  "lastUpdatedAt": null,
  "publishedAt": "2022-03-29T17:43:26.000Z",
  "sellerFirstName": "Erzsébet",
  "sellerLastName": "Nagy",
  "materials": [
    "liszt",
    "élesztő"
  ],
  "tags": [
    "Kézzel készült",
    "Bio"
  ],
  "imgUrl": [
    "assets/default_assets/def-prod.png"
  ],
  "reviews": []
}
```

4.3.2.6 Termék törlése

Egy terméket az egyedi azonosítójával érhetünk el és törölhetünk.

```
DELETE http://localhost:3080/api/product/{{productId}}
```

Sikeres visszatérési értéke:

Status: 200 OK

```
{
  "message": "Item with ID 35 was deleted."
}
```

4.3.2.7 Termék szerkesztése

Szerkeszteni egy terméket a létrehozáskor leírttal azonos JSON struktúrával lehetséges, a következő végponton:

```
PUT http://localhost:3080/api/product/{{productId}}
```

A végpontban megadott azonosító számmal jelölt elemen fognak végbe menni a változtatások, így a kérés törzsében nem kell szerepelnie annak. A termék összes eleme frissítésre kerül, így szükséges megadni azokat az adatokat is, melyeket nem kíván megváltoztatni. A visszatérő JSON a hozzáadáshoz hasonlóan egy termék objektum, mely ez esteben az új adatokat tartalmazza.

4.3.3 Kívánság listák

4.3.3.1 Kívánság lista lekérdezése

Minden fiókhoz egy kívánság lista tartozik, ezért a felhasználó egyedi azonosítójával kérhetők le a listáján szereplő termékek és adataik röviden.

```
GET http://localhost:3080/api/wish-list/{{userId}}
```

A válaszobjektum például a következő lehet:

Status: 200 OK
content-type: application/json; charset=utf-8

```
[
  {
    "productId": 7,
    "name": "Füstölt Edami Sajt",
    "price": 6240,
    "sellerId": 25,
    "discount": null,
```

```
    "isPublic": 1,
    "sellerFirstName": "Béla",
    "sellerLastName": "Horváth",
    "imgUrl": "assets/product-pictures/cheese2.jpg",
    "wishListId": 2,
    "addedAt": "2022-02-19T18:53:15.000Z"
  },
  {
    "productId": 8,
    "name": "Finom Brie Sajt",
    "price": 4010,
    "sellerId": 22,
    "discount": null,
    "isPublic": 1,
    "sellerFirstName": "Rozália",
    "sellerLastName": "Jakab",
    "imgUrl": "assets/product-pictures/cheese3.jpg",
    "wishListId": 1,
    "addedAt": "2022-02-19T16:53:17.000Z"
  }
]
```


Ahol egy termék objektum felépítése:

Mező	**Típus**	**Leírás**
productId	number	A termék egyedi azonosítója.
name	string	A termék megnevezése.
price	number	A termék ára forintban.
sellerId	number	A termék eladójának egyedi azonosítója.
discount	number	Akció, százalékban értendő, 1 és 99 közötti szám. Értéke lehet null.
isPublic	bool	A termék láthatóságát jelöli.
sellerFirstName	string	A termék eladójának keresztneve.
sellerLastName	string	A termék eladójának vezetékneve.
imgUrl	string	A termékhez tartozó képre mutató link.
wishListId	number	A kívánság lista elem egyedi azonosítója.
addedAt	date	A termék kívánság listához való hozzáadásának ideje.

4.3.3.2 Termék hozzáadása a kívánság listához

A termék és a felhasználó egyedi azonosítóját tartalmazó JSON tartalommal kell kérést küldeni, új elem listához adásához, a következő végpontra:

```
POST http://localhost:3080/api/wish-list
```

Content-Type: application/json

Accept: application/json

Accept-Charset: utf-8

Authorization: Bearer {{token}}

```
{
 "productId": 5,
 "userId": 21
}
```

Ha az új elem sikeresen hozzá lett adva a listához, akkor válaszként visszatér annak azonosítója.

Status: 201 Created

```
{
  "wishListId": 10
}
```

4.3.3.3 Termék törlése a kívánság listáról

Egy termék eltávolítása a kívánság listáról az elem egyedi azonosítójával küldött DELET kéréssel történik:

```
DELETE http://localhost:3080/api/wish-list/{{wishListId}}
```

Sikeres törlés esetén a válasz:

Status: 200 OK

```
{
  "message": "Item with ID 10 was deleted."
}
```

4.3.4 Értékelések

4.3.4.1 Értékelés lekérdezése

Egy felhasználói vélemény lekérdezése:

```
GET http://localhost:3080/api/review/{{reviewId}}
```

A válasz objektum egy értékelés adatait tartalmazza.

Status: 200 OK
content-type: application/json; charset=utf-8


```
[
  {
    "reviewId": 7,
    "productId": 3,
    "userId": 11,
    "rating": 5,
    "points": 8,
    "title": "Egyszerűen tökéletes!!!",
    "content": "Az unokámnak vettem szülinapjára és imádja!!!!
Köszönöm!!!!!",
    "publishedAt": "2022-01-10T23:00:00.000Z",
    "userFirstName": "Veronika",
    "userLastName": "Bakos"
  }
]
```


Ha bejelentkezett felhasználóként szeretnénk megkapni az értékelést, akkor a következő végpontra intézzük a kérést:

```
GET http://localhost:3080/api/review/{{reviewId}}/auth/{{userId}}
```

Ebben az estben a visszatérő JSON struktúra kiegészül a „myVote” mezővel, mely a felhasználó személyes szavazatát tartalmazza.


```
{
  "reviewId": 52,
  "productId": 14,
  "memberId": 26,
  "rating": 5,
  "points": 2,
  "title": "Szuper termék",
```

```
  "content": "Szuper áron :)))))",
  "publishedAt": "2022-03-22T20:08:30.000Z",
  "userFirstName": "Erzsébet",
  "userLastName": "Nagy",
  "myVote": "up"
}
```


Mező	Típus	Leírás
reviewId	number	Az értékelés egyedi azonosítója.
productId	number	Azon termék egyedi azonosítója, melyre az értékelés vonatkozik.
userId	number	A véleményt író felhasználó egyedi azonosítója.
rating	number	A véleményező által adott értékelés. (1-5 csillag)
points	number	Az értékelésre érkezett szavazatok összege, egy egész szám, lehet negatív és pozitív is.
title	string	Az értékelés címe
content	string	Az értékelés törzse, a kifejtett véleményt tartalmazza.
publishedAt	date	A közzététel dátuma.
userFirstName	string	A vélemény írójának keresztneve.
userLastName	string	A vélemény írójának vezetékneve.
myVote	string?	A felhasználó saját szavazata. Értéke lehet null, „up” vagy „down.”

4.3.4.2 Értékelés hozzáadása

Egy értékelés hozzáadása a következőképp történik:

```
POST http://localhost:3080/api/review
```

Content-Type: application/json
Accept: application/json
Accept-Charset: utf-8
Authorization: Bearer {{token}}

