



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

Tanuló(k) neve¹: Németh Kristóf Ármin, Tóth Bálint
Képzés: nappali
Szak: 5 0613 12 03 Szoftverfejlesztő és tesztelő technikus

A záródolgozat címe:

REFLEX.

Konzulens: Horváth Norbert
Beadási határidő: 2023. 04. 28.

Győr, 2022. 10. 01

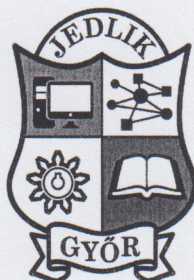
Módos Gábor
igazgató

¹ Szakmajegyzékes záródolgozat esetében több szerzője is lehet a dokumentumnak, OKJ-s záródolgozatnál egyetlen személy ad le záródolgozatot.



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

Konzultációs lap²

	A konzultáció		Konzulens aláírása
	ideje	témája	
1.	2022.11.11.	Témaválasztás és specifikáció	
2.	2023.03.10.	Záródolgozat készültségi fokának értékelése	
3.	2023.04.21.	Dokumentáció véglegesítése	

Tulajdonosi nyilatkozat

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

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

Győr, 2023. április 28.

tanuló aláírása

tanuló aláírása

tanuló aláírása

² Szakmajegyzékes, csoportos konzultációs lap

1 Tartalom

Bevezetés	3
Specifikáció	3
Fejlesztői csapat	4
A csapatmunka megvalósítása	4
Adatbázis	5
Az adatbázis diagramja	5
Az adatbázis leírása, magyarázata	5
REST API	7
Áttekintés	7
Authentikáció	7
Bejelentkezés	7
Erőforrások	8
Felhasználók	8
Regisztráció	9
Az összes user kilistázása	9
Egy adott user lekérése	10
Egy adott user frissítése	10
User statisztika	11
Termékek	11
Termékek lekérdezése	12
Termék hozzáadása	13
Termék frissítése	14
Termék törlése	15
Rendelés hozzáadása	16
Rendelés statisztika	17
Kosár	17
Lehetséges visszatérések	18
Backend	19
Keretrendszer és könyvtárak	19
Útválasztás	19
Megugrandó feladatok	20

Adatmodellek.....	21
Tesztelés	22
Frontend	22
Keretrendszer, könyvtárak, fejlesztői környezet	22
Fejlesztés menete, komponensek, oldalak.....	23
Redux	27
Mobilnézet	28
Tesztelés	30
Felhasználói kézikönyv	30
Authentikáció	30
Kapcsolat	31
Termékböngészés	31
Kosár	31
Fizetés.....	31
Üzemeltetés	32
Összegzés.....	32
Forrás.....	33

2 Bevezetés

2.1 Specifikáció

Középiskolás éveink során saját bőrünkön érzékeljük a hazai játék-, illetve hardverpiac kiforratlanságát. Szoftverünkkel erre a problémára szeretnénk megoldást kínálni.

REFLEX. gamer periféria webshopunk célja, hogy az igényes játékosok számára minőségi és hatékony perifériákat biztosítson, amelyek javítják az élményt és növelik a játék teljesítményét.

Webshopunk kialakításakor kamatoztattuk az e-sportokban szerzett tapasztalatunkat, ez a garancia arra, hogy termékeink biztosan megfelelnek bármely gamer számára, legyen szó casual vagy professzionális játékosról. Mindketten szeretünk játszani, Bálint versenyszerűen is e-sportolt, több hazai organizációban megfordult, mint igazolt játékos.

Kínált perifériáink közé tartozik a billentyűzet, az egér, a fejhallgató és a számítógépes egérpad. Minden eszköz kiválasztása során szem előtt tartjuk a minőséget, a funkcionalitást és a játékélményt.

A shopban található eszközeink közül a legtöbb kompatibilis a legújabb operációs rendszerekkel és játékokkal, valamint lehetőség van a különféle konfigurációkra és színekre is.

A billentyűzetek között találhatóak a mechanikus billentyűzetek, amelyek gyors és pontos reakciókat biztosítanak, az egerek között a vezetékes és a vezeték nélküli opciók is megtalálhatóak, az egérpadok közt fellelhetők gyorsabb, illetve lassabb padok is, a fejhallgatók pedig kiváló minőségű hangzást és kényelmet biztosítanak a hosszabb játékmenetekhez.

Oldalunk esztétikája hűen tükrözi a gamer perifériák látványvilágát, ezzel egy átlátható, könnyed és letisztult weboldalt eredményezve. Az eszközök működését és precizitását pedig a webshop fantázianeve, a REFLEX. hordozza magában.

Célunk, hogy shopunk összegyűjtse és egy helyen, megfizethető áron kínálja a termékeket, minden játszani vágyó számára.

A jövőben tervezünk fejleszteni a projektet, ugyanis jó felkészülést szolgálhat az egyetemi évekre, tapasztalatot adhat és bármikor referenciaként hivatkozhatunk erre a webshopra. Legközelebbi cél egy megfelelő admin page kialakítása lenne, ahonnan könnyen lehetne új termékeket hozzáadni, kezelni a felhasználókat, illetve elemezni az előző hónapok statisztikáit. A megvalósítás

könnyítésének érdekében, a backend rendelkezik két, olyan API kéréssel, amely az elmúlt két hónap statisztikáit mutatja be.

2.2 Fejlesztői csapat

Csapatunk kialakításánál törekedtünk arra, hogy olyan emberekkel kerülhessünk össze, akikkel a középiskolás évek során megtaláltuk a közös hangot, ezáltal megkönnyítsük a projektmunkát. A tanév eleji fejlesztést hárman kezdtük meg, azonban csak ketten, Németh Kristóf Ármin és Tóth Bálint fejeztük be.

2.3 A csapatmunka megvalósítása

Két személy közös munkája átláthatóbb és könnyebb, mint egy három fős csoport esetében, így nem bontottuk le a munkát backendre, frontendre, hanem közösen szerkesztettük a projektet, apróbb részfeladatokra bontva azokat. Így kerültük el az esetleges ütközéseket, várakozásokat, amelyek kialakulhattak volna egyéb esetben.

A fájlok tárolására eleinte *google drive*-ot használtunk, azonban megismerkedtünk idővel a *github*-al, amely sokkal könnyedebb fájmegosztást, fejlesztést és tárolást biztosít, mint bármely másik online tárhely.

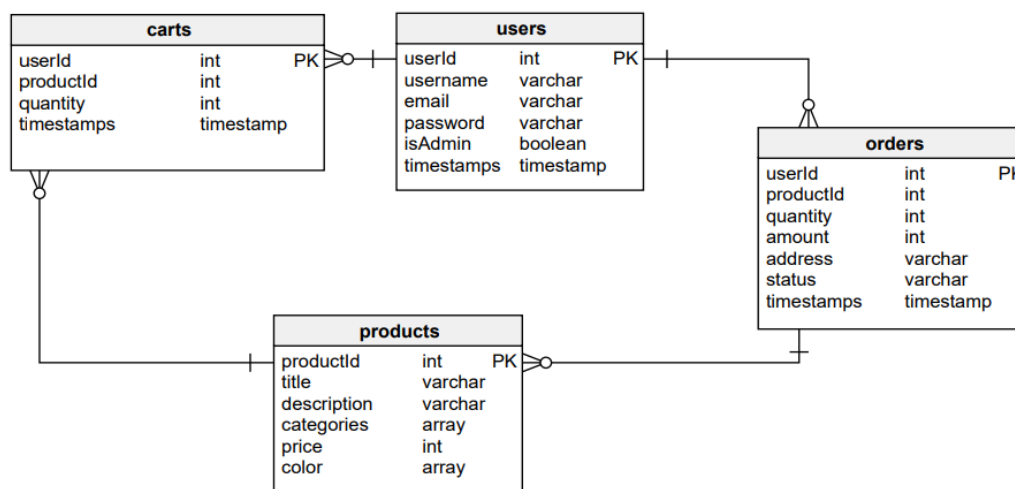
Közös hobbiainknak és programjainknak köszönhetően óránkénti kapcsolatban voltunk az év során, ezért mindig volt lehetőségünk egyeztetni személyesen is, de az otthonról történő közös fejlesztések során általában *Discord*-ot használtunk. Számunkra ez tökéletesen megfelelt az év egészében, nem is váltottunk. Hetente, esetenként kéthetente tartottunk ehhez hasonló „értekezleteket”.

A dokumentáció írását a *Microsoft Online Word* segítségével készítettük, amely lehetővé teszi az egyidejűleg történő közös szerkesztést.

3 Adatbázis

3.1 Az adatbázis diagramja

A relációs adatmodell létrehozására a *Vertabelo* nevű online adatbázis modellező oldalt alkalmaztuk. Kezdetekben több táblát terveztünk, azonban egész hamar kialakult az alábbi verzió, amely véglegessé is vált már a fejlesztés elején. Természetesen a további projektek során még bővíülhet az adatbázis, de jelen állapotban még nem igényel több táblát.



