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) neve[1]:	Polocz Máté, Szilágyi Márk, Nyári Gábor
Képzés:	nappali
Szak:	5 0613 12 03 Szoftverfejlesztő és tesztelő technikus

A záródolgozat címe:

Okosgolf webshop

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 lap[2]

	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. Bevezetés

Manapság már minden automatizálva, vagy le egyszerűsítve van számunkra. Miért ne lehetne a kedvenc hétvégi időtöltésünket, a golf, is egyszerűbbé téve, hogy még az otthonunk kényelméből vagy éppen játék közben is elérhetővé válhasson az éppen lejátszott játék gyors eltárolása vagy annak vissza nézése egy weboldalon keresztül. Mellé egy működő webshop funkcióval bármikor a recepcióhoz rendelhet bármit az ajánlataink közül, ha netán a pályán van vagy éppen otthon gondolkozik azon, hogy ellátogat a golfpályánkra.

Az ötletünk egy fesztiválnak való contactless alkalmazásnak indult, ami a fesztiválon belüli fizetésben és eligazodásban segít QR kód segítségével, viszont ez átalakult egy golfpálya karbantartására alkalmas webhellyé, ami lehetővé teszi a különböző adatok, mint például a játszott játékok, tárolását és meglévő menük egyszerű használatát minden látogató számára. Ezen kívül egy beépített webshoppal rendelkezik, amely egyszerűbbé teszi a látogatók vagy a személyzet számára, hogy az elérhető árukat nyomon kövessék. A webes regisztráció után player tagságot kapunk, ami a pálya recepcióján manuálisan átírható, ha a személy más feladatot tölt be a pálya életében. A jövőben a webshop használata ki fog teljesülni egy működő kosárral és az illetőhöz rendelt pénztárcával, amit egy külön menüpont alatt kell feltölteni.

2. A csapat

2.1. A csapat tagjai és feladataik

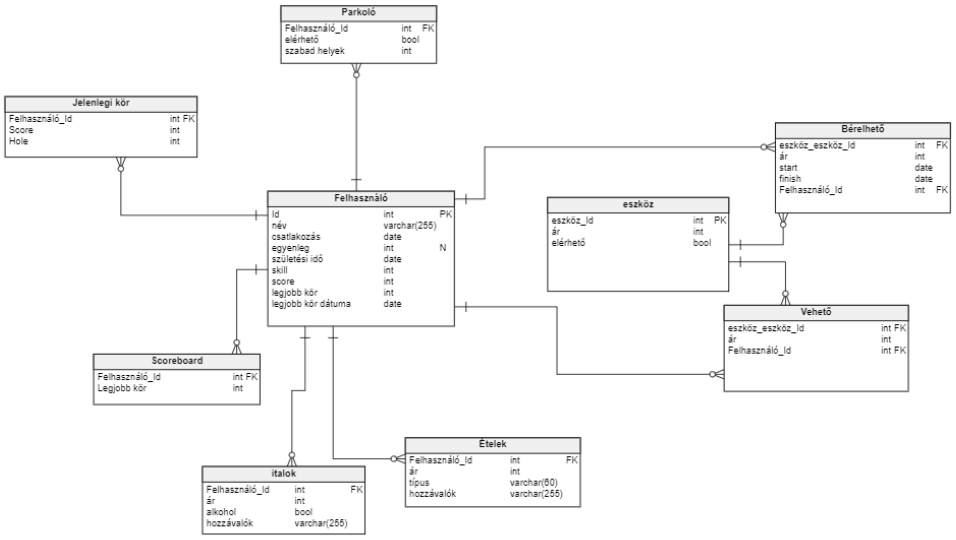
A csapat visszamenőleg szeptemberben alakult meg három fővel, Nyári Gábor, Szilágyi Márk és Polocz Máté. Volt a csapaton belül kételkedés az egész évvel kapcsolatban, megéri-e maradni vagy inkább ki kéne iratkozni, de összeszedve az akaraterőnket mindegyikűnk úgy döntött marad és ezzel megszilárdította a csapatszellemet. Még akkor nem tudva, hogy ki milyen részét fogja képviselni a kész programnak, de ez idővel kibontakozott azzal, hogy ki melyik részét szereti jobban a program készítésének. Így lett a backend Gábor és Máté által megírva, a frontend, a kevés rászánt idő miatt, pedig a hármunk munkája, melynek nagy részét Márk tette ki. Így kényelmesen elkészülve a programmal a beadási határidőre és nem stresszelni a projekt nem leadásával.