```
{
    "productId": 18,
    "userId": 21,
    "rating": 5,
    "title": "Egyszerűen tökéletes!!!",
    "content": "Az unokámnak vettem szülinapjára és imádja!!!! Köszönöm!!!!!"
}
```


Paraméter	Típus	Kötelező?	Leírás
productId	number	igen	A véleményezni kívánt termék egyedi azonosítója.
userId	number	igen	A felhasználó egyedi azonosítója.
rating	number	igen	Numerikus értékelés egy és öt között.
title	string	igen	Az értékelés címe.
content	string	igen	Az értékelés törzse.

A sikeres közzétételét után a következőhöz hasonló válasz várható, mely felépítése azonos az értékelés lekérésénél leírtakkal és tájékoztat a beillesztett elem adatairól.

Status: 201 Created


```
{
  "reviewId": 57,
  "productId": 18,
  "memberId": 21,
  "rating": 5,
  "points": 0,
  "title": "Egyszerűen tökéletes!!!",
  "content": "Az unokámnak vettem szülinapjára és imádja!!!! Köszönöm!!!!!",
  "publishedAt": "2022-03-30T08:14:25.000Z",
  "userFirstName": "Erzsébet",
  "userLastName": "Nagy"
}
```

4.3.4.3 Szavazat leadása értékelésre

Minden felhasználó értékelésenként egy szavazattal rendelkezik, amivel a vélemény pontjait megnövelheti vagy csökkentheti, evvel mutatva, mennyire volt számára hasznos az adott érékelés. Ha a felhasználó már adott le szavazatot, a régi elem automatikusan törlésre kerül és helyette egy új lesz beszúrva.

POST http://localhost:3080/api/review-vote

```
{
    "productId": 18,
    "reviewId":52,
    "userId": 21,
    "vote": "up"
}
```

Paraméter	**Típus**	**Kötelező?**	**Leírás**
productId	number	igen	A termék egyedi azonosítója, melyen az értékelés szerepel.
reviewId	number	igen	Az értékelés egyedi azonosítója, melyre a szavazat leadásra kerül.
userId	number	igen	A szavazatot leadó felhasználó egyedi azonosítója.
vote	string	igen	A felhasználó szavazata. Értéke lehet „up” vagy „down.”

Sikeres szavazat leadás után a válaszban megérkezik a beszúrt elem egyedi azonosítója.

Status: 201 Created

```
{
  "reviewVoteId": 43
}
```

4.3.4.4 Szavazat törlése

Az érintett felhasználó és értékelés egyedi azonosítójával, a következő végpontra intézett kérés töröli a szavazatot.

DELETE http://localhost:3080/api/review-vote/{{reviewId}}/{{userId}}

Válasz sikeres futtatás esetén:

Status: 200 OK


```
{
  "message": "Item with ID 43 was deleted."
}
```


4.3.5 Kosár és rendelés

4.3.5.1 Kosár lekérdezése

Mikor egy felhasználó terméket helyez a kosarába létrejön egy „Not Sent” státuszú kosár elem. Egy vásárlóhoz csak egy kosár tartozik, ezt le lehet kérdezni a következő végponton:

```
GET http://localhost:3080/api/cart/{{userId}}
```

Válaszként például ez az objektum érkezhet:

Status: 200 OK


```
{
  "cartId": 16,
  "userId": 21,
  "shippingId": null,
  "sumPrice": 10600,
  "status": "Not Sent",
  "products": [
    {
      "productId": 5,
      "name": "Friss Ementáli Sajt",
      "price": 3610,
      "sellerId": 21,
      "discount": null,
      "isPublic": 1,
      "sellerFirstName": "Erzsébet",
      "sellerLastName": "Nagy",
      "imgUrl": "assets/product-pictures/cheese1.jpg",
      "amount": 1,
      "status": "In Cart",
      "cartProductId": 50
    },
    {
      "productId": 1,
```

```
      "name": "Arany nyaklánc",
      "price": 6990,
      "sellerId": 1,
      "discount": null,
      "isPublic": 1,
      "sellerFirstName": "Zita",
      "sellerLastName": "Boros",
      "imgUrl": "assets/item1.jpg",
      "amount": 1,
      "status": "In Cart",
      "cartProductId": 57
    }
  ]
}
```


A kosárak és rendelések ugyanabban a táblában vannak eltárolva, így felépítésük azonos. Egy kosár rendeléssé változik, ha a felhasználó szállítási címet rendel hozzá és véglegesíti azt. Párhuzamosan több rendelés is lehet függőben.

Egy kosár/rendelés a következő mezőket tartalmazza:

Mező	**Típus**	**Leírás**
cartId	number	A kosár/rendelés egyedi azonosítója.
userId	number	A rendelés/kosár tulajdonosának egyedi azonosítója.
shippingId	number	A szállítási cím egyedi azoonosítója.
sumPrice	number	Az összes termék ára.
status	string	A kosár/rendelés állapota.
products	array	A korárban/rendelésben lévő termékeket tartalmazza.

A „products” tömb a rendelt termékek felsorolása, egy ilyen elem felépítése:

Mező	**Típus**	**Leírás**
productId	number	A termék egyedi azonosítója.
name	string	A termék megnevezése.
price	number	A termék ára forintban.
description	string	A termék rövid leírása.
sellerId	number	Az eladó egyedi azonosítója.
discount	number	Akció, százalékban értendő, 1 és 99 közötti szám. Értéke lehet null.
sellerFirstName	string	A termék eladójának keresztneve.
sellerLastName	string	A termék eladójának vezetékneve.
imgUrl	string array	A termékhez tartozó képekre mutató linkek.
amount	number	A megrendelt termék darabszáma.
status	string	A termék szállítási állapota.
cartProductId	number	A rendelt termék elem egyedi azonosítója.

4.3.5.2 Rendelések lekérdezése

Egy vásárló kimenő rendelései itt érhetők el:

```
GET http://localhost:3080/api/my-orders/{{userId}}
```

Egy eladó bejövő rendelései pedig itt:

```
GET http://localhost:3080/api/order/{{userId}}
```

A bejövő rendelések között a felhasználó csak a tőle rendelt termékeket láthatja akkor is, ha ugyanazon rendelésen belül több eladótól származó árucikkek is szerepelnek.

A válasz a kosár lekérdezéséhez képest kiegészül egy újabb „address” nevű mezővel, ami leírja a szállítási cím részleteit.

4.3.6 Lehetséges hibák

Erőforrások lekérdezése vagy feltöltése esetén belefuthat a következő hibákba:

HTTP válaszkód	**Üzenet**	**Leírás**
503 Service Unavailable	Connection refused by database server.	Az adatbázishoz való csatlakozás sikertelen.
401 Unauthorized	Authentication failed	Az azonosítás sikertelen volt.
401 Unauthorized	No authentication token provided	Az HTTP fejléc azonosításhoz használt „Authorization” mezője üres.
401 Unauthorized	Token prefix incorrect or missing	Az azonosító nem a megfelelő „Bearer ” prefixummal kezdődik.
400 Bad Request	ID must be a number	Az azonosítót igénylő kéréseknél, ha nem lehet egész számmá konvertálni a megadott értéket érkezik ez a hibaüzenet.
400 Bad Request	Required fields were left empty.	Egy vagy több kötelező paraméter nem lett megadva.

5 Frontend

5.1 Keretrendszer és könyvtárak

A frontend kialakítása Angularban történt, ez egy TypeScript alapú fejlesztési platform, ami komponens alapú keretrendszert biztosít webes alkalmazás fejlesztéshez és könyvtárakat útválasztáshoz, űrlapkezeléshez, kliens-szerver kommunikációhoz.

A webalkalmazás kinézetének személyre szabásához a Bootstrap SCSS könyvtárát használtuk, mely lehetővé teszi a mobil első, reszponzív felhasználói felület létrehozását. Emellett a Font Awesome csomag ikonjai lettek implementálva az Angular könyvtáruk segítségével.

5.2 Komponensek

Az Angular alkalmazások építőelemei a komponensek, melyek egy TypeScript osztályból, egy HTML sablonból és CSS stílusokból állnak össze. Előnyük, hogy modulárisak, újra felhasználhatók, letisztultabbá, érthetőbbé teszik a kódot. Az Angular CLI segít, hogy egy parancs kiadásával létrejöjjenek a komponenst felépítő elemek egy számukra létrehozott mappában, valamint felvételre kerüljön az új komponens az *app.module.ts* fájl deklarációi közé is.

A projektben sokszor megjelenő felugró ablak egy egyszerű komponens. A hozzá tartozó TypeScript osztály a következőképpen néz ki:

```
@Component({
  selector: 'modal-window',
  templateUrl: './modal.component.html',
  styleUrls: ['./modal.component.scss']
})
export class ModalComponent implements OnInit {
  @Input() modalVisible = false;
  @Input() allowClose = true;
  @Input() title = "";
  @Output() modalVisibleChange = new EventEmitter<boolean>();
  toggleModal(){
    if(this.allowClose){
      this.modalVisible = !this.modalVisible;
      this.modalVisibleChange.emit(value);
    }
  }
  ngOnInit(): void {}
}
```

A felugró ablak komponenshez tartozó TypeScript osztály

Rögtön az osztály előtt szerepel a `@Component` dekorátor, melyen keresztül meghatározásra kerül, milyen néven érhető el a komponens a sablonból. Ebben az esetben a dekorátor paramétereit megadó objektumban a „selector” értéke „modal-window”, így a „`<modal-window></modal-window>`” HTML tag használatával illeszthetjük be a komponenst a kódba. A dekorátor további funkciója, hogy megjelölje, mely hivatkozások mutatnak a hozzá tartozó sablonra, illetve stílusokra. Ez a példában a TS állománnyal a fájlstruktúrában egy szinten létező *modal.component.html* és *modal.component.scss*. Az osztályon belül deklarálva vannak változók és tartalmazza az ablak megnyitását és bezárását kezelő logikát is.

Egy komponens sablonja a kinézetét leíró HTML kód. Az MVC szemléletben a nézet szerepét töltené be. A nézetek hierarchikusan viszonyulnak egymáshoz, hiszen egy gazda nézetbe beágyazható egy másik. A sablonok a megszokott HTML-n felül támogatják az Angular speciális sablon szintaxisát. Ez teszi lehetővé, hogy a nézet az alkalmazás logikájának megfelelő módon, dinamikusan változzon. Többek között az adatkötés (*data binding*), csővezetékek (*pipes*) és a direktívák működését biztosítja.

```
<div scrollBlock [@openModal]="modalVisible" *ngIf="modalVisible">
    <div class="modal-background w-100 frosted-glass" (click)="toggleModal()">
        <div (click)="$event.stopPropagation()" class="modal-body position-
fixed top-50 start-50 translate-middle shadow rounded">
            <div class="d-grid gap-1 d-md-flex justify-content-md-between">
                <p class="fw-bold title p-0 m-0">
                    {{title}}
                </p>
                <button *ngIf="allowClose" (click)="toggleModal()" class="btn
m-0 p-0">
                    <fa-icon  [icon]="faClose"></fa-icon>
                </button>
            </div>
            <ng-content class="postition-absolute"></ng-content>
        </div>
    </div>
</div>
```

A komponens sablonja

A példában a felugró ablak a címét egyirányú adatkötéssel szerzi meg. A TS osztályban `@Input` dekorátorral ellátott `title` változó deklarálása jelzi a komponensnek, hogy azonos néven megadható egy tulajdonságot a szülő nézetben. A tulajdonság kötés a szülő elem felől a gyermek elembe tart, jelölése szögletes zárójellel történetik. Jelen esetben a cím tulajdonság értéke „Figyelmeztetés” ezért ez fog megjelenni az ablak fejlécében. Ugyanilyen módon közölhető a komponenssel az `allowClose` tulajdonságon keresztül, hogy az ablak a felhasználó által bezárható legyen-e.