1. Adatmodell Vertabelo segítségével

3.2

Az adatbázis leírása, magyarázata

Az adatbázis 4 táblából (kosár, felhasználók, termékek, rendelések) áll. A felhasználó táblát a `userId` mező köti össze a kosárral és a rendelésekkel, míg a termékeket a `productId` kulcs. Az adatbázist a *MongoDB* nyílt forráskódú dokumentumorientált adatbázis szoftver tárolja. Dokumentumorientáltságának köszönhetően rugalmasabb, mint bármely másik SQL alapú adatbázis. Nem sorokban és oszlopokban, hanem kulcs-érték párokban raktározza az adatokat. Az adatokat BSON, azaz bináris JSON-ben archiválja, amelyet lekérdezéskor lefordít JSON-re az olvashatóság érdekében. A *MongoDB* nem alkalmaz tárolt eljárásokat, azonban számos saját lekérdező metódust vonultat fel, például a `find()`, vagy az `aggregate()` metódus, amely könnyed, erős lekérdezéseket, illetve analízist biztosít. Példaként, nézzünk meg egy egyszerű get kérést,

amely kilistázza az összes terméket, valamint leszűri, szükség szerint rendezi azokat, ha a request-ben van termékkategóriára szűrés vagy rendezési feltétel, hogy a legújabbakat listázza elől.

```
router.get("/", async (req, res) => {
  const qNew = req.query.new;
  const qCategory = req.query.category;
  try {
    let products;
    if (qNew) {
      products = await Product.find().sort({ createdAt: -1 }).limit(5);
    } else if (qCategory) {
      products = await Product.find({ categories: { $in: [qCategory] } });
    } else {
      products = await Product.find();
    }
    res.status(200).json(products);
  } catch (error) {
    res.status(500).json(error);
  }
});
```

2: Termékek lekérdezés szűrési feltételekkel

Az adatbázis cluster-t a backend egy url-en keresztül éri el, amelyet a `.env` fájlból tettünk elérhetővé környezeti változóként, ezt dolgozza fel `dotenv`, majd a `mongoose` package és létesít kapcsolatot a mongo szerverrel. Ezt a backend fejezeten belül mutatjuk be részletesebben.

Github repository-nkban fellelhető az adatbázis export fájlja (dump), illetve az API kérések tesztjei, amelyekkel együtt teszteltük az adatok elérhetőségét, illetve a kapcsolatok megfelelő működését is.

4 REST API

4.1 Áttekintés

Ez a dokumentáció a REFLEX. webshop által használt JSON alapú API használatát mutatja be. A kéréseket a fejlesztési fázisban egy, a 5000-s porton futó Node.js webservert fogadta, így a példákban a végpontok „http://localhost:5000/” kezdetűek. A teszteléshez a *Thunder Client*, Visual Studio Code Extension-t használtuk.

4.2 Authentikáció

Az erőforrások post és get kéréseihez a legtöbb esetben token-re, azaz hozzáférési azonosítóra van szüksége. A token a JSON Web Token package biztosítja. Bejelentkezést követően egy JWT-t tartalmazó objektum érkezik válaszként, amely később verifikálja az adott felhasználót és engedélyezi azon tevékenységek elvégzését, amit beosztása megenged.

4.2.1 Bejelentkezés

A kérés:

```
POST localhost:3000/api/auth/login
```

```
{
  "username": "test123",
  "password": "test1234"
}
```

A válasz:

```
Status: 200 OK
```

```
{
  "_id": "6448f4f3dce75e1c45c48bed",
  "username": "test123",
  "email": "test@gmail.com",
  "isAdmin": false,
  "createdAt": "2023-04-26T09:54:59.853Z",
  "updatedAt": "2023-04-26T09:54:59.853Z",
  "_v": 0,
  "accessToken":
  "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjY0NDhmNGYzZGNlNzVlMWM0NWU0OGJlZCIsImIzQWRtaW4iOmZhbHN1LCJpYXQiOiJlMjODI1MDMzMzcsImV4cCI6MTY4MjUxNDEzN30.ouVjaE3XNKSDeUtMKcI-6K9l8YkRh11g0NQMTG47Yow"}
```

A sikeres bejelentkezést után a HTTP fejléc „token” mezőjébe helyezzük a „Bearer {{token}}” kulcsot és a további kérések innen lesznek autentikálva.

4.3 Erőforrások

Az API kérések egyik legfőbb célja, hogy elérhetővé tegye a különböző erőforrásokat a frontend számára. A következőkben a felhasználó, termékek és a rendelések táblára vonatkozó kérések felépítését, schema-ját, illetve a várható válaszokat fogjuk taglalni.

4.3.1 Felhasználók

A JSON objektumnak kötelezően tartalmaznia kell a táblázatban foglalt mezőket, az isAdmin logikai mezőt kivéve, hiszen azt csak később, az admin felületen lehet megváltoztatni, természetesen a tesztek kivételével. Továbbá kritérium az email és a username egyedisége, amelynek hiányában nem regisztrálható egy felhasználó. A tábla rendelkezik még timestamp mezőkkel, amik a *MongoDB* beépített mezői, tartalmazzák a létrehozás és a legutóbbi módosítás dátumát.

Paraméter	Típus	Kritérium	Leírás
_id	objectId	-	A mongo rendeli hozzá
username	string	unique, required	Felhasználónév
email	string	unique, required	Felhasználó emailje
password	string	unique, required	Felhasználó jelszava
isAdmin	boolean	default: false	Admin jog
timestamps	date	-	A mongo rendeli hozzá

A megfelelő adatkezelés és a biztonság érdekében az adatbázisba a felhasználó jelszavak már titkosítva érkeznek meg, amelyet a CryptoJS package és az AES titkosítási módszer segítségével kódolunk és dekódolunk.

```
password: CryptoJS.AES.encrypt(  
  req.body.password,  
  process.env.PASS_SECRET  
).toString(),
```

3: Jelszótitkosítás CryptoJS-el

4.3.2 Regisztráció

A kérés:

```
POST localhost:3000/api/auth/register
```

```
{
  "username": "test123",
  "email": "test@gmail.com",
  "password": "test1234"
}
```

A válasz:

```
Status: 200 OK
```

```
{
  "username": "test123",
  "email": "test@gmail.com",
  "password": "U2FsdGVkX1/y9g+kwB2S3GA5Bo/hgR7bnbNSIXkmnUE=",
  "isAdmin": false,
  "_id": "6448f4f3dce75e1c45c48bed",
  "createdAt": "2023-04-26T09:54:59.853Z",
  "updatedAt": "2023-04-26T09:54:59.853Z",
  "__v": 0
}
```

4.3.3 Az összes user kilistázása

Főként az admin fiúknél lehet használni, amikor keresni kell egy kilistázott user-t.

A kérés:

```
GET localhost:3000/api/users/
```

Egy példa a válaszből:

```
Status: 200 OK
```

```
{
  "_id": "6448f4f3dce75e1c45c48bed",
  "username": "test123",
  "email": "test@gmail.com",
  "password": "U2FsdGVkX1/y9g+kwB2S3GA5Bo/hgR7bnbNSIXkmnUE=",
  "isAdmin": true,
  "createdAt": "2023-04-26T09:54:59.853Z",
  "updatedAt": "2023-04-26T09:54:59.853Z",
  "__v": 0
}
```

4.3.4 Egy adott user lekérése

Felhasználók is lekérdezhetik, szükséges lehet, mikor a webshopban a saját adataikat szeretnék látni, vagy jelszóemlékeztetőt kérnek. A kérésben meg kell adni annak a user-nek az id-ját, amelyet meg akarunk találni.

A kérés:

```
GET localhost:3000/api/users/find/{_{id}}
```

A válasz:

Status: 200 OK

```
{
  "_id": "6448f4f3dce75e1c45c48bed",
  "username": "test123",
  "email": "test@gmail.com",
  "isAdmin": true,
  "createdAt": "2023-04-26T09:54:59.853Z",
  "updatedAt": "2023-04-26T09:54:59.853Z",
  "__v": 0
}
```

4.3.5 Egy adott user frissítése

Egy, már regisztrált user adatainak szerkesztése, például a felhasználónevének átírása, admin jogokkal való felruházás az admin fülön, jelszóváltoztatás elfelejtett jelszó esetén. A kérésben meg kell adni annak a user-nek az id-ját, amelyiket szerkeszteni szeretnénk, valamint a szerkesztendő mezőt és az új értékét.

A kérés:

```
PUT localhost:3000/api/users/{_{id}}
```

```
{
  "username": "test456"
}
```

A válasz:

Status: 200 OK

```
{
  "_id": "6448f4f3dce75e1c45c48bed",
  "username": "test456",
  "email": "test@gmail.com",
  "password": "U2FsdGVkX1/y9g+kwB2S3GA5Bo/hgR7bnbNSIXkmnUE=",
  "isAdmin": true,
  "createdAt": "2023-04-26T09:54:59.853Z",
}
```



```
"updatedAt": "2023-04-26T11:00:01.933Z",  
  "_v": 0  
}
```

4.3.6 User statisztika

A jövőbe tekintve, már most megírtuk az API kérését a felhasználók statisztikáinak, amely a későbbi fejlesztések során nagyon hasznos lehet. Válaszobjektumként a kérés az összes idén regisztrált user számát visszaküldi hónapos bontásban, ezáltal figyelhetjük és vezethetjük statisztikaként, a regisztrációkat. Hasonló lekérdezést van tervben éves bontást figyelembe véve is, hogy abszolút minden regisztrációt analizálhasson az oldal üzemeltetője.