2.2. A csapatmunka megvalósítása

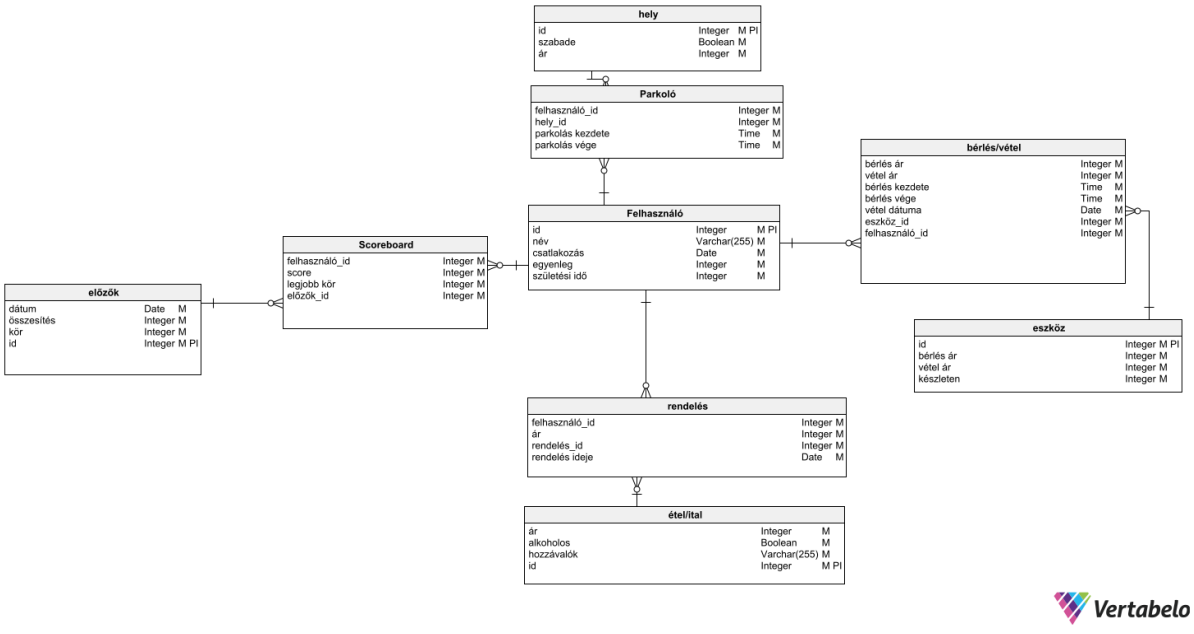
A csapatmunkát GitHubon és Discordon keresztül oldottuk meg, ezen kívül az iskolában is folytatva munkásságainkat. Szerencsére a programok megbízhatósága miatt problémába nem ütközünk használatuk közben, bár, ha mégis lett volna hiba, akkor a projektet minden nagyobb átírás után elmentettük egy úgynevezett biztonsági mentés mappába biztosítva működését, abban biztosak lehetünk, hogy következő nagyobb projekt létrehozásában is fogjuk ezeket a programokat használni. Tervezés szempontjából, ahogy haladtunk az évvel egyre feltűnőbb lett, hogy mit akarunk használni programozási nyelv szempontjából és olyan február környékén eldöntöttük azt, hogy a backend Node JS-be lesz írva és a hozzá csatolt frontend pedig a Vue JS technológiáját fogja használni. Volt egy szakasz, ahol egy másik backenden gondolkoztunk, de az alapból választott program használatát jobban átvettük az iskolában, így maradt az a választás. A Vueból való tanulmányok megszilárdították a nyelv használatát, nem volt kérdéses annak használata. Ahogy közeledtünk a beadási határidő felé, annál gyakrabban gyűltünk össze az osztályteremben és Discordon is, hogy időben sikerüljön leadni a munkánkat, ezek a megbeszélések az idő elteltével hosszabbak és munkával teljesebbek lettek lehetővé téve a projekt befejezését, bár nem mindig sikerültek a megbeszélések összehangolása, de mindenki megcsinálta az az előtti gyűlésen való kitűzött célját.