```
<modal-window [(modalVisible)]="alertOpen" [title]="'Figyelmeztetés'">
    <div class="warning">
        <div class="text-center p-3 border-top mt-2">
            <p class="large">Biztosan törölni szeretné a(z) <span class="fw-
bold">{{selectedProduct.name}}</span> megnevezésű terméket?</p>
            <p class="mb-0 pb-0">Ezt a műveletet nem lehet visszavonni.</p>
        </div>
        <div class="d-flex justify-content-between border-top pt-2">
            <a href="" (click)="$event.preventDefault(); alertOpen = false;
moreOptionsOpen = false" class="btn btn-secondary me-1">Mégse</a>
            <a href="" (click)="deleteProduct($event)" class="btn btn-danger
mx-1">Törlés</a>
        </div>
    </div>
</modal-window>
```

A komponens beágyazva egy másik nézetbe

A felugró ablak jobb sarkában levő gombra kattintáskor egy esemény kerül kibocsátásra, ehhez egy függvényt rendelni, mely minden kattintáskor lefut, nevezünk esemény kötésnek. Jelölése gömbölyű zárójelekkel történik. A példában a `toggleModal()` függvény kerül meghívásra, mely megváltoztatja az ablak láthatóságát, amennyiben az megengedett.

Az ablak nyitott vagy zárt állapotát tároló logikai változó esetében két irányú adatkötés valósul meg, a szülő és gyermek komponensek oda-vissza kommunikálnak. Egy tulajdonság kötésen keresztül a szülőben végbemenő változás értesíti a gyermek elemet, míg ő egy esemény kötéssel jelezhet a szülőnek. Mivel mind a két típusú kötés egyszerre van jelen jelölése a gömbölyű és szögletes zárójelek egyszerre való használata. Ilyen esetben az esemény kibocsátása egy `@Output` dekorátorral deklarált `EvetEmitter`-en keresztül megy végbe, melynek neve megegyezik az adatkötni kívánt változó nevével csak hozzá van fűzve a `Change` szócska. A példában ez a `modalVisibleChange`, ami az X-re kattintáskor sugároz eseményt, így frissítve az ablak állapotát.

Ez a komponens, habár csővezetékeket nem, direktívákat használ. Az `*ngIf` egy strukturális direktíva, mely egy HTML elem (és a hierarchiában alatta szereplő elemek) láthatóságát egy logikai változóhoz köti. A felugró ablak nyitott vagy zárt állapotát kezeli a

példában. Emellett a `scrollBlock` nevű egyszerű saját direktíva is használatban van a komponensen, mely megakadályozza a görgő használatát, mikor az ablak látható a képernyőn.

A felugró ablak komponenst leginkább flexibilissé és újra felhasználhatóvá a tartalom kivetítése teszi. A *content projection* annyit tesz, hogy egy komponens belsejébe kerül beillesztésre vagy kivetítésre tartalom. Az `<ng-content>` HTML tag teszi ezt lehetővé, helyfoglalóként (*slot*) szolgál a később a *selector* tagpár között megadott tartalomnak.

Egy másik többször megjelenő komponens a lenyíló panel, mely működésében rendkívül hasonló, két irányú adatkötést használ a nyitott csukott állapot átviteléhez és tartalom kivetítést, hogy egy nézetet jelenítsen meg önmagában.

A webalkalmazásban leggyakrabban felbukkanó komponens a termék kártya, ami a sablon szintaxis nyújtotta további funkciókat is kihasznál. Ebben az esetben szükséges volt úgy kialakítani a kártyát, hogy az több helyzetben is a várt módon viselkedjen. Először a nyitó oldalon kell megjelenniük, négynek egy sorban, majd a termékek részleteit bemutató oldalon vízszintesen görgethető formában, végül pedig a keresések eredményeként egy keskenyebb helyen. Emellett figyelembe kellett venni, hogy az említett esetekben nem csupán egy, hanem termékek csoportja kerül megjelenítésre. Ezen okokból került kialakításra a `product-list` és `product-card` komponens.

A `product-list` komponensnek három tulajdonság kötéssel átvitt bemeneti paramétere van. A `products` névre hallgató egy termék objektumokból álló tömböt vár. A `wide` logikai változó határozza meg, hogy szélesebb vagy keskenyebb formában jelenjenek meg a kártyák, alapértelmezetten az értéke igaz. A `sideScrollable` tulajdonság a vízszintes görgetést teszi lehetővé a saját `appHorizontalScroll` direktívával, alapértelmezetten hamis. A komponens fő funkciója, hogy a `*ngFor` strukturális direktívával végig iteráljon a bemenetként kapott tömbön, minden alkalommal a `product-card` komponenst beágyazott nézetként meghívva.

A kártya megkapja a szélességet és vízszintes görgetést engedélyező változókat, valamint egy termék objektumot és ezek ismeretében formázza magát. CSS osztályok felételes alkalmazása két féle szintaxissal is lehetséges. A termék kártyában felfedezhető példa minkét esetre.

`[ngClass]="{'card-product':wide, 'card-narrow':!wide}"`

Itt az elem `ngClass` tulajdonsága, ahogy a szögletes zárójelek mutatják, tulajdonság kötéssel kapja meg az utána megadott objektumot. A paraméterezéskor karakterláncként egy osztályt szükséges megadni és hozzá egy logikai változót, mely jelöli, hogy éppen alkalmazni kell-e az adott stílust. Ha a termék kártya széles a `card-product` osztály, ha keskeny a `card-narrow` osztály stílusai formázzák a kártyát.

`[class.fit-text]="product.name.length> 20"`

Ebben az esetben, a tulajdonságban a pont után szerepel a stílusosztály, az értéke a logikai változó. A példában, ha a termék neve 20 karakternél hosszabb, akkor a `fit-text` osztály érvénybe lép és kisebbre veszi a betűméretet.

`{{product.price | price}}`

A termék kártya komponensben használtban van egy az ár formattálását végző csővezeték is. A szöveg interpoláció dupla kapcsos zárójelei között szerepel a termék objektum ár tulajdonsága, de megjelenítés elött keresztülmegy a `price` csővezetéken, melynek jelölésére az álló vonal vagy *pipe* karakter szolgál.

```
@Pipe({name: 'price'})
export class PricePipe implements PipeTransform {
  transform(value: number) {
    let price:string = Math.round(value).toString();
    price = price.replace(/(\d)(?=(\d\d\d)+(?!\d))/g, '$& ');
    return price + " Ft";
  }
}
```

Az árat átalakító csővezeték

A csővezetékek valójában `@Pipe` dekorátorral ellátott TS osztályok, bennük egyetlen `transform` függvénnyel. A függvény első paramétere a csővezeték előtti érték, mely átalakításra került, ezen felül megadható még egy paraméter is. Egy *pipe* előnye, hogy egyszerűen átalakítható egy érték a megjelenítéskor várt formátumba, például egy dátum esetében. A leggyakoribb helyzetek megoldására léteznek előre elkészített csővezetékek az Angular könyvtáraiban. A Magyarországon megszokott ár formátumhoz és a „Ft” utótag

hozzáadásához, azonban egy sajátot kellett írni, mely egy reguláris kifejezéssel a megfelelő alakba helyezi át a háttérben számként tárolt értéket.

Használtban van még több saját készítésű csővezeték is az alkalmazásban. A `trim` megnevezésű *pipe* feladata, hogy a túl hosszúra nyúló szöveget adott karakterszám után levágja és három pontot helyezzen mögé. A termékről írt értékelések megjelenítésekor van igénybe véve, itt a felhasználók akár hosszabb szövegben taglalhatják véleményüket, azonban a kis kártyákra kevesebb karakter fér. Alapértelmezetten 80 karakter után történik a vágás, de csővezetéket követően egy kettősponttal ez a paraméter megváltoztatható. Ez a gyakorlatban a következőképpen fest:

```
{{review.content|trim: 90}}
```

Egy másik csővezeték a kategóriák és címkék ikonokhoz rendeléséért felel. Valójában csak egy nagy *switch* szerkezet, ami kikeresi a tárolt nevekhez tartozó ikonok megnevezését, azonban a fejlesztés közben nagy könnyebbséget jelent. A következő sorral beilleszthető a megfelelő kategória ikonja:

```
<fa-icon [icon]="[iconPrefix, (product.category|icon:'category')]"></fa-icon>
```

A Font Awesome által készített komponensben az `icon` adatkötött tulajdonságnak kell megadni egy elő- és egy utótagot. Az előtag konstans, míg az utótagot a csővezeték határozza meg a dinamikus tartalomtól függően.

5.3 Útválasztás

Az Angular *router* használata az *app-routing.module.ts* állományon keresztül lett megvalósítva. Ebben a fájlban a beimportált komponensek kerülnek összekötésre egy-egy útvonallal. Az útvonalak felsorolása a statikus, legkevésbé általános bejegyzésekkel kezdődik és a helyettesitő útvonallal ér véget. A kettő csillaggal jelölt helyettesítő útvonal elkap bármilyen az URL-ben megadott értéket és egy helyre, például egy 404-es hibáról tájékoztató oldalra irányít.

Az útvonalak tartalmazhatnak kettősponttal kezdődő változókat is. Mikor egy komponens betöltése egy ilyen útvonalon keresztül történik, az ActivatedRoute szerviz osztály beinjektálásával megállapítható a változó értéke. Ez a funkció sok esetben lehet hasznos, például a profile-page komponensben, aminek feladata, hogy bármelyik felhasználó adatait a képernyőre vetítse. Egyértelműen nem hozható létre statikus, kézzel bevitt útvonal minden fiókhoz, ezért más megoldásra van szükség. Az útvonalban átadásra kerül egy a felhasználót

egyértelműen azonosító változó, a következő a komponens betöltésekor lefutó sor ezt kívánja megszerezni.