A kérés:

```
GET localhost:3000/api/users/stats
```

A válasz:

```
Status: 200 OK
```

```
[  
  {  
    "_id": 3,  
    "total": 2  
  },  
  {  
    "_id": 2,  
    "total": 1  
  },  
  {  
    "_id": 4,  
    "total": 3  
  }  
]
```

4.3.7 Termékek

A termékek a webshop legnagyobb táblája, legtöbb mezővel rendelkezik és a legtöbb elemet is tartalmazza. Get metódussal a termékeket lekérdezni, megtekinteni bárki tudja, azonban új terméket rögzíteni, csakis admin jogokkal rendelkező felhasználó képes. A termékekhez rendelt kép mezőbe a képek cím url-jét kell megadni, így fogja elérni és megjeleníteni a frontend. Ez hasznos, ugyanis nem kell minden termék képét lokálisan vagy táblában tárolni.

Paraméter	Típus	Kritérium	Leírás
_id	objectId	-	A mongo rendeli hozzá
title	string	unique, required	Termék neve
description	string	required	Termék leírása
img	string	required	Termék képe
categories	array	required	Termék kategóriái
price	number	required	Termék ára
brand	string	required	Termék gyártója
color	array	required	Termék színei
inStock	boolean	default: true	Elérhetőség a shopban
timestamps	date	-	A mongo rendeli hozzá

4.3.8 Termékek lekérdezése

Mindenki számára elérhető, nem szükséges se autentikáció, se admin jog.

A kérés:

```
GET localhost:3000/api/products/
```

A válaszobjektum egy példája:

Status: 200 OK

```
{
  "_id": "64478cc92831980875f3439b",
  "title": "SteelSeries Arctis 5",
  "description": "Surround sound RGB gaming headset with the
best mic and stunning hearing!",
  "img":
    "https://media.steelseriescdn.com/thumbs/catalogue/products/0096
4-arctis-5-black-2019-
edition/2d05914f40f34ebe8accceeffe9d2f41.png.1200x627_q100_crop-
fit_optimize.png",
  "categories": [
    "headset",
    "peripherals"
  ],
  "price": 129.99,
  "brand": "SteelSeries",
  "color": [
    "black",
    "white"
  ]
}
```

```
    ],  
    "inStock": true  
  }  
}
```

4.3.9 Termék hozzáadása

Csak admin joggal rendelkező user-ek tudnak post-olni terméket. Egy megfelelő admin fül kötelező tartozéka. A válaszban visszkapjuk a terméket, minden mezőjével együtt.

A kérés:

```
POST localhost:3000/api/products/
```

```
{  
  "title": "test",  
  "description": "test",  
  "img":  
  "https://media.steelseriescdn.com/thumbs/catalogue/products/0096  
4-arctis-5-black-2019-  
edition/2d05914f40f34ebe8accceeffe9d2f41.png.1200x627_q100_crop-  
fit_optimize.png",  
  "categories": ["test", "wasd"],  
  "price": 10,  
  "brand": "test",  
  "color": ["black", "white"],  
  "inStock": true  
}
```

A válasz:

```
Status: 200 OK
```

```
{  
  "title": "test",  
  "description": "test",  
  "img":  
  "https://media.steelseriescdn.com/thumbs/catalogue/products/0096  
4-arctis-5-black-2019-  
edition/2d05914f40f34ebe8accceeffe9d2f41.png.1200x627_q100_crop-  
fit_optimize.png",  
  "categories": [  
    "test",  
    "wasd"  
  ],  
  "price": 10,  
  "brand": "test",  
  "color": [  
    "black",  
    "white"  
  ]  
}
```

```
  ],
  "inStock": true,
  "_id": "64491210dce75e1c45c48bf8",
  "createdAt": "2023-04-26T11:59:12.794Z",
  "updatedAt": "2023-04-26T11:59:12.794Z",
  "__v": 0
}
```

4.3.10 Termék frissítése

Hasonlóan a post-hoz, a termékek frissítése is csak admin joggal lehetséges. A kérésben meg kell adni a szerkeszteni kívánt termék id-ját, a szerkesztendő mezőt és az új értékét. A válasz a termék minden mezőjével és már az új értékekkel tér vissza.

A kérés:

```
PUT localhost:3000/api/products/{_id}
```

```
{
  "price": 333
}
```

A válasz:

```
Status: 200 OK
```

```
{
  "_id": "64491210dce75e1c45c48bf8",
  "title": "test",
  "description": "test",
  "img":
    "https://media.steelseriescdn.com/thumbs/catalogue/products/0096
    4-arctis-5-black-2019-
    edition/2d05914f40f34ebe8accceeffe9d2f41.png.1200x627_q100_crop-
    fit_optimize.png",
  "categories": [
    "test",
    "wasd"
  ],
  "price": 333,
  "brand": "test",
  "color": [
    "black",
    "white"
  ],
  "inStock": true,
  "createdAt": "2023-04-26T11:59:12.794Z",
  "updatedAt": "2023-04-26T12:12:40.590Z",
}
```



```

    "___v": 0
  }

```

4.3.11 Termék törlése

Adminként lehetőségünk van egy terméket törölni is, a megszokott módon, saját azonosítóját megadva.

```
DELETE localhost:3000/api/products/{{_id}}
```

Status: 200 OK "Product has been deleted!"

Rendelések

A rendelések táblába mentődik minden sikeresen kifizetett rendelés. Fizetési módusként *Stripe*-

```

const OrderSchema = new mongoose.Schema(
  {
    userId: {
      type: String,
      required: true,
    },
    products: [
      {
        productId: {
          type: String,
        },
        quantity: {
          type: Number,
          default: 1,
        },
      },
    ],
    amount: {
      type: Number,
      required: true,
    },
    address: {
      type: Object,
      required: true,
    },
    status: {
      type: String,
      default: "pending",
    },
  },
  { timestamps: true }
);

```

4:Rendelések modell

ot használ a weboldal, amely segít átadni a különböző mezőknek értékeket, így megkönnyítve a rendelések létrehozását. Minden order-hez tartozik egy egyedi user azonosító, annak meghatározására, melyik felhasználó rendelte az adott terméket. Fizetni csak autentikált user-ek tudnak, így minden rendelés fog rendelkezni felhasználó azonosítóval. Ezt a *userRedux.js* adja át. Továbbá a *product* tömbbe a *cartRedux.js* beletölti a termékeket és azok mennyiségét, így megkapja a post az összes mezőértéket, amely szükséges egy rendelés létrehozására. Jövőbelátóan a felhasználóstatistikás API kéréshez hasonlóan, rendelésstatistikás lekérdezéssel is rendelkezik a backend, amely tartalmazza az idei év hónapjait és a havi bevételt az adott hónapban. Ugyan a *Stripe* rendelkezik összegzéssel, egy admin felületen való nyomonkövetés egyszerűbb és áttekinthetőbb, mint a *Stripe* weboldalán.

4.3.12 Rendelés hozzáadása

Az előbbi bekezdésben említett módon megkapja az adatokat a post és sikeresen létrehozza a rendelést, amely válaszként a rendelés mezőivel és értékeivel tér vissza. Közvetlenül nem lehet létrehozni rendelést, csakis közvetett módon, a kosárból és a fizetés után átpasszolva a különböző értékeket.

A kérés:

```
POST localhost:3000/api/orders/
```

```
{
  "userId": "640b88d40e899f107f78575a",
  "products": [
    {
      "productId": "1b",
      "quantity": 1
    },
    {
      "productId": "8e",
      "quantity": 6
    }
  ],
  "amount": 300,
  "address": "HUN"
}
```

A válasz:

```
Status: 200 OK
```

```
{
  "userId": "640b88d40e899f107f78575a",
  "products": [
    {
      "productId": "1b",
      "quantity": 1,
      "_id": "64492e73dce75e1c45c48c00"
    },
    {
      "productId": "8e",
      "quantity": 6,
      "_id": "64492e73dce75e1c45c48c01"
    }
  ],
  "amount": 300,
  "address": "HUN",
}
```

```
"status": "pending",
"_id": "64492e73dce75e1c45c48bff",
"createdAt": "2023-04-26T14:00:19.919Z",
"updatedAt": "2023-04-26T14:00:19.919Z",
"__v": 0
}
```

4.3.13 Rendelés statisztika

A kérés:

```
GET localhost:3000/api/orders/income
```

A válasz:

Status: 200 OK

```
[
  {
    "_id": 3,
    "total": 50
  },
  {
    "_id": 4,
    "total": 300
  }
]
```

Jól látható a visszatérő válaszból, hogy a 3. hónapban, márciusban 50 egységnyi „bevétele” volt a shopnak, áprilishoz pedig, az általunk előbb tesztelésként létrehozott 300 egységnyi rendelés van feltüntetve.

4.3.14 Kosár

Az alábbi képen látható a kosár modellje, benne található a felhasználó egyedi azonosítója, amely összeköti a cart és a user táblát, valamint a products tömb, amelyben tárolódnak a megvásárolni kívánt termékek azonosítói, illetve mennyiségük.