3. Adatbázis

3.1.Az adatbázis diagramja



1. ábra 1.0 verzió



2. ábra A végleges, amiről dolgoztunk

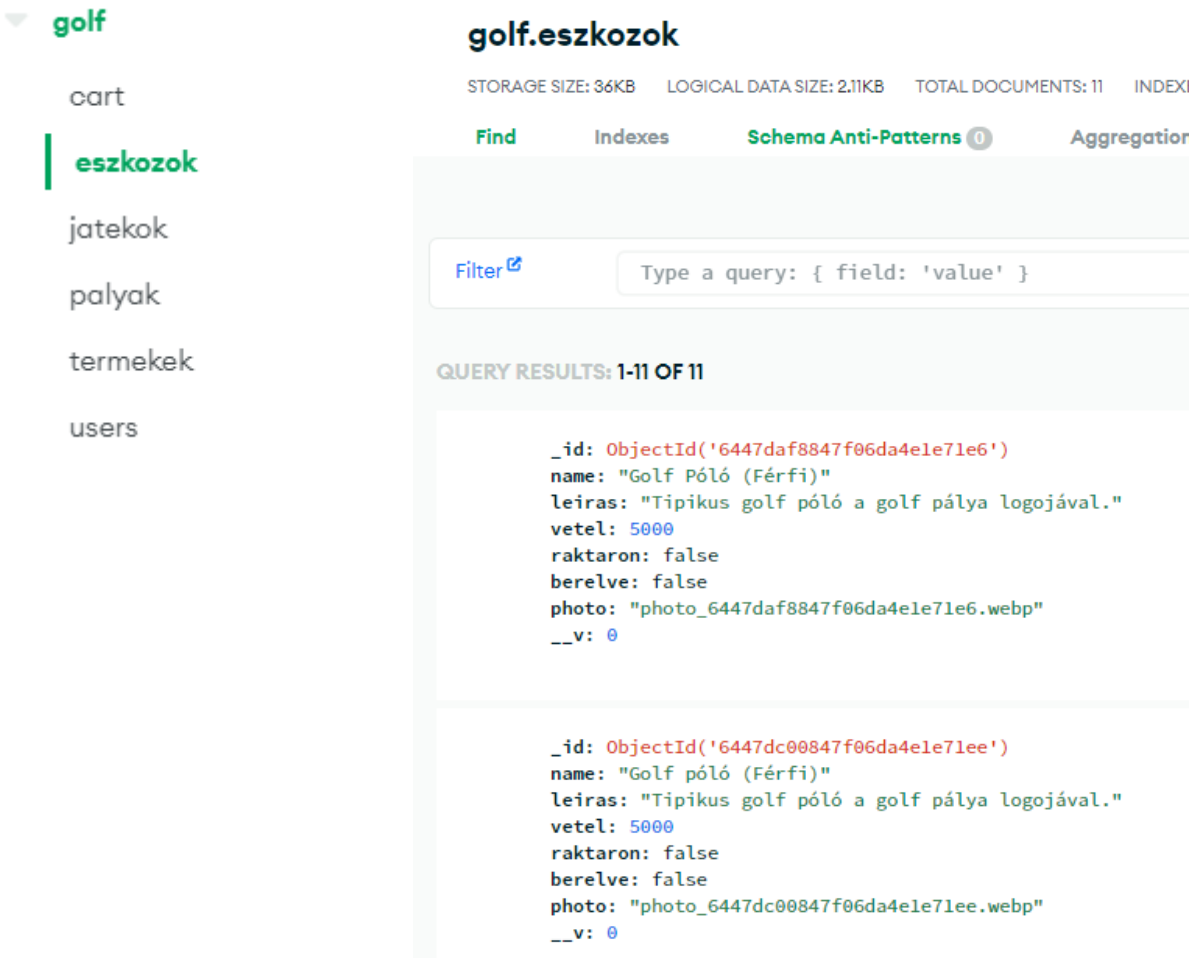
3.2.Az adatbázis leírása, magyarázata

Az adatbázist az elején SQL rendszerben terveztük megoldani, így a hozzá készült ábrák annak használatát mutatják, viszont az idő elteltével és tudásunk gyarapodásával változtattunk

eredeti terveinken, és áttértünk egy NoSQL alapú cloud szerű adatbázisra MongoDB néven így megkönnyítve az adatbázis készítését.

3.3. Az adatbázis háttér technológiája, adatbáziskezelő program

MongoDB-n belül elhagyhattuk a tradicionális táblázatokat és bonyolult Primary és Foreign Key összekötéseket, csak egy úgynevezett kollekciót kellett létrehozni, majd annak összekötése a másik kollekciókkal nem igényelt mást, mint annak beírása a meghatározott kollekcióba.



4. Rest API és a backend

4.1. Áttekintés

Ez a dokumentáció az okosgolf weboldal által használt JSON alapú API használatát mutatja be. A kéréseket a fejlesztési fázisban egy a 3000-es porton futó Node.js webszerver fogadta, így a példákban a végpontok „http://localhost:3000/” 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:3000/api/auth/login
```

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

{

”email”:”12@gmail.com”,

”password”:”123546”

}

Erre válasz:

{

"success": true,

"token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjY0MzgwZDZlNTQ0NzZjMTZjZjEyMTAzNCIsImlhdCI6MTY4MjU3OTU4NSwiZXhwIjoxNjkwMzU1NTg1fQ.04Vz7yidHiMYg6acsyFuKi2dzR89xj8WZmn8nphvz9Y"

}

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

Authorization: Bearer {{TOKEN}}

Ez a Bearer {{TOKEN}} minden login után megváltozik és csak az aktuálisan bejelentkezett felhasználó adatai kerülnek feldolgozásra.

1. 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.

1. 4.3.1 Felhasználók

4.3.1.1 Felhasználó lekérdezése

GET http://localhost:3000/api/auth/me

Content-type: application/json
Accept: application/json
Accept-Charset: utf-8
Authorization: Bearer {{TOKEN}}

Ezt a GET kérést elküldjük a szerver felé, ezt a választ kapjuk vissza:


```
{
  "success": true,
  "data": {
    "_id": "64380d6e54476c16cf121034",
    "name": "admin wannabe",
    "email": "12@gmail.com",
    "role": "admin",
    "wallet": 12,
    "createdAt": "2023-04-13T14:10:54.886Z",
    "__v": 0,
    "keepLoggedIn": false,
    "photo": "../public/default-avatar.png",
```

"resetPasswordExpire": "2023-04-25T19:52:22.645Z",

"resetPasswordToken":"4fcd07c81fd2106707ef87a1f26d605c580f452a0b6656dd2adfdab9a a1098c8"

}

}

A fentiekben említett Bearer {{TOKEN}} felhasználásával íratjuk ki a felhasználó adatait.

Mező	**Típus**	**Leírás**
success	bool	A lekérdezés működését jelzi.
data	string tömb	A felhasználó adatai.
_id	number	A felhasználó id-je.
name	string	A felhasználó neve.
email	string	A felhasználó email címe.
role	string	A felhasználó szerepköre.
wallet	number	A felhasználó pénztárcája (nincs használva). Lehet null.
createdAt	date	A felhasználó létrehozásának ideje.
keepLoggedIn	bool	Bejelentkezve tartás más ablakon is (nincs használva)
photo	string	A felhasználó profilképére mutató link. Alap képnek egy default-avatar kép.
resetPasswordExpire	date	A felhasználói jelszó reset token idejének lejárata.
resetPasswordToken	token	A felhasználó jelszó reset tokenje.

4.3.1.2 Felhasználó regisztrálása és ellenőrzése

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

Content-type: application/json
Accept: application/json
Accept-Charset: utf-8
Authorization: Bearer {{TOKEN}}

Ezzel a kéréssel tudunk regisztálni az oldalra:

{

"name" : "player",

"email" : "1@gmail.com",

"role" : "player",

"password" : 123456,

"wallet": 12

}

Válasz:

{

"success": true,

"token":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjY0NGEzMzdiOWY2MWQ5MDkyNWVmM2U4YiIsImlhdCI6MTY4MjU4NDQ0NCwiZXhwIjoxNjkwMzYwNDQ0fQ.t5AqKBSGrMr64eNivqQOaDyuVTlYu27wsjUDK27m_Ok"

}

Ezt ellenőrizni, egy POST login kérés után a GET me kéréssel tudjuk

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

{

”email”:”1@gmail.com”,

”password”:”123546”

}

Válasz:

{

"success": true,

"token":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjY0NGEzMzdiOWY2MWQ5MDkyNWVmM2U4YiIsImlhdCI6MTY4MjU4NDQ0NCwiZXhwIjoxNjkwMzYwNDQ0fQ.t5AqKBSGrMr64eNivqQOaDyuVTlYu27wsjUDK27m_Ok"

}

Ezt a tokent, amit visszakapunk a belépés után használjuk fel a Bearerbe, hogy mindenhol tudjuk authentikálni az éppen bejelentkezett usert.

GET http://localhost:3000/api/auth/me

Content-type: application/json
Accept: application/json
Accept-Charset: utf-8
Authorization: Bearer {{TOKEN}}


```
{
  "success": true,
  "data": {
    "_id": "644a337b9f61d90925ef3e8b",
    "name": "player",
    "email": "1@gmail.com",
    "role": "player",
    "wallet": 12,
    "photo": "default-avatar.png",
    "keepLoggedIn": false,
    "createdAt": "2023-04-27T08:34:03.909Z",
    "__v": 0
}
```


Ez ha biztosan tudni szeretnénk a regisztrációt és hogy létrejött-e a felhasználó, a Mongodb szerveren a users collection alatt a felhasználónak így kell kinézni:


```
_id: ObjectId('644a337b9f61d90925ef3e8b')
name: "player"
email: "1@gmail.com"
role: "player"
password: "$2a$10$2urrLZRBJyLTVN4DYKpl4.6QCI.CRWNySEl4USjvb0fjI7lMhiZGG"
wallet: 12
photo: "default-avatar.png"
keepLoggedIn: false
createdAt: 2023-04-27T08:34:03.909+00:00
__v: 0
```

A POST regiterben megadott adatok:

Mező	Típus	Leírás
name	string	A felhasználó neve.
email	string	A felhasználó email címe.
role	string	A felhasználó szerepköre.
password	string	A felhasználó jelszava.
wallet	number	A felhasználó pénztárcája (nincs használva). Lehet null.

4.3.1.3 Felhasználó szerkesztése

Regisztrált felhasználó adatait szerkeszteni a következő végpontra intézett kéréssel lehet:

PUT http://localhost:3000/api/auth/updatedetails

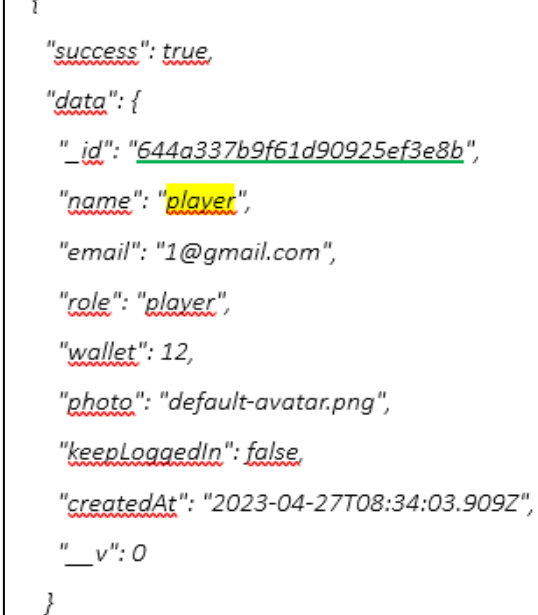
A felhasználó itt tudja megváltoztatni felhasználó nevét és email címét.

Content-type: application/json
Accept: application/json
Accept-Charset: utf-8
Authorization: Bearer {{TOKEN}}