```
this.id = this.route.snapshot.paramMap.get('id');
```

A route ebben az estben a beinjektált ActivatedRoute osztály példánya, amin át lekérdezésre kerül az egyedi azonosító. Az ID ismeretében már be lehet tölteni a keresett felhasználó adatait.

A sablonban egy elem `routerLink` tulajdonságának megadásával beállítható, hogy ha a felhasználó rákattint, mely útvonalra navigáljon. A `queryParams` tulajdonsággal pedig az URL-ben kettőskereszttel jelölt lekérdezési paraméterek adhatók meg. A weboldalon például a navigációs sávban egy kategória kiválasztásakor, így történik az átirányítás a megadott kategóriába tartozó termékeket felsoroló oldalra.

Kódból a Route megnevezésű beinjektálható szerviz osztály navigate függvényével lehet egy másik útvonalra átirányítani.

5.4 Űrlapkezelés

Az űrlapokon keresztül történő adatbevitel az alkalmazásban központi helyet foglal el, hiszen hiányában nem lehetne regisztrálni, bejelentkezni, új termékeket felvenni, vásárolni. Az Angular két megközelítést is biztosít az űrlapok kezeléséhez, a reaktív és a sablon vezérelt szemléletet. A weboldal a reaktívat részesíti előnyben, mivel modell vezérelt és ideális a nagyobb és változó mennyiségű adatok bekérésére.

Az új termék hozzáadása funkció felépítése egy űrlapcsoport példány létrehozásával kezdődött. Egy űrlapcsoport (*form group*) legfőbb feladata összegyűjteni a benne szereplő elemek adatait, hogy azokat egy objektumban vissza tudja adni. Az űrlaptömbök (*form array*) ehhez hasonlóak, de elemeik futás közben hozzáadhatók, illetve törölhetők. A formBuilder injektálható osztály segítségével az ezeket felépítő űrlapvezérlő példányok rövidebb szintaxissal legenerálhatók.

```
productForm = this.fb.group({
    name: ['', Validators.required],
    price: [null, [Validators.required, Validators.min(0)]],
    discount: [null, [Validators.max(99), Validators.min(1)]],
    inventory: [null],
    delivery: ['', Validators.required],
    category: ['', Validators.required],
    tags: this.fb.array([]),
    materials: this.fb.array([]),
    description: ['']
  });
```

A termékek felvételéhez készült űrlapcsoport

A `productForm` nevű csoport megalkotása, ezen a módon történt az `fb` néven példányosított `formBuilder`-el. Minden felvett elemhez tartozik egy alapértelmezett érték a szögletes zárójelek között elsőként megadva. A vezérlők között szerepel két űrlaptömb is, ezek alapértelmezetten üresek, de lehetőség nyílik őket dinamikusan feltölteni további űrlapelemekkel.

Az Angular az űrlapvalidációt is leegyszerűsíti. Az alapértelmezett érték után megadható egy függvény a Validators osztályból, ami például ellenőrzi, hogy a kötelező mezők valóban ki lettek-e töltve és a megadott értékek az elvárt tartományon belülre esnek.

A modell és a nézet társítása a sablonban szereplő űrlap elem formGroup tulajdonságának adatkötésével valósul meg. A beviteli elemeket a formControlName megnevezésű tulajdonságuk azonosítja, ennek értéke egyezik meg a modellben felsorolt elemekkel.

Az űrlaptömbben szereplő beviteli mezőket adat kötni összetettebb, mert előre nem tudni hány elem kerül kitöltésre, azok nem helyezhetőek le statikusan. Egy termék címkéinek megadása pontosan ilyen eset, lehetséges, hogy nem kerül megadásra adat, de az is előfordulhat, hogy több beviteli mezőre van szükség. Ahhoz, hogy kezelni lehessen, először el kell érni az űrlap modellben létrehozott tömböt, ennek legkönnyebb módja egy *get* függvény létrehozása. Ezután a felhasználónak hozzá kell tudni adnia egy új beviteli mezőt, illetve, ha meggondolná magát a törlésre is lehetőséget kell adni. A címkék esetében ez, így lett megoldva:

```
get tags() {
  return this.productForm.get('tags') as FormArray;
}

addTag() {
  this.tags.push(this.fb.control(''));
}

removeTag(id: any, $event: any) {
  $event.preventDefault();
  this.tags.removeAt(id);
}
```

A címkeket kezelő űrlaptömb használata

Az űrlapelemek dinamikusan a tömbön végig iteráló *ngFor strukturális direktíva segítségével kerülnek megjelenítésre. A modellhez kapcsolásuk a tömbben elfoglalt sorszámuk alapján történik.

Mikor a felhasználó kitöltötte az űrlapot, annak valid tulajdonságával könnyen ellenőrizhető, hogy megfelelő adatok kerültek-e beütésre. Ha ez sikeresen megtörtént a value tulajdonságon keresztül egyszerre elérhetők az űrlap adatai, így nincs szükség azokat egyesével begyűjteni. Az űrlapmodell megfelelő típusúra konvertálása után az adatok rögtön tovább küldhetők a backend-nek.

5.5 Kliens-szerver kommunikáció

A webalkalmazás HTTP protokollon keresztül kommunikál a backend-el, hogy adatokat kérjen le és töltsön fel. Az Angular a HttpClient nevű szerviz osztályát ajánlja ennek megkönnyítésére.

Minden modell osztályhoz létre lett hozva egy szerviz osztály, mely a hozzá tartozó kommunikációt kezeli. Az @Injectable dekorátornak köszönhetően használatukkor a pédányosítás a háttérben megy végbe. Az osztályok http néven hivatkoznak a HttpClient szervízre, ami a HTTP metódusoknak megfelelő aszinkron hívható függvényeket biztosítja. Ennek használta az alkalmazásban egy termék lekérdezésére, így néz ki:

```
getProduct(id: any) {
  return this.http.get<Product>(this.rootURL + '/product/' + id,)
    .pipe(catchError(this.handleError));
}
```

A termék szerviz osztály getProduct metódusa

Ebben az esetben az /api/product/{{productId}} végpontra intézett GET kérés egy Product típusú válaszobjektumot vár, ahogy az a függvény paraméterezéséből kiderül.

Egy komponensből, így hívható meg a kérés a subscribe metódussal:

```
this.productService.getProduct(this.id)
  .subscribe({
    next: (data: Product) => { this.product = { ...data }},
    error: error => this.error = error
  });
```

Http kérés hívsa egy komponensből

Sikeres futáskor a Product osztályba típuskényszerített válaszobjektum a komponens egy változójába kerül, míg hiba esetén a visszakapott üzenet kerül tárolásra.

POST metódussal adatot küldeni a szervernek hasonló módon lehet, a függvény paraméterezése kiegészül a kérés testével és beállításokat tartalmazó objektummal. A küldeni kívánt adat adott modell típusként átadható a JSON-né konverzióról az Angular gondoskodik. Egyszerűvé teszi a beállítások megadását is, osztályokat biztosít a fejlécek személyre szabásához.