Fizetés után a táblából adódnak át az értékek a rendelések közé.

```
const CartSchema = new mongoose.Schema({
  {
    userId: {
      type: String,
      required: true,
    },
    products: [
      {
        productId: {
          type: String,
        },
        quantity: {
          type: Number,
          default: 1,
        },
      },
    ],
  },
  { timestamps: true }
});
```

5:Kosár modell

4.4 Lehetséges visszatérések

Az erőforrásokkal kapcsolatos API kérések közben ezeket a válaszokba futhat bele:

HTTP válaszkód	Válasz	Leírás
200 OK	A létrehozott, lekérdezett, törölt vagy szerkesztett JSON objektum	Megfelelően lefutott a kérés
400 Bad request	Bad request	Hibás a kérés, a lekérdezést nem lehet értelmezni, esetleg hibás az útvonal, hiányzik valamely kötelező mező
401 Unauthorized	Wrong credentials	Sikertelen a belépés, ezt okozhatja rossz, vagy nem egyező felhasználónév és jelszó
401 Unauthorized	You are not authenticated	Nincs autentikálva az adott felhasználó, az nem végezhet el adott tevékenységet, amíg be nem lép
403 Forbidden	You are not allowed to do that	Admin jogok nélkül, adminok számára elérhető kérés elindításakor
500 Internal Server Error	Internal Server Error	Nem sikerült feldolgozni vagy nem sikerült megfelelő válasszal visszatérni

5 Backend

5.1 Keretrendszer és könyvtárak

A Node.js aszinkron futású, esemény vezérelt platformra épülő Express keretrendszerben került fejlesztésre a http kéréseket kiszolgáló backend. A mongoose nevű package segítségével csatlakozik a backend az adatbázishoz, valamint végez rajta módosításokat. Környezeti változók olvasását a dotenv végzi, a jelszó titkosítást pedig, a korábbiakban említett CryptoJS. A fizetést a Stripe biztosítja, amely rendelkezik test móddal, ezért olyan felhasználói élményt nyújt, mintha valóban fizetni tudnánk az oldalon, továbbá magával hordozza az evolúciós lehetőséget, hogy egyszer elinduljon a weboldal, mint valódi webshop. JWT, teljes nevén jsonwebtoken segítségével végződik az autentikáció és a token-képzés és legutolsó sorban a cors, egy biztonsági modul, amely megakadályozza, hogy a rosszindulatú szoftverek hozzáférjenek bármilyen erőforráshoz.

5.2 Útválasztás

A backend szerveret indít a 3000-s localhost porton, majd várja a különböző végpontokon a beérkező API kérések kiszolgálását. Ezen route-ok definiálása egy Express objektum függvényeivel lehetségesek. Ilyen függvény például a get vagy a post, amelyek mindig várnak egy string-et, hogy meghatározzák az „útvonalat” és egy függvényt, amivel visszatudnak térni az adott kérés meghívásakor. Gyakori megoldás, hogy ezeket a visszatérési függvényeket controllerek végzik és beimportálva ezt hívják meg a route-ok, azonban mi a route-on belül írtuk meg ezeket az aszinkron funkciókat.

Vegyük példának az egy adott terméket lekérdező kérést, amelyben a `http://localhost:3000/api/products/find/{_id}` útvonal visszatér az url-ben feltüntetett egyedi azonosítóval megegyező termék mezőivel.

```
const router = require("express").Router();
router.get("/find/:id", async (req, res) => {
  try {
    const product = await Product.findById(req.params.id);
    res.status(200).json(product);
  } catch (error) {
    res.status(500).json(error);
  }
});
```

6: Egy termék lekérdezése

Minden útvonal rendelkezik egy saját végponttal a könnyedebb átláthatóság érdekében. Ezeket az index.js állományban határoztuk meg. Így minden felhasználókkal kapcsolatos kérés a „localhost:3000/api/users” kezdetű végponton lesz elérhető, amelyhez adódik még hozzá az express get, post, delete vagy put függvényében meghatározott útvonal. A backend 6 route-al, valamint egy verifyToken.js-el rendelkezik, amely a token verifikálók és az autentikáló függvényeket tartalmazza a tisztább kód érdekében. Ezen megoldásnak köszönhetően nem szükséges minden route-nál külön token ellenőrző function-t írni, hanem elég meghívni az előbb említetteket, ezzel leegyszerűsítve a kódot.

```
app.use("/api/users", userRoute);
app.use("/api/auth", authRoute);
app.use("/api/products", productRoute);
app.use("/api/carts", cartRoute);
app.use("/api/orders", orderRoute);
app.use("/api/checkout", stripeRoute);
```

7: Route-ok

```
const verifyToken = (req, res, next) => {
  const authHeader = req.headers.token;
  if (authHeader) {
    const token = authHeader;
    jwt.verify(token, process.env.JWT_SECRET, (err, user) => {
      if (err) res.status(403).json("Token is not valid!");
      req.user = user;
      next();
    });
  } else {
    return res.status(401).json("You are not authenticated!");
  }
};
```

8: A tokenellenőrző függvény

5.2.1 Megugrandó feladatok

Felsorolunk néhány említésre méltó függvényt, amelyek akár komplikáltságukkal, akár újdonságukkal kisebb nehézséget okoztak a fejlesztés során. Az egyik a Stripe fizetést lebonyolító funkció, amelyet nem ismertünk, de egyszerűségének, felhasználóbarátságának köszönhetően nem okozott nagy nehézségeket.

Ide soroljuk a bejelentkezés függvényét, főként a titkosított, illetve a dekódolt jelszó közti átváltások miatt. A hash-elést követően ráadásképp a jsonwebtoken-től kapott token is helyet kellett kapjon a function-ben, ezzel minimálisan megbonyolítva azt.

Különlegesnek találjuk a statisztikákat biztosító függvényt is, hiszen számos egyedi *MongoDB* operátort alkalmazunk benne, kihasználva a szoftver adta lehetőségeket. Ilyenek például a `setMonth()` és a `lastMonth()` metódusok, valamint a `$gte`, `$project`, `$group` és `$match` operátorok.

```
router.get("/income", verifyTokenAndAdmin, async (req, res) => {
  const date = new Date();
  const lastMonth = new Date(date.setMonth(date.getMonth() - 1));
  const previousMonth = new Date(new Date().setMonth(lastMonth.getMonth() - 1));

  try {
    const income = await Order.aggregate([
      { $match: { createdAt: { $gte: previousMonth } } },
      {
        $project: {
          month: { $month: "$createdAt" },
          sales: "$amount",
        },
      },
      {
        $group: {
          _id: "$month",
          total: { $sum: "$sales" },
        },
      },
    ]);
    res.status(200).json(income);
  } catch (error) {
    res.status(500).json(error);
  }
});
```

9: OrderStats függvény

5.3 Adatmodellek

Az adatmodellek feladata, hogy kapcsolatot teremtsenek az adatbázis és a backend vezérlők közt. Úgynevezett schema-kat tartalmaznak, amelyek meghatározzák az adatbázis és a backend közt folyó objektumok mezőit, azok típusát és feltételeit. Webshopunk négy adatmodellel rendelkezik, csupán azzal, a korábbiakban említett különbséggel, hogy a vezérlők helyett az útvonalak közt lettek megvalósítva a végpontok logikái. A model schema-kra példa látható az 5. ábrán.

5.4 Tesztelés

A backend tesztelésére, akárcsak az API kérésekhez, a *Thunder Client*-et használtuk. Ezen tesztek eredményei megtalálhatóak a repositoryban, a *thunder-client_apitest.json* állományban, amely beimportálását követően magában a kliensben is megtekinthetővé válik.

6 Frontend

6.1 Keretrendszer, könyvtárak, fejlesztői környezet

A frontend feladata megjeleníteni egy weboldal vagy internetes alkalmazás felhasználói felületét kliensoldalon. Fontos a megfelelő megjelenítés, a látványvilág, az egyszerű, de érthető kommunikáció és a könnyed kiigazodás. Ezek hiányában a domain-re tévedő internetfelhasználók negatív élmennel gazdagodnak látogatásuk során, amely nagyban befolyásolja döntésüket például egy vásárlásnál. A fizikai boltok, szolgáltatások közül is hasonló tényezők miatt voksol az egyik vagy másik mellett az ügyfél, nemhogy a virtuális alternatívák közt.

Mi keretrendszerként react-ot választottunk, hiszen hatékony strukturális felépítése könnyen módosíthatóságot, kezelhetőséget biztosít. Népszerűsége és frissessége miatt sok anyagot, forrást találni, amelyek segítségével nem okozott gondot a nyelv elsajátítása, annak ellenére, hogy az iskolai órák keretén belül nem tanuljuk. A react kódokat .jsx-ben, egy bővített javascript nyelvben írják, amelyben elérhetőek a HTML alkotóelemei.