```
{
  "name":"játékos 1"
}
```

Válasz:

```
{
  "success": true,
  "data": {
    "_id": "644a337b9f61d90925ef3e8b",
    "name": "játékos 1",
    "email": "1@gmail.com",
    "role": "player",
    "wallet": 12,
    "photo": "default-avatar.png",
    "keepLoggedIn": false,
    "createdAt": "2023-04-27T08:34:03.909Z",
```



"__v": 0

}

}

POST http://localhost:3000/api/auth/forgot

A felhasználó egy mailtrappes email cím segítségével kap egy kódot, amit a frontend egy input mezőjébe be kell írnia, mellé meg kell adni az új jelszavát, majd a ”Get a new password” gombra kattintani.

Content-type: application/json
Accept: application/json
Accept-Charset: utf-8
Authorization: Bearer {{TOKEN}}

{

"email":"1@gmail.com"

}

Válasz:

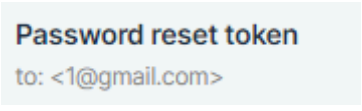
{

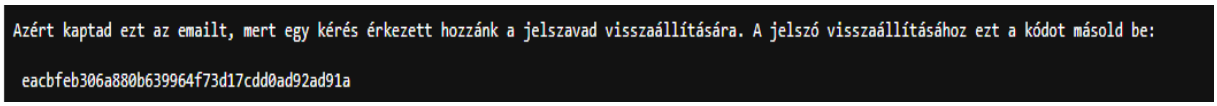
"success": true,

"data": "Email elküldve"

}

Mailtrappen a következő emailt kapjuk:





Ennek a kódnak a bemásolásával, majd vele együtt az új jelszó elküldésével megváltoztathatjuk az elfelejtett jelszót.

Egy Status: 200 OK válasszal

{

"success": true,

"token":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjY0NGE3ZGE5MTI1MjFmNDgxOGE2NmRkZiIsImlhdCI6MTY4MjYwNzAwMCwiZXhwIjoxNjkwMzgzMDAwfQ.ySuYzucEPwMig0YYSw3hmEivDBDBvMAY3VFD4lqAz6I"

}

és a verifyoláshoz járó tokennel, bár ez frontenben nem így lett megoldva.

2.4.3.2 Eszközök

4.3.2.1 Eszköz felvitele

Eszközt csak admin tud felvinni az adatbázisba, admin rolet pedig csak kézzel beállítva lehet adni.

POST localhost:3000/api/eszkozok

Content-type: application/json
Accept: application/json
Accept-Charset: utf-8
Authorization: Bearer {{TOKEN}}

{
"name" : "golf labdák",
"leiras": "A golf pálya saját márkájú labdája. 5db / pack.",
"vetel" : 6000
}

Válasz:

{
"success": true,
"data": {
"name": "golf labdák",
"leiras": "A golf pálya saját márkájú labdája. 5db / pack.",
"vetel": 6000,
"raktaron": true,
"berelve": false,
"photo": "DefaultEszkoz.jpeg",
"_id": "644b66caed5bc783178e0fb9",
"__v": 0
}
}

Mező	**Típus**	**Leírás**

success	bool	A lekérdezés státuszát jelzi (true/false).
data	string tömb	Az eszköz modellből beolvasott modell alapján feltöltve.
name	string	Az eszköz neve.
leiras	string	Az eszköz leírása.
vetel	number	Az eszköz vételi ára.
raktaron	bool	Azt jelzi, hogy az adott termékből van-e még raktáron (nincs használva). Alap értéke true.
berelve	bool	Azt jelzi hogy az adott eszköz ki van-e bérelve (nincs használva). Alap értéke false.
photo	string	Az eszközök default képére mitató linket tartalmazza.
id	string	Az eszköz id-je.

4.3.2.2 Eszközök lekérdezése

Az összes eszköz lekérdezése egy GET kéréssel történik amit bárki meg tud tenni, aki be van jelentkezve. Ennek a GET kérésnek elküldése után:

```
GET localhost:3000/api/eszkozok
```

Ezt a választ kapjuk:


```
{
  "success": true,
  "count": 12,
  "data": [
    {
      "_id": "6447daf8847f06da4e1e71e6",
      "name": "Golf Póló (Férfi)",
      "leiras": "Tipikus golf póló a golf pálya logojával.",
      "vetel": 5000,
      "raktaron": false,
      "berelve": false,
```