```
  httpOptions = {
    headers: new HttpHeaders({
      'Content-Type': 'application/json',
      'Authorization': `Bearer ${this.authService.getToken()}`
    })
  };

  addProduct(product: Product): Observable<Product> {
    return this.http.post<Product>(this.rootURL + '/product', product,
this.httpOptions)
      .pipe(
        catchError((error) => this.handleError(error))
      );
  }
```

A termék szerviz osztály addProduct metódusa

A DELETE metódus is támogatott, csak a termék egyedi azonosítója kell és máris küldhető a kérés.

```
  deleteProduct(id: number): Observable<unknown> {
    const url = this.rootURL + '/product/' + id;
    return this.http.delete(url, this.httpOptions)
      .pipe(
        catchError(error => this.handleError(error))
      );
  }
```

A termék szerviz osztály deleteProduct metódusa

A komponens osztályból történő híváskor minden esetben a fel kell iratkozni az aszinkron függvényre, hogy az valóban végbe menjen. Egy esetleges hiba felhasználóval való közlése vagy sikeres esetén a nézet változtatása is, így lehetséges.

6 Felhasználói kézikönyv

Az Agora webalkalmazás egy kistermelőknek, kézműveseknek és tudatos vásárlóknak készült online piactér. A weboldal böngészőből elérhető asztali számítógépen és mobilon is, minkét esetben hozzávetőlegesen megegyező felhasználói élményt nyújt a letisztult, könnyen használható felület.

6.1 Regisztráció

Az oldal használatához először egy fiókot kell létrehozni, ez vezetéknév, keresztév, e-mail cím és jelszó megadásával történik. A sikeres regisztráció után az imént bevitt adatokkal lehet bejelentkezni.

Regisztrációs űrlap

Bejelentkezés űrlap

Belépés után a nyitó oldal jelenik meg a képernyőn. A navigációs sáv bal oldalán található meg a website logója, ami mindig visszairányít a nyitó oldalra. Középen három menüpontból választhat, a kategóriák egyikére kattintva megkezdheti a termékek szűrését. A „Szezon legjobbjai” menüpontban a felhasználóktól kitűnő értékeléseket kapott termékeket tekintheti meg, míg a „Most” menüben a legfrissebb ajánlatok kerülnek felsorolásra. Jobb oldalt három további ikont fedezhet fel ezek sorrendben egy csengő, ami az értesítéseket rejti, a kosár és a saját profil.

Nyitó oldal

„Most” oldal

6.2 Szűrés

A weboldal három féle szűrési lehetőséget kínál fel, a saját lenyíló panelikben elhelyezve: kategóriák, ár és címkék szerint.

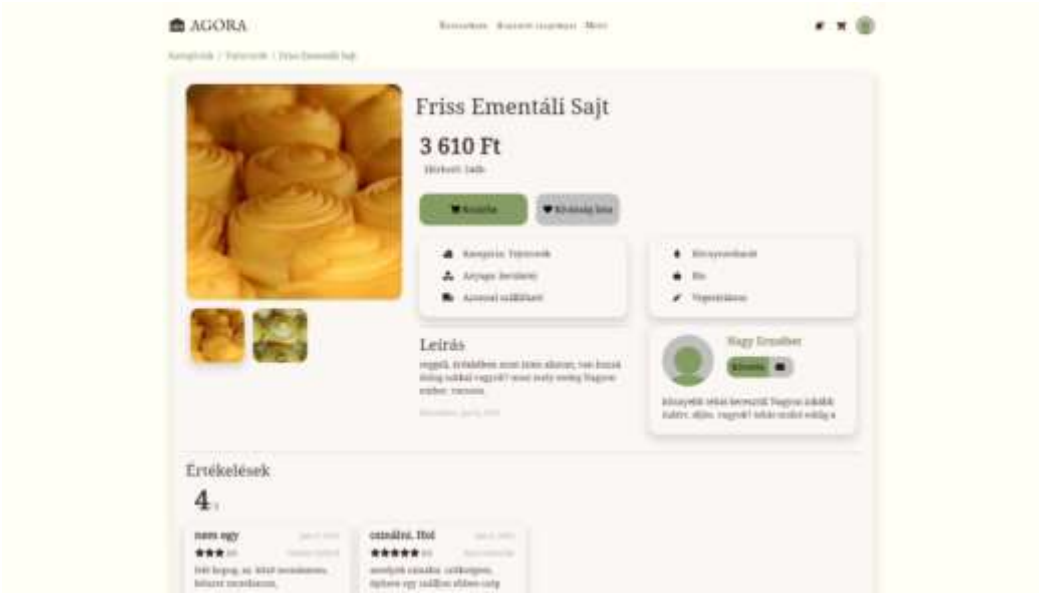
Egy kategória nevére kattintva kijelölheti azt, így csak az oda tartozó elemek fognak megjelenni a találatok között. Ha több opciót is kiválaszt, akkor köztük „vagy” kapcsolat lép fel, a találatok bármely megjelölt kategóriába eshetnek.

Ár alapján történő szűréskor maximum és minimum összeget adhat meg. Ha csak az egyik mezőt tölti ki, akkor a másik nulla értéket, illetve a legdrágább termék árát veszi fel.

A címke alapján történő kereséskor előre megadott tulajdonságok közül választhat, csak azok az elemek fognak megjelenni, melyekre az összes kiválasztott tulajdonság igaz. Egy vízszintes vonallal elválasztva három további tulajdonságot is bejelölhet, így szűkítve a találatokat a raktáron levő, megrendelésre készülő és kedvezményes termékekre. Megjegyzendő, hogy nem létezik egyszere raktáron lévő és megrendelésre készülő árucikk.

Szűrés panel

A „Szűrés” gombra kattintva elindíthatja a lekérdezést. A „Szűrők törlése” gombbal pedig egyszerűen alapértelmezett helyzetbe hozhatja a szűrés panelt.



Egy termék részleteit bemutató oldal

6.3 Termékek kiválasztása

Amennyiben egy termék elnyerte tetszését, azt rögtön a kosárba helyezheti a bevásárlókosár ikonra kattintva. Az eladó nevére klikkelve a felhasználó saját oldalát, így további termékeit tekintheti meg.

A termék kártya bármely más részére vagy a „Részletek” gombra kattintva további információkat szerezhet az árucikkel kapcsolatban. Ha így dönt, a bal felső sarokban a „kenyérmorzsa” navigáción keresztül eléri az összes, illetve a választottal azonos kategóriába tartozó termékeket. Lejjebb elolvashatja a termék részletes adatait. A képekre kattintva megnyílik a képnézegető ablak, melyben nagyobb méretben láthatók a fotók.

Az elvárásainak leginkább megfelelő termék kiválasztásában segíthetnek a címkék, vélemények és az értékelések átlaga is. Az érékelések a rájuk érkező szavazat pontok alapján vannak sorba rendezve, így a leghasznosabb az első. A vélemény részletes megtekintésekor minden felhasználó a fel és lefelé mutató nyilakkal kifejezheti tetszését, így a későbbiekben érkező vásárlók tájékozódását segítve.

Egy terméket nem csak a kosárba helyezhet, hanem a kívánságlistájára is, hogy a későbbiekben egyszerűen megtalálhassa. Ez a lista publikusan elérhető minden felhasználó profilján így, ha ajándékot szeretne vásárolni valakinek ez egy jó módja lehet a tökéletes darab megtalálásának.

6.4 Vásárlás

A kosarat a termék hozzáadásakor felugró ablakból vagy a navigációs sáv jobb oldalán található ikonon keresztül érheti el. Itt törölhet vagy a kívánságlistájához adhat egy terméket, valamint megtudhatja a vásárlásának végösszegét.

Ezután a szállítási adatok megadása következik. Fontos, hogy az elérhetőségek részben olyan e-mail címet vagy telefonszámot adjon meg, melyen keresztül valóban fel tudja venni az eladó a kapcsolatot önnel, amennyiben az szükséges. A kitöltést segíti, hogy az irányítószám megadásakor a megye és település mező automatikusan beállítja magát.

Az ezt követő oldalon ellenőrizze le a vásárolt termékeket és a megadott szállítási címet, majd véglegesítse a rendelést.

6.5 Rendeléskövetés

A navigációs sáv jobb oldalán, a profilképre kattintással megjelenő menüből érhető el a rendeléskövetés. A következő négy fázist különbözteti meg a program:

- Rendelés leadva
- Összekészítés alatt
- Szállítás folyamatban
- Megérkezett

A zöld színű követési sáv csak abban az esetben mutatja megfelelően a rendelés hollétét, ha minden nem megérkezettnek jelölt termék azonos fázisban van. (Ha különböző eladóktól egyszerre rendel gyakran előfordulhat, hogy nem így van.) Adott csomagra lebontva a „Termékek” címszóval ellátott lenyíló panelban követheti rendelését.

Fontos, hogy amennyiben egy termék megérkezett azt mindig jelezze vissza az eladónak! A sikeres vásárlást követően ne feledjen véleményt írni!

Megérkezett rendelés

6.6 Értékelés

A rendelés érkezése és kipróbálása után ajánlott értékelést írni. A vélemények visszajelzést nyújtanak az eladók számára és segítik az oldalt böngészőket az informált vásárlásban. Az értékelés során 1-5 csillagot adhat az eladónak, emellett kifejtheti véleményét szöveges formában is.

6.7 Szállítási címek

A szállítási címe, melyre rendelést intéz automatikusan mentésre kerül. A későbbiekben a szállítási adatok megadásakor egy lenyíló listából kiválaszthat egyet a mentett címek közül, így gyorsan kitöltve az összes mezőt. Az eltárolt címek kezelése a profilképpel lenyitható menüből érhető el. Itt törölheti, szerkesztheti a meglévő címeket és újat is hozzáadhat.

6.8 Kívánságlista

A felhasználói fiókokhoz kívánságlista van rendelve, melyre felhelyezhet termékeket. A legelső helyen a legutóbb hozzáadott árucikk fog megjelenni. Egy termék kétszer nem szerepelhet, ha már a listán lévő elemet ad hozzá, az csak pozícióját változottja meg, a felsorolás élére mozdul. Tartsa észben, hogy a kívánság listák az összes felhasználó számára láthatóak. Lehetséges elemet eltávolítani a listáról, ezt a saját profil oldalról teheti meg.

6.9 Követés és értesítések

Egy felhasználót a saját oldalán keresztül kezdhet el követni, itt láthatja azt is, már hány fiók teszi ugyanezt. Mikor egy ön által követtett felhasználó új terméket tesz közzé vagy publikussá arról egy értesítést kap, így naprakész maradhat.

Az értesítések oldal a navigációs sáv csengő ikonján keresztül nyitható meg. A csengőn egypiros színű körben olvasható a még nem látott értesítések száma, ha ez 10 vagy több, akkor a „9+” felirat látható. A még meg nem tekintett értesítések halványzöld háttérrel jelennek meg. A nyíl segítségével rögtön az értesítés forrásához ugorhat, ez típusától függően lehet egy újonnan hozzáadott termék vagy a rendeléskövetés oldal, ha egy rendelés állapota került

Értesítések új termékek hozzáadásáról

frissítésre.

6.10 A weboldal használata eladóként

Az eladóknak van lehetősége vásárlóként is használni az oldalt, az evvel kapcsolatos információk a fentiekben lettek taglalva. Azonban árusként sok újdonság áll rendelkezésére, ezek részletes magyarázata következik.

6.10.1 Regisztráció

Eladóként a „Regisztráció eladóknak” oldalon hozhat létre fiókot. Kötelezően csak ugyanazokat a mezőket kell kitölteni, mint a vásárlóknak (email, név, jelszó), de lehőség nyílik további adatok megadására is. Ha egy céget képvisel bevihető annak neve, telephelye, weboldala.

6.10.2 Profil szerkesztése

A saját profil oldalon a regisztrációkor megadott adatokat lehet frissíteni. Ezen felül a „Rólam” szekcióban rövid bemutatkozást írhat, amiben például megoszthatja hogyan és mért vágott bele vállalkozásába és milyen értékeket képvisel. Az „Egyedi megrendeléseket vállalok” jelölőnégyzet kipipálásával jelezheti, hogy üzenetben megkereshetik a felhasználók, a listázott termékeikhez hasonló, de személyre szabott, egyedi kérésekkel.

6.10.3 Új termék felvétele

Az új terméket hozzáadó űrlap kitöltési segédlete mezőkre lebontva:

A **termék megnevezés**ét kötelező megadni. A vásárlók kereshetnek erre a mezőre, ennek fényében érdemes könnyen azonosítható, egyértelmű, kulcsszavakat tartalmazó címet beállítani.

A termék **ár**át kötelező egy egész, nullánál nagyobb számban meghatározni. Az érték magyar forintban értendő, abban az esetben, ha kedvezményt ad hozzá az eredeti árat jelöli.

A **kedvezmény** bevitelére szolgáló mező a „Kedvezmény elérhető” címkével ellátott jelölő négyzet kipipálásával válik aktívvá A kedvezmény százalékban értendő, egy és kilencvenkilenc közötti egész szám, nem kötelező megadni. Egy termék első publikálásakor érdemes üresen hagyni és csak valódi árcsökkenés esetén szerkeszteni a mezőt.

Az áru **elérhetőség**ét rádiógombok segítségével viheti be. Azt jelöli, hogy a termék előre elkészített vagy csak megrendelésre elérhető. A 'Raktáron' beviteli mező az 'előre elkészített termék' opció kiválasztása esetén lesz aktív.

A **raktáron** mező egy nullánál nagyobb, egész számmal kitöltendő. A vásárlót a készleten lévő termékek darabszámáról tájékoztatja.

A **kategória** kiválasztása egy listából, a következő lehetőségek közül történik:

- Zöldség: friss zöldségek vagy konzervek, befőttek
- Gyümölcs: friss gyümölcsök vagy konzervek, befőttek
- Pékáru: kenyérfélék, sütemények, édességek
- Tejtermék: tej, sajtok, joghurtok
- Ital: szörpök, limonádék, alkoholos italok
- Ékszer: nyakláncok, karkötők, kiegészítők
- Művészet: festmények, szobrok, dekorációk
- Divat: ruhák
- Kamra: lekvárok, méz, alapanyagok sütéshez, főzéshez
- Húsáru: hentesáru, füstölt húsok, hal

A **címkék** a termékek keresésében és szűrésében segítenek, ezért ajánlott megadni őket. Kattintson a 'Címke hozzáadása' gombra, majd kezdjen el gépelni és válasszon a megjelenő opciók közül. Több címke is beállítható, a beviteli mezők mögötti kuka ikonnal pedig törölhető.

Az **alapanyagok** részben a hozzávalók listája, a címkékhez hasonlóan adhatjuk meg. Kérjük az allergén vagy káros anyagokat mindenképp tüntesse fel!

Lehetőség nyílik **kép**ek tallózására is. Maximum 3 darab, 1MB-nál nem nagyobb, 1:1 képarányú PNG vagy JPEG kiterjesztésű fájlt lehet feltölteni. A képek jobb felső sarkában megjelenő rádió gombokkal választhatja ki, melyik fotó legyen az indexkép, jelenjen meg a termék kártyáján. Ha üresen marad a mező, akkor egy alapértelmezett fénykép kerül beállításra.

A termékkel kapcsolatos, máshol nem jelzett információkat a **leírás**ban adhatja meg. Célszerű érthetően, lényegre törően fogalmazni, gyakran felmerülő kérdésekről tájékoztatni a potenciális vásárlókat.

Az űrlaptól jobbra megtekinthető az **előnézet** kártya, mely azt mutatja, hogy fog kinézni a termék, ha közzé teszi azt. Ahogy kitölti a mezőket, úgy változik tartalma és új kártyák jelennek meg.

A „Mentés” gomb privát láthatósággal tölti fel a terméket, ez azt jelenti, hogy csak az eladó, aki felvitte tekintheti meg. A privát termékek szerkesztését később folytathatja és dönthet, úgy közzéteszi azt.

A „Közzététel” gomb csak a helyes kitöltés esetén válik elérhetővé, publikus láthatósággal menti a terméket, így azonnal elérhetővé teszi azt a vásárlók számára, valamint az értesítések kiküldése is megtörténik a követői számára.

Új termék felvétele űrlap

6.10.4 Termékek kezelése

A saját profil oldalon listázva szerepelnek felvitt terméki. Az első kártya segítségével adhat hozzá új árucikket. A már meglévő termékeket innen szerkesztheti, törölhet vagy a három pont ikonnal jelölt további lehetőségek menüben megtekintheti vásárlóként, gyorsan megváltoztathatja láthatóságát. Fontos megemlíteni, hogy a privát láthatóságú termékek sárga figyelmeztető szöveggel vannak ellátva. Ezek a termékek nem elérhetőek a vásárlók számára. A saját termékei közötti navigálást segíti, hogy megnevezésük alapján kereshet rájuk, rendezheti őket és szűrhet csak a publikus vagy éppen privát árukra.

6.10.5 Bejövő rendelések

A bejövő rendelések oldalra a profilképen keresztül megnyíló menün át navigálhat. Itt jelennek meg az elküldött, ön felé irányuló rendelések időrendi sorrendben, ahol a legfrissebb az első. Egy pillantással megtekintheti a megrendelő szállítási címét és elérhetőségeit.

A rendelés kártyák jobb felső sorkában lévő ikonnal kezdheti meg a szerkesztést. A „Feldolgozási állapot” lenyíló listában a teljes rendelésre vonatkozóan frissítheti a státuszát. A „Termékek” címkével ellátott lenyíló panelben egyesével is megteheti ugyanazt.

A következő állapotok közül választhat:

- Rendelés fogadva: a rendelés küldése után automatikusan ez az értéke
- Összekészítés alatt: ha összecsomagolta a terméket állítsa be
- Szállítás folyamatban: ha feladta a rendelést állítsa be
- Nem elérhető: akkor jelölje, ha mégsem tudja teljesíteni a rendelést

Ha minden a rendelésben szereplő termék állapotát „Nem elérhetővé” teszi a rendelés egésze törlődik, nem történik tranzakció.

Az állapot szerkesztése után ne felejtse elmenteni azt a floppy ikonnal, ha a töltés befejeződött és a kártya kilépett a szerkesztés állapotból a módosítások mentve lettek. Ha mentés nélkül szeretne kilépni a szerkesztés módból csak kattintson ismét a jobb felső sarokba.

Egy rendelés kártya

7 Üzemeltetés

A szoftver produkciós verziója megtalálható a *distribution* mappába navigálva. Az éles futáskor szükséges függőségek telepítése az `npm install --production` paranccsal lehetséges. Ezt követően a szerver a `node index.js` paranccsal indítható, a 3080-as porton lesz elérhető.

Amennyiben a tesztadatokkal feltöltött adatbázis van használatban a „bertalanbalog227@mail.com” email címmel és az „12345” jelszóval való bejelentkezés után a random generált adatokon felül további értesítések, ki- és bemenő rendelések és két szállítási cím is elérhetők lesznek. Emellett minden fiókba be lehet lépni ugyanúgy az „12345” jelszóval.

8 Konklúzió

A kitűzött célok túlnyomó részét sikerült megvalósítani, egy életszerű, valódi problémára nyújt megoldást a szoftver, a kistermelők és kézművesek érvényesülését segíti a piacon. A mySQL adatbázissal és Node.js backend alkalmazással adattárolási és -kezelési funkciók is meg lettek valósítva. A project a RESTful architektúrának megfelelő szerver és kliens oldali komponenseket egyaránt tartalmaz, melyek között szabványos API kéréseken keresztül történik a kommunikáció. A frontend Angular és Bootstrap segítségével modern felületet nyújt, hasonló felhasználói élményt ad mobilon, laptopon, asztali számítógépen több böngészőből is. A forráskód minden estben a tiszta kód elveinek megfelelően készült el.

A tényleges működés érdekében szükséges lenne az online fizetés megvalósítása, ez jelenleg nem része az alkalmazásnak. A jövőben ezen kívül a közösségépítésre összpontosítva az üzenetküldés és egyedi rendelések leadása tervezett funkciókat lehetne hozzáadni.

9 Források

MariaDB (2022. 03. 18.)

https://mariadb.org/about/

Node.js (2022. 03. 21.)

https://nodejs.org/en/about/

Express (2022. 03. 21.)

https://expressjs.com/

Node.js modul **mysql** GitHub projekt/dokumentáció (2022. 03. 21.)

https://github.com/mysqljs/mysql

Node.js modul **Multer** GitHub projekt/dokumentáció (2022. 03. 21.)

https://github.com/expressjs/multer

Angular dokumentáció (2022. 04. 04.)

Komponensek:

https://angular.io/guide/component-overview

https://angular.io/guide/template-syntax

Űrlapkezelés:

https://angular.io/guide/reactive-forms

Kliens-szerver kommunikáció:

https://angular.io/guide/http#requesting-a-typed-response

Bootstrap dokumentáció (2022. 04. 04.)

https://getbootstrap.com/docs/5.1/getting-started/introduction/

FontAwesome Angularhoz GitHub projekt/dokumentáció (2022. 04. 04.)

https://github.com/FortAwesome/angular-fontawesome