Az alapvetően is effektív react fa-struktúrát tovább egyszerűsítettük azzal a megoldással, hogy styled-componenteket alkalmaztunk, ami azt jelenti, hogy a stílusolt, formázott HTML tag-eket konstansokként hoztuk létre és azzal építettük fel komponenseinket. Ehhez szükségünk volt a *vscode-styled-components* extension-re. Egy styled component és létrehozása:

`<Container>`

```
const Container = styled.div`
  height: 60vh;
  background-color: lightgray;
  display: flex;
  align-items: center;
  justify-content: center;
  margin-top: 10px;
  flex-direction: column;
  border-top: 1.5px solid black;
`;
```

10. ábra: styled-components

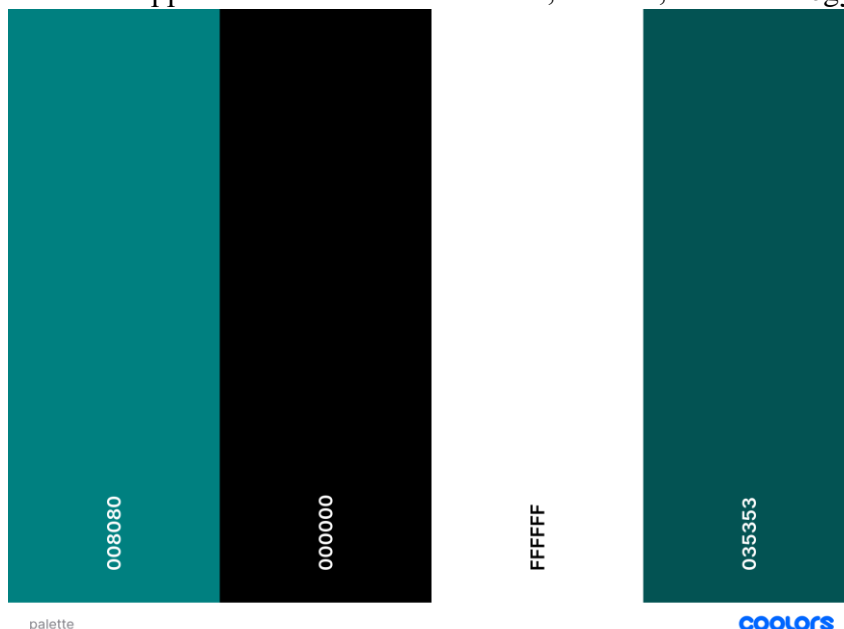
A webshop sok állapotváltozásának kezeléséhez a *react-redux* toolkitet használtuk, amely lehetővé teszi, illetve megkönnyíti ezen módosítások logikai lekövetését. Gyakori változást egy ilyen oldalnál főként a kosár tartalma és a felhasználók jelentik. Percenként többször kilépünk, belépünk fiókokba, új termékeket helyezünk be és ki kosarunkból.

Megszokott ikonjainkat a *@material-ui/icons* modul segítségével importáltuk be. Mindenféle, közel 3 ezer ilyen icon érhető el ennek a package-nek köszönhetően, amelyek nagyban megkönnyítik a fejlesztést, illetve megelőzik a különböző grafikai munkákat, ha valaki precíz weboldalt szeretne létrehozni.

A http kéréseket és az arra kapott válaszok transzformálását az *axios* promise-based könyvtára végzi, lényegében ez felelős az API-val való kommunikációért. Az útválasztást a frontendben a *react-router* metódusaival hajtjuk végre.

6.2 Fellesztés menete, komponensek, oldalak

Rendelkeztünk némi elképzeléssel már a frontend megkezdése előtt is, azonban első lépésként a színvilág tisztázását végeztük. Mindenképpen szerettünk volna feketét, fehérét, valamint egy közepes kékszerű árnyalatot az arculatnak, de rábíztuk magunkat a *colors.co* nevű palettakészítő weboldalra. A végeredmény az ábrán látható színekavalkád lett. Úgy gondoljuk hűen tükrözik a színek a letisztultságot és azt a gamer érzést, amelyet nyújtani akarunk ügyfeleinknek.



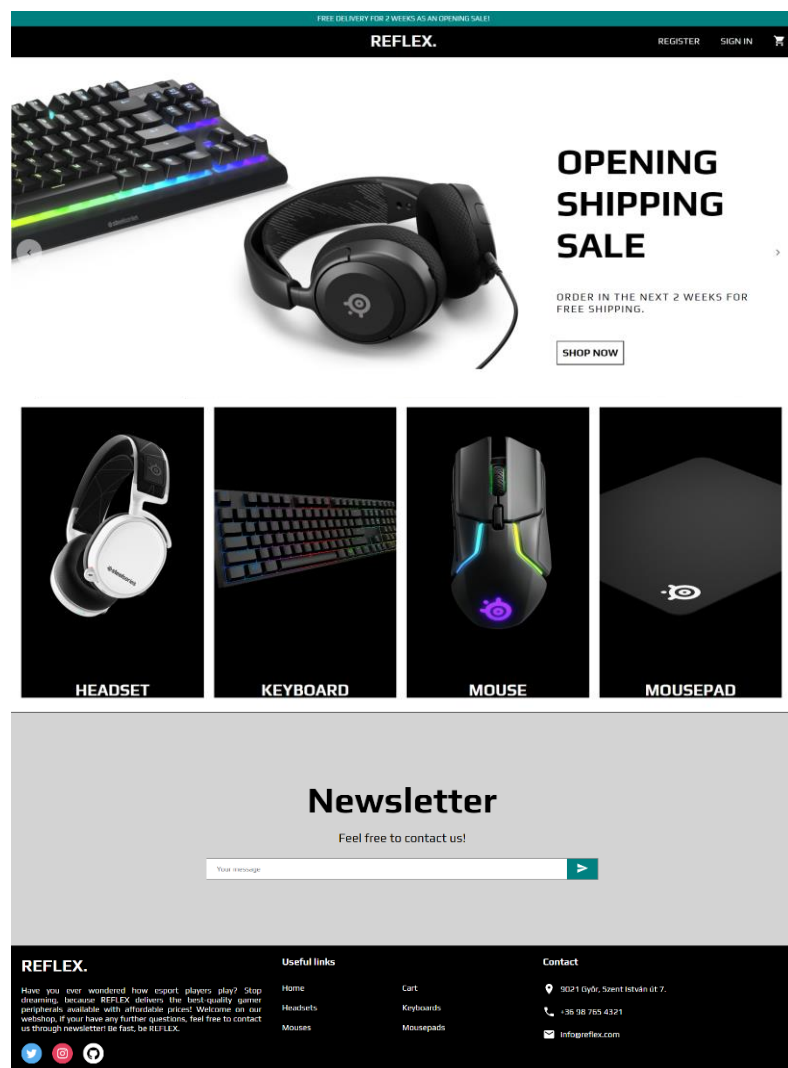
11. ábra: Színpaletta

Képszerkesztési ismereteinket kamatoztatva logót szerkesztettünk a weboldalnak, amelybe próbáltuk beleintegrálni a REFLEX. fantázianév minél több betűjét. Favicon-ként és a Stripe fizetés felületén jelenik meg egyelőre. Ezt követően adatgyűjtés következett, az általunk forgalmazott termékek összeszedését végeztük, képekkel és minden egyéb mezővel együtt, hogy a lehető legfrissebb adatokkal tudjuk kínálni azokat.



12. ábra: REFLEX. logo

Felmértük milyen oldalakra lesz szüksége a webshopnak és megkezdtuk a főoldal, a Home.jsx fejlesztését. Tudniillik a react page-k komponensekből állnak, ez biztosítja a könnyű struktúrát. A korábbiakban említett styled-components-ekkel hoztuk létre magát a komponenseket is, így kialakítva a lehető lekezelhetőbb felületet. A tetején látható egy bejelentő,