```
    "photo": "photo_6447daf8847f06da4e1e71e6.webp",
    "__v": 0
  },
  {
    "_id": "6447dc00847f06da4e1e71ee",
    "name": "Golf póló (Férfi)",
    "leiras": "Tipikus golf póló a golf pálya logojával.",
    "vetel": 5000,
    "raktaron": false,
    "berelve": false,
    "photo": "photo_6447dc00847f06da4e1e71ee.webp",
    "__v": 0
  }
}
```

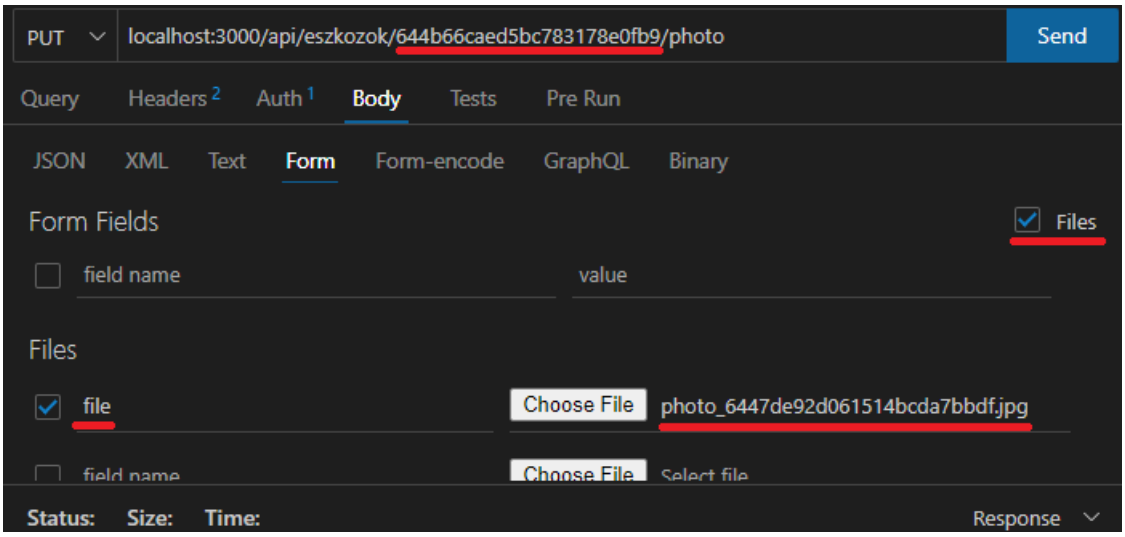

Mező	**Típus**	**Leírás**
success	bool	A lekérdezés státuszát jelzi (true/false).
count	number	Megadja, hogy hány eszköz van a data tömbben.
data	string tömb	Az eszköz modellből beolvasott modell alapján feltöltve.

4.3.2.3 Eszközhöz kép társítása

Ezt a PUT kérést csak staff és admin rolet betöltő emberek tudják elérni. A kérés a következő: Példának a fentiekben felvitt golflabdákat fogom használni:

```
PUT localhost:3000/api/eszkozok/644b66caed5bc783178e0fb9/photo
```

Content-type: application/json
Accept: application/json
Accept-Charset: utf-8
Authorization: Bearer {{TOKEN}}



Majd erre a kérésre ezt a választ kapjuk:


```
{
  "success": true,
  "data": "photo_644b66caed5bc783178e0fb9.jpg"
}
```


Ennek sikerességét a Mongodb eszkozok collection részében találjuk meg:


```
_id: ObjectId('644b66caed5bc783178e0fb9')
name: "golf labdák"
leiras: "A golf pálya saját márkájú labdája. 5db / pack."
vetel: 6000
raktaron: true
berelve: false
photo: "photo_644b66caed5bc783178e0fb9.jpg"
__v: 0
```


Itt a data az adott eszközhöz rendelt képet mutatja, míg a success pedig a folyamat sikerességét jelzi.

4.3.2.4 Eszköz törlése

Eszközt csak admin tud törölni. 2 módja van rá vagy a frontendben egy gomb segítségével vagy kézzel. A következő DELETE kérés az utóbbit mutatja, példának a golflabdák lesznek felhasználva:

```
DELETE localhost:3000/api/eszkozok/644b66caed5bc783178e0fb9
```

Content-type: application/json
Accept: application/json

Accept-Charset: utf-8
Authorization: Bearer {{TOKEN}}

Erre kapunk egy választ:

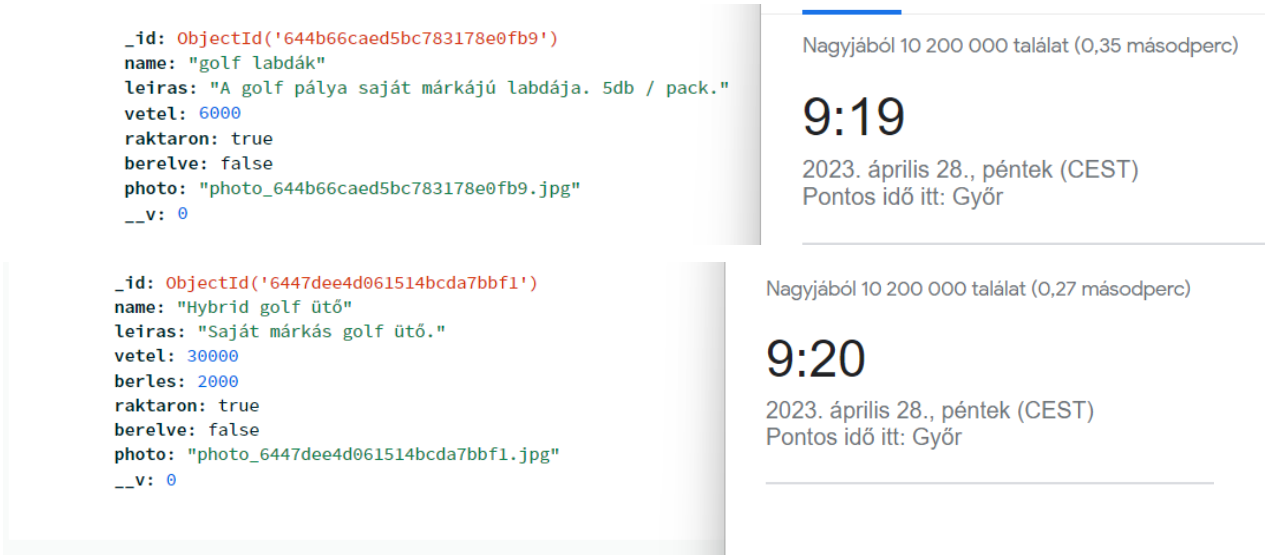
{

"success": true,

"data": {}

}

Ezt ismét a Mongodb collection fogja mutatni:



Ezek a Mongodb utolsó rekordjai az eszkozok collectionön bellül. Ezen kívül lehet egy GET kérést küldeni ami id alapján szűr:

GET localhost:3000/api/eszkozok/644b66caed5bc783178e0fb9

Content-type: application/json
Accept: application/json
Accept-Charset: utf-8
Authorization: Bearer {{TOKEN}}

Erre a kérésre ezt a választ kapjuk:

{

"success": false,

"msg": "Not found"

}

Ezzel végleg meggyőződve, hogy töröltük a collectionből.

3. 4.3.3 Termékek

4.3.3.1 Termékek felvitele

Felvitelt ismét csak admin tud, a következő képp:

POST localhost:3000/api/etelekitalok

Content-type: application/json
Accept: application/json
Accept-Charset: utf-8
Authorization: Bearer {{TOKEN}}

```
{
  "name":"példa",
  "price":"600",
  "ingrediencie":"só, bors, paprika"
}
```

Válasz:

```
{
  "success": true,
  "data": {
    "name": "példa",
    "price": 600,
    "etel": true,
    "alcohol": false,
    "photo": "DefaultTermek.png",
    "_id": "644b7650701c74a88cb318f2",
    "__v": 0
  }
}
```

Mező	**Típus**	**Leírás**
success	bool	A kérés sikeressége.
data	string tömb	az etelital modelből beolvasott adatok alapján.
name	string	A termék neve.
price	number	A termék ára.
etel	bool	Étel és ital megkülönböztetése (nincs használva).

alcohol	bool	Alkoholos termékek megkülönböztetése (nincs használva).
photo	string	A default termék képre mutat.
_id	string	A termék id-je.

4.3.3.2 Termék update

Update kérést csak staff vagy admin tud végrehajtani. Példának a fent készített rekordot fogom használni.

PUT localhost:3000/api/etelekitalok/644b7650701c74a88cb318f2

Content-type: application/json
Accept: application/json
Accept-Charset: utf-8
Authorization: Bearer {{TOKEN}}

```
{
  "name":"updated"
}
```

Erre válasz:

```
{
  "success": true,
  "data": {
    "_id": "644b7650701c74a88cb318f2",
    "name": "updated",
    "price": 600,
    "etel": true,
    "alcohol": false,
    "photo": "DefaultTermek.png",
    "__v": 0
  }
}
```

Mező	**Típus**	**Leírás**
success	bool	A kérés sikeressége.
data	string tömb	Az etelital model alapján.

4.4.3.4 Játékok

4.3.4.1 Játék felvitel

Játék felvitelhez csak bejelentkezés után tud feltölteni a felhasználó. A következő POST kérés segítségével:

```
POST localhost:3000/api/jatekok
```

Content-type: application/json
Accept: application/json
Accept-Charset: utf-8
Authorization: Bearer {{TOKEN}}

```
{
  "score":"5",
  "szam":"6447e449d061514bcda7bc46"
}
```

Válasz:

```
{
  "success": true,
  "data": {
    "user": {
      "_id": "64380d6e54476c16cf121034",
      "name": "admin wannabe",
      "email": "12@gmail.com",
      "role": "admin",
      "wallet": 12,
      "createdAt": "2023-04-13T14:10:54.886Z",
      "__v": 0,
      "keepLoggedIn": false,
      "photo": "../public/default-avatar.png",
      "resetPasswordExpire": "2023-04-25T19:52:22.645Z",
      "resetPasswordToken":
"4fcd07c81fd2106707ef87a1f26d605c580f452a0b6656dd2adfdab9aa1098c8"
    },
    "szam": "6447e449d061514bcda7bc46",
    "playedAt": "2023-04-28T07:55:22.421Z",
    "score": 5,
    "_id": "644b7bea701c74a88cb318fb",
    "__v": 0
  }
}
```

Mező	Típus	Leírás
success	string	A kérés sikeressége.

data	string tömb	A jatek model beolvasva.
user	string tömb	A user model beolvasva.
szam	string	A pálya id-je.
playedAt	date	A rögzítés időpontja.
score	number	A pályán elért pontszám.
_id	string	A játék id-je.

4.3.4.2 Játék törlése

Játékot csak admin tud törölni. A következő DELETE kérés segítségével:

```
DELETE localhost:3000/api/jatekok/644b7bea701c74a88cb318fb
```

Content-type: application/json
Accept: application/json
Accept-Charset: utf-8
Authorization: Bearer {{TOKEN}}

Erre válasz:


```
{
  "success": true,
  "data": {
    "_id": "644b7bea701c74a88cb318fb",
    "user": "64380d6e54476c16cf121034",
    "szam": "6447e449d061514bcda7bc46",
    "playedAt": "2023-04-28T07:55:22.421Z",
    "score": 5,
    "__v": 0
  }
}
```


Ezt egy GET kéréssel user id alapján ellenőrizzük:

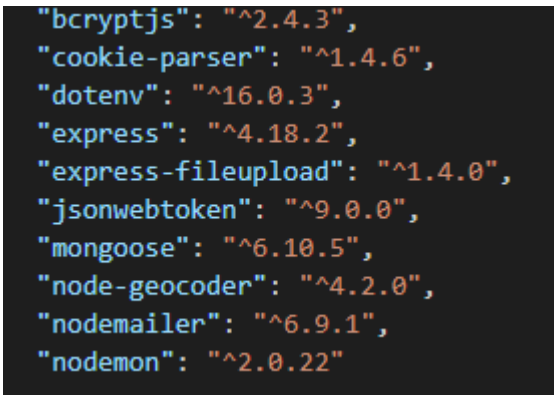
```
GET localhost:3000/api/jatekok/userjatek/ 64380d6e54476c16cf121034
```


```
{
  "success": true,
  "data": []
}
```


Ezzel meggyőződünk a törlés sikerességéről.

2. 4.4. A fejlesztés eszközei, szoftverek, fejlesztői környezetek

A fejlesztésre a fentiekben megemlített Node Js-t használtuk Visual Studio Codeon belül, több fajta kiegészítővel, beleértve az expresst, mongooset, nodemont, dotenvet, JWT-t, és annak cookieait, bcryptjs kódolását és a NodeMailert is.



*SEQ ábra * ARABIC 5. ábra Backendben használt kiegészítők*

3. 4.5. A fejlesztés menetrendje, mérföldkövek

Kicsi lépésekben haladva kezdtük el a backend fejlesztését, hiszen ez az első nagyobb projektünk, amit nem szabad elsietni. Kiindulási pontnak kiválónak valósult az órán tanult alapanyagok, amely egy szilárd, biztos alapot adott a programunknak és sikeresen tudtunk belőle tovább építeni. Először elkezdve a modellek felépítésével és azok tartalmával, amihez a MongoDB mongoose kiegészítőjét használtuk megkönnyítve az adatbázisban való kollekciók létrehozását, majd célként kitűzve a további kérések létrehozását, amit egy API tud. Kezdve a GET kéréssel, ami mondhatni könnyedséggel sikerült, nem volt nehéz az alapját megérteni és egyről a kettőre le is tudtuk kérni a még manuálisan feltöltött adatbázis adatait. Innentől kezdve végig mentünk minden alap API kérésen, folytatva a POST-al, hogy már ne keljen manuálisan betölteni a kollekciókat. Utána a PUT és DELETE kéréseket is sikerült bele rakni a programunkba. Ezek készen létével azonosítás adtunk hozzá a projekthez, ami a JWT token rendszert használja, ez lehetővé tette, hogy felhasználókat hozzunk létre, akik úgynevezett Bearer tokent kapnak. Ennek segítségével a meglévő útvonalakat le tudtuk védeni, hogy csak is bejelentkezett személyek férhessenek hozzá a meglévő tokenekkel. Erre építve ha már vannak felhasználók, akkor már lehessen emailt is küldeni nekik, így került a programba az új jelszó kérése ami egy Reset Tokent küld Emailben, amivel a jelszó megváltoztatása lehetséges, ebben segítségül a mailtrap weboldalát vettük igénybe.

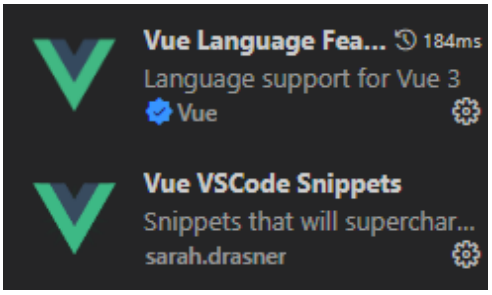
4. 4.6. A fejlesztés fontosabb megoldandó problémái

A Backend fejlesztése közben, ami nagyobb problémát okozott az a MongoDB-hez tartozó populate függvény. Nehéz volt megoldani, hogy a felhasználót keresse, ha a bevásárló kosarat keresi vagy, hogy a játékok részében a pályákat szám szerint keresse meg ne ObjectId alapján,

de végül arra nem lett megoldás, ezért kell a frontend részén az úgy nevezett speciális pályakódot megadni, mert máshogy nem lehetett megoldani. Ennek ellenére annyi problémánk nem akadt programmal, amit ne tudtunk volna az internet vagy tanáraink segítségével megoldani.

5. A frontend fejlesztése

5.1. A fejlesztés eszközei, szoftverek, fejlesztői környezetek