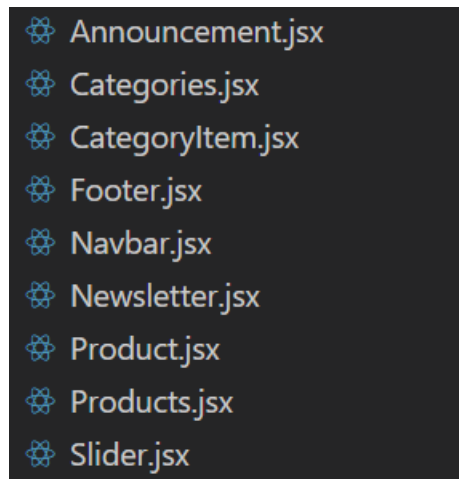


13. ábra: A főoldal

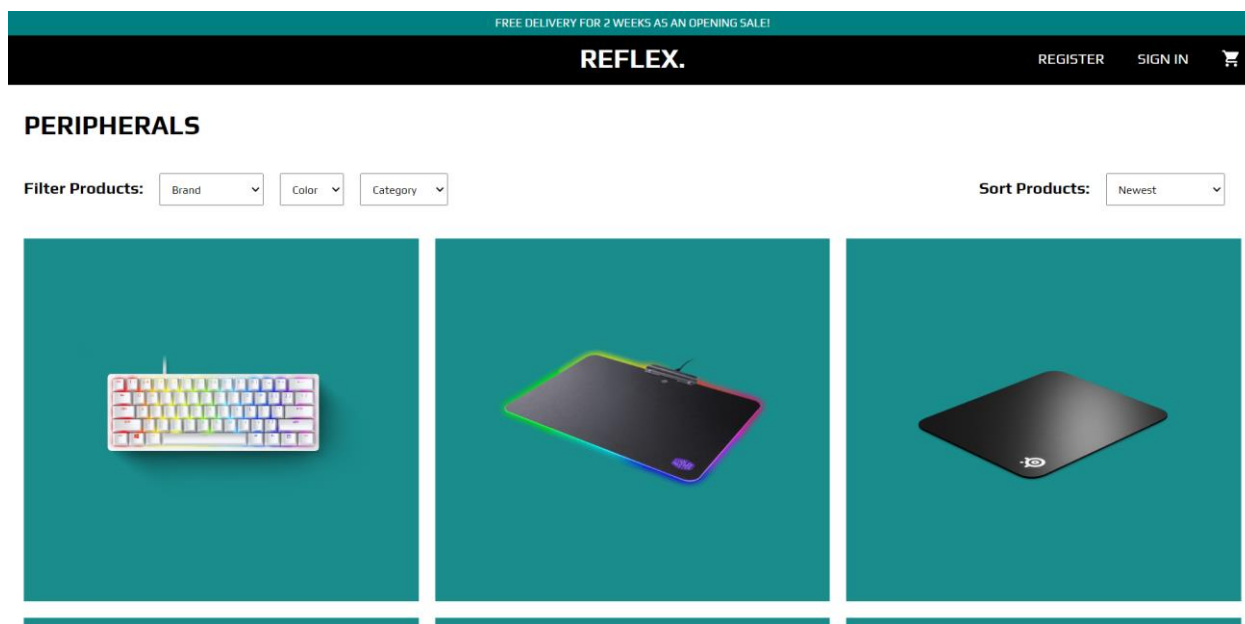
amely szerkesztésekor bármikor értesíthetjük a vásárlót egy vonzó ajánlatról. Jelenleg nyitási akcióként ingyenes szállítást kínálunk. Alatta egy navbar található regisztrációhoz, belépéshez, kosárhoz vezető gombokkal. Tovább görgetve egy hirdetőtáblát találunk, amelyet lapozva több promóciót is találni. Lentebb egyből rászűrhetünk az általunk célzott kategóriára a választó segítségével, levelet küldhetünk egy input mezőből, majd legalul a footer, amelyről a social media felületekre és a különböző oldalakra ugorhatunk a linkkelt feiratokkal. Bejelentkezés után a register és a sign in fül eltűnik, felváltja egy logout gomb, bal oldalt a navbaron pedig,

megjelenik a bejelentkezett felhasználó neve. Ez további fejlesztési lehetőségeket vetít előre, a jövőben tervezzük kialakítani az „Én profilom” oldalt, ahol a user szerkesztheti a saját adatait, jelszót módosíthat, stb.

A már meglévő komponensek segítségével végtelenül könnyebb és gyorsabb volt a többi oldal létrehozása. Beimportálásukkal, egyszerű styled-componentként beilleszthettük a következő elkészítendő oldalra. A terméklista oldal volt a következő, ahova a slider és a kategóriaválasztót kivéve minden meglévő komponenst fel tudtuk használni. Legfelül találjuk a szűrőket és a rendezési lehetőséget. Filtert állíthatunk be a termék gyártójára, a típusára és elérhető színére, továbbá rendezhetjük a megjelenésük és áruk szerint.

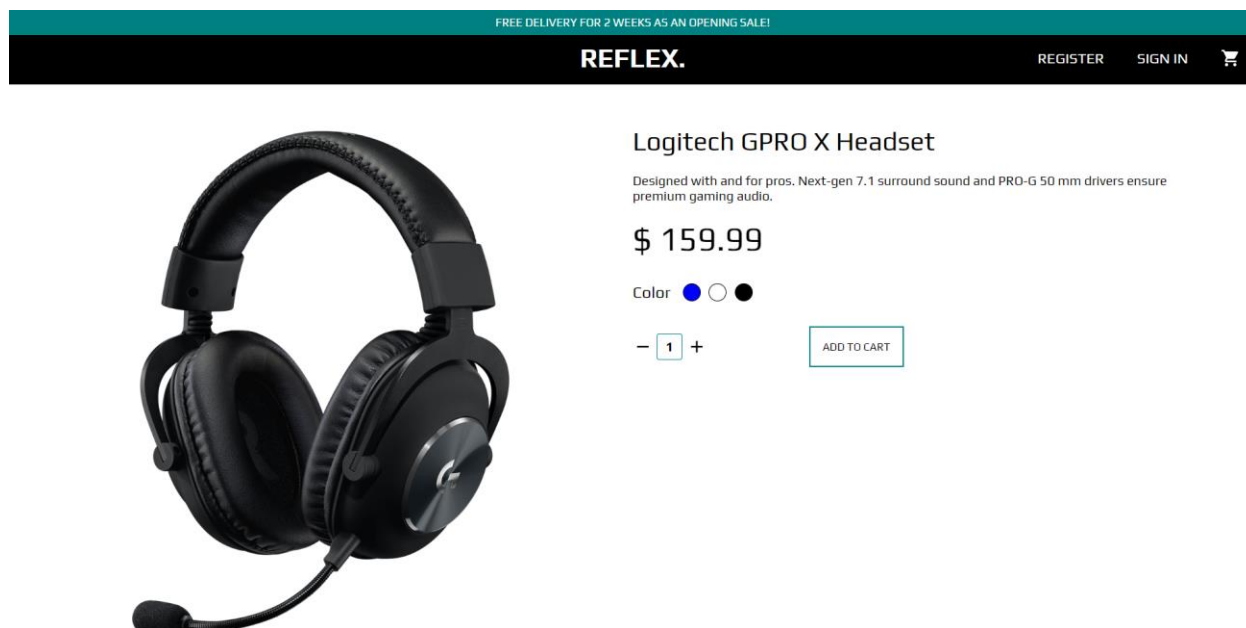


14. ábra: Komponensek



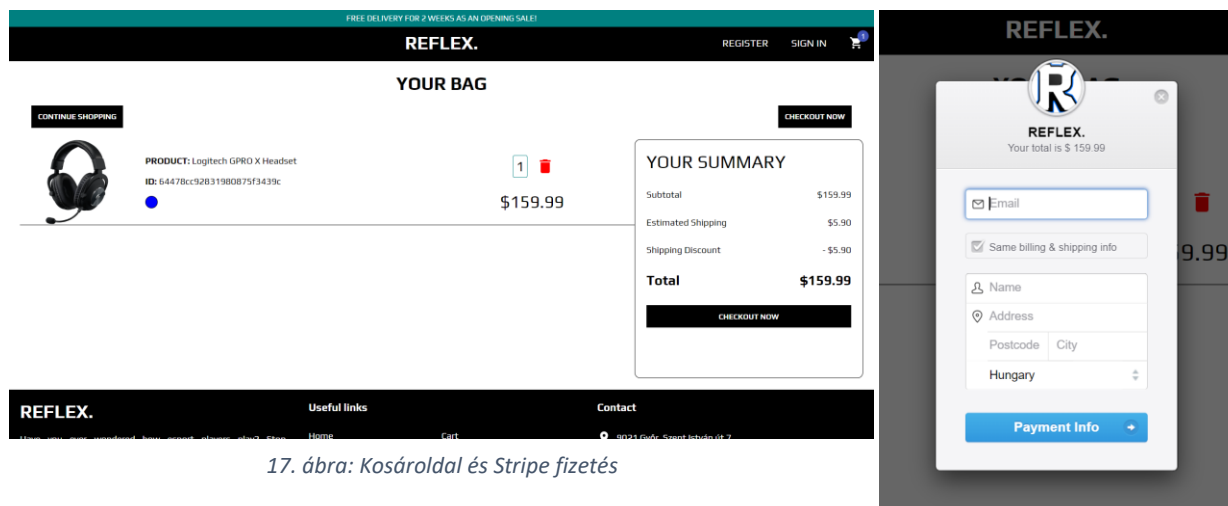
15. ábra: Termékkereső

Egy termékre kattintva megjelenik a termék saját oldala, ahol elolvashatjuk leírásukat, megtekinthetjük árukat és választhatunk az elérhető színek közül. Növelhetjük, csökkenthetjük 1-ig a mennyiséget és belehelyezhetjük a kosárba termékünket.



16. ábra: Termékdoldal

Ezt követően a kosárololdal lett létrehozva, ahol láthatjuk termékeinket összesítve, azok mennyiségeit és az összeadott árukat a summary blokkban. A termékek totális árát egy külön totalprice elnevezésű függvény számolja, annak érdekében, hogy lekövethető legyen további termékek hozzáadása és az általunk meg nem rendelendő, kosárból kivett termékek kivonása. Erre a piros material-ui kuka ikonnal van módunk, amely kitörli az összes azonos terméket, legyen az bármilyen színű vagy bármekkora mennyiség. Bejelentkezés után ki is tudjuk fizetni a Stripe checkout test mode-jának segítségével. Természetesen ez csak, a már említett, nem éles rendszerrel. Következőleg visszajelzést kapunk a sikeres rendelésről, majd megkapjuk az order egyedi azonosítóját is, amelyet már, a sikeresen fel post-olt rendelésből kapunk vissza.



17. ábra: Kosárololdal és Stripe fizetés

Végül, de nem utolsó sorban a regisztráció és bejelentkezés oldalát készítettük el, amely háttéréiben gamer stock fotók találhatóak. Sikeres regisztrációt követően a bejelentkezés oldalára érkezünk meg, ahol beléphetünk az új profilunkba.

Ezen oldalakat végül az App.jsx-be vannak beimportálva és styled componentként használva, ahol a *react-router* végzi az útválasztást, a „http://localhost:3000/api/„ bázis url utáni végpontok megadását követően.

```
const App = () => {
  const user = useSelector((state) => state.user.currentUser);
  return (
    <Router>
      <Switch>
        <Route exact path="/">
          <Home />
        </Route>
        <Route path="/products/:category">
          <ProductList />
        </Route>
        <Route path="/product/:id">
          <Product />
        </Route>
        <Route path="/cart">
          <Cart />
        </Route>
        <Route path="/success">
          <Success />
        </Route>
        <Route path="/login">{user ? <Redirect to="/" /> : <Login />}</Route>
        <Route path="/register">
          {user ? <Redirect to="/" /> : <Register />}
        </Route>
      </Switch>
    </Router>
  );
};
```

18. ábra: App.jsx részlet

6.3 Redux

Központi állapotkezelőként a react-redux paketet használtuk, ennek segítségével tároljuk a kosárban lévő termékeket, illetve mentjük payloadba az aktuálisan bejelentkezett user adatait. A redux-persist-nek köszönhetően oldalfrissítés után is benn marad a felhasználó, nem lépteti ki az

```
const cartSlice = createSlice({
  name: "cart",
  initialState: {
    products: [],
    quantity: 0,
    total: 0,
  },
  reducers: {
    addProduct: (state, action) => {
      const product = state.products.find((product) => product._id === action.payload._id)
      if (product) {
        product.quantity += action.payload.quantity;
      } else {
        state.quantity += 1;
        state.products.push(action.payload)
      }
    },
    removeItem: (state, action) => {
      state.products = state.products.filter(
        (products) => products._id !== action.payload
      );
      state.total = action.payload.price * action.payload.quantity;
      state.quantity -= 1;
    },
    resetCart: (state) => {
      state.products = [];
      state.quantity = 0;
      state.total = 0;
    },
  },
});
```

19. ábra: cartRedux.js

oldal. Ugyanez igaz a kosár tartalmára is. A cartRedux.js adja át a rendelés post-nak a termékek adatait, így jön létre az order táblában a megrendelés. Az ezekben a javascript fájlokban létrehozott függvényeket hívják meg a különböző célú gombok, például az addProduct() vagy a removeItem().

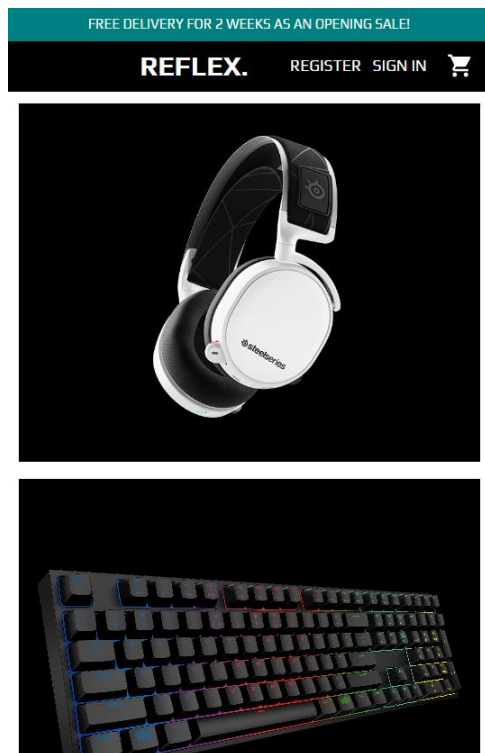
7 Mobilnézet

Napjaink felgyorsult világában internetes vásárlásainkat sokszor telefonon végezzük, ezért mindenképp fontosnak találtuk a mobilnézet kialakítását. A legnépszerűbb weboldalnak mobil applikációjuk van, azonban még az oldal kezdetekor úgy gondoljuk nem kötelező lépés egy ilyen lefejlesztése. Ami viszont elhanyagolhatatlan, az a reszponzivitás létrehozása, hogy okostelefonokról is kellemes felhasználói élményt nyújthassunk. 430 px szélességű képernyőig határoltuk be a telefonos nézetet. Ez szinte minden okostelefonnál nagyobb, így nem fog problémát okozni a különböző eszközök használata közben.

Ehhez egy külön létrehozott, props-okat váró responsive.js fájlban található mobile metódus, amelyet a komponensekbe és az oldalakba beimportálva megadhatjuk minden styled-component mobilnézeti változásait.

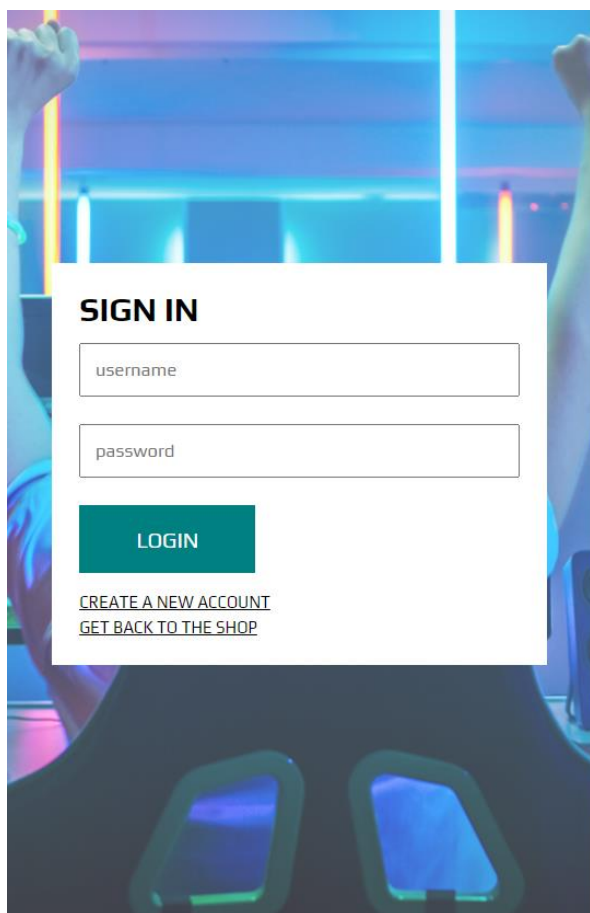
```
export const mobile = (props) => {  
  return css`  
    @media only screen and (max-width: 430px) {  
      ${props}  
    }  
  `;  
};
```

20. ábra: Mobile függvény



21. ábra: Mobilnézet főoldala

A főoldalon érezhető a legjelentősebb változás, ott a hirdetőtábla sliderje nem jelenik meg, hanem egyből a kategóriáknál találják magukat a felhasználók. Soros osztás helyett oszlopos elrendezésben találhatók meg a választások. Más mobilon a footer, azonban nem jelent nagy változást, csupán nem láthatunk pár oldallinket. Az asztali nézethez hasonlóan elirányíthatjuk magunkat a bejelentkezés és a regisztráció oldalára.



SIGN IN

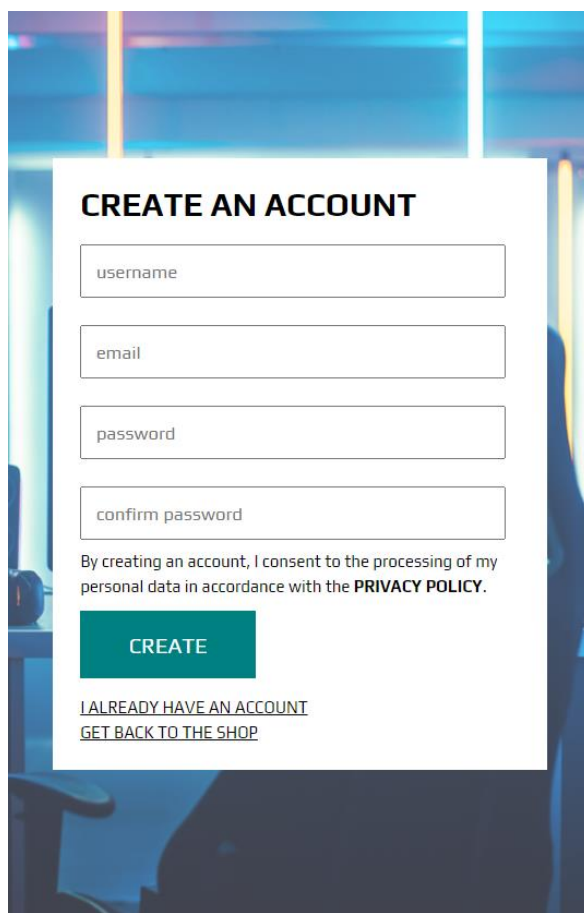
username

password

LOGIN

[CREATE A NEW ACCOUNT](#)
[GET BACK TO THE SHOP](#)

22. ábra: Bejelentkezés



CREATE AN ACCOUNT

username

email

password

confirm password

By creating an account, I consent to the processing of my personal data in accordance with the **PRIVACY POLICY**.

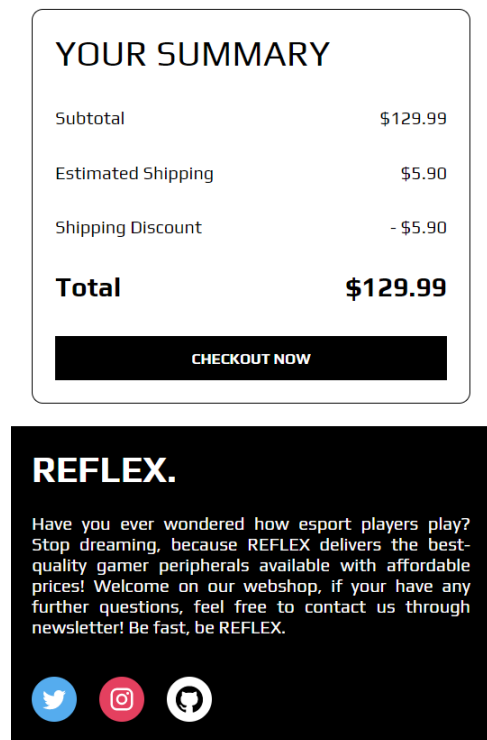
CREATE

[I ALREADY HAVE AN ACCOUNT](#)
[GET BACK TO THE SHOP](#)

23. ábra: Regisztráció

Regisztrálhatunk és bejelentkezhethetünk a két fenti felületen. Regisztrációkor ügyelnünk kell arra, hogy helyesen írjuk be az email címet, jelszavunk meghaladja a 8 karaktert és a jelszó megismétlésekor beírt string megegyezzen jelszavunkkal. Csupán ennyi kritériuma van az új felhasználó létrehozásának, amelyek elmulasztása esetén egy alert üzenet emlékeztet minket a követelményekre. Sikeres regisztráció után az oldal átirányít minket a login page-re.

A terméklistában és a kosárban sem láthatunk nagy változást, kivéve természetesen, hogy a méretek a telefon szélességének megfelelőek, és a weben egymás mellett elhelyezkedő elemek, például a szűrők, függőleges rendezéssel rendelkeznek. Bejelentkezés után ugyanúgy, a megszokott módszerrel fizethetünk, hasonlóan responsive-an. Köszönhetjük ezt a stripe mobilfizetési támogatásának felületének, amely legalább annyira felhasználóbarát, mint az asztali verzió.



24. ábra: Kosár summary

7.1 Tesztelés

A visual studioban helyi hálózaton történő elindítást követően telefonról is elérhetővé válik az oldal. Ezt használtuk ki a mobilról való tesztelés lebonyolításához. Két különböző készüléken is megnyitottuk az oldalt, ellenőrizve, hogy bármilyen méretű okostelefon képernyőjén szépen fest a frontend. Továbbá sikeresen végigfutottuk azokat a lépéseket, amelyeket számítógépen. Tesztelve lett a regisztráció, a bejelentkezés, a kijelentkezés, a termékek kiválasztása, törlése a kosárból, valamint a fizetéssel történő rendelésleadás is.

8 Felhasználói kézikönyv

REFLEX. Reakcióidőt jelent, azt az intervallumot, amely egy adott impulzus lereagálásához szükséges az embernek. A szó hallatán a gyorsaságra asszociálunk, termékeink ezt a precizitást próbálják tükrözni. Az alábbiakban körültekintő leírást olvashatnak weboldalunkhoz.

8.1 Authentikáció

A register fülön létre tud hozni új felhasználói fiókot egy felhasználónév, egy email és egy jelszó kétszeri megadásával. Az adatkezelésre vonatkozó szabályokban feltüntetett minden pontot elfogad a create gombra kattintással. Egy kérés akkor lehet sikeres, ha megfelelő formátumú email címet és legalább 8 karakter hosszú jelszót ad meg. Ezt követően fiókja bekerül az adatbázisba, amibe bármikor bejelentkezhet login felületünkön. Sikeres bejelentkezés után felhasználóneve megjelenik a navbar-on, valamint egy új, log out gombot is látni fog, amellyel kiléphet fiókjából. Termékeink közt bármikor böngészhet kijelentkezve, azonban vásárolni csakis autentikálva lehet.

8.2 Kapcsolat

Newsletter felületünkön, illetve a footerben is megtalálható email címen és telefonszámon bármikor kapcsolatba léphet velünk, munkaidőben(8:00-16:00). Felelőseink órákon belül kapcsolatbalépnek az ügyféllel, amennyiben megkeresése elsőre sikertelen.

8.3 Termékböngészés

Termékeink közt bármikor, bármilyen készülékről keresgélhet. A böngészést megkönnyíti a 3 különböző szűrő és a rendezést állító filter. Az alábbi opciókat veheti igénybe:

- Brandre: Logitech, SteelSeries, Cooler Master, Razer, Xtrfy
- Színre: Fekete, Fehér, Kék, Piros
- Kategóriára: Headset, Keyboard, Mouse, Mousepad
- Rendezés: Újdonságok, Csökkenő ár, Növekvő ár

A perifériára kattintva ráugorhat a termék oldalára, ahol részletes leírást kaphat termékünkről, megtekintheti árát, majd mennyiséget növelve, kiválaszthatja színét és belehelyezheti a kosárba.

8.4 Kosár

A kosárba lépve látja odahelyezett termékeit, azok mennyiségét és árát. Összesítő fülünkön láthatja az összárát, amelyet a kosárban lévő perifériák megvásárlása jelentene. A piros szemetesikonnal bármikor kitörölheti a listából az adott termék összes darabját. Amennyiben be van jelentkezve és fizetni szeretne a checkout now gombbal meg is teheti ezt. Ha bejelentkezése lejárt, vagy még nem regisztrált, a gomb átirányítja a megfelelő végpontra, hogy authenticálhassa magát. Vásárlását könnyedén folytathatja a continue shopping gombbal.

8.5 Fizetés

Megfelelő feltételek teljesülését követően a Stripe checkout segítségével kifizetheti rendelését. Meg kell adnia email címét, szállítási és számlázási címet, amennyiben ezek nem egyeznek. Ezt követően beviheti bankkártya adatait. Jelenleg még csak ez az egyetlen fizetési metódus érhető el oldalunkon, de a jövőben tervezzük bővíteni kínálatunk. Sikeres payment-et követően megkapja rendelésszámát és visszatérhet a főoldalra, ekkor már biztosra veheti, rendelését rögzítettük.

9 Üzemeltetés

A szoftver megtalálható a REFLEX mappában, azon belül szétszítva backendre és frontendre. Visual Studio Code-ban indítva két terminált szükséges nyitni, az egyikben a `cd backend`, a másikban a `cd frontend` parancsot kiadva lépünk bele a mappákba. Ezt követően adjuk a frontendnél adjuk ki az `npm install --force`, backendnél az `npm install` commandot.

Ezt követően mindkettő elindíthatjuk az `npm start` parancssal és némi idő, illetve pár success válasz után elindul a weboldal a 3000-s localhost porton.

Admin panel hiányában jelenleg még nem kezelhetőek a user-ek és a product-ok grafikai felületen, azonban fejlesztés alatt áll ezek kidolgozása.

10 Összegzés

Eleinte három fővel, nagyobb ambíciókkal vágott neki a csapat a projektmunkának, mint ahogy a végén befejezni sikerült. István feladatának megvalósítása, az admin fül, nem készült el, ezt majd, a további fejlesztések során kívánjuk elkészíteni, amennyiben döntünk a projekt folytatásáról. Új dolgokkal ismerkedtünk meg a feladatba csöppenve, amelyet ugyan nehézségként éltünk meg, de büszkén és jókedvvel végeztünk el. Sem Ármin, sem Bálint nem a informatikapiacra tervezi el jövőjét, az évek során kifejezetten más irányt vett érdeklődésünk. Ennek ellenére úgy érezzük, minden Tőlünk telhetőt megtettünk és kiadtunk magunkból, hogy a lehető legpraktikusabb projektet létrehozassuk.

A csapatmunka segített személyes barátságunk szorosra fűzésében is, amelyet úgy gondoljuk folytatni fogunk. Kommunikációnk az elejétől fogva kifogástalanul működött, támogattuk egymást a feladatokban. A munkamegosztás is megfelelően zajlott, mindegyikünk belerakta tudásának minden részletét és vette a fáradságot újak elsajátítására.

A webshop fejlesztési lehetőségeivel tisztában vagyunk és a korábbiakban említett fejezetekben taglaltuk miként tudna eladhatóvá és piaciilag értékelhetővé válni a REFLEX.

Bár az kiderült, hogy nem mi leszünk a google cégcsoport újgenerációs fejlesztői, azt gondoljuk a REFLEX.-re és erre a technikumi évre örömmel fogunk visszatekinteni életünk során, akár csak a közös munkára, vagy a nehézségekre.

11 Forrás

NodeJS - <https://nodejs.dev/en/learn/>

Express - <https://expressjs.com/>

MongoDB - <https://learn.mongodb.com/>

Thunder Client - <https://www.thunderclient.com/>

dotenv - <https://www.npmjs.com/package/dotenv>

jsonwebtoken - <https://jwt.io/>

CryptoJS - <https://cryptojs.gitbook.io/docs/>

Stripe - <https://stripe.com/docs/videos>

Google Fonts - <https://fonts.google.com/>

Axios - <https://axios-http.com/docs/intro>

Styled-components - <https://styled-components.com/docs/tooling>

React - <https://legacy.reactjs.org/docs/getting-started.html>

Material-icons - <https://mui.com/material-ui/material-icons/>

React-router - <https://reactrouter.com/en/main>

Redux - <https://redux.js.org/>

Redux-persist - <https://blog.logrocket.com/persist-state-redux-persist-redux-toolkit-react/>

W3Schools - <https://www.w3schools.com/>

StackOverflow - <https://stackoverflow.com/>

Bólya Gábor Tanár Úr és Soós Gábor Tanár Úr órai forrásai - 2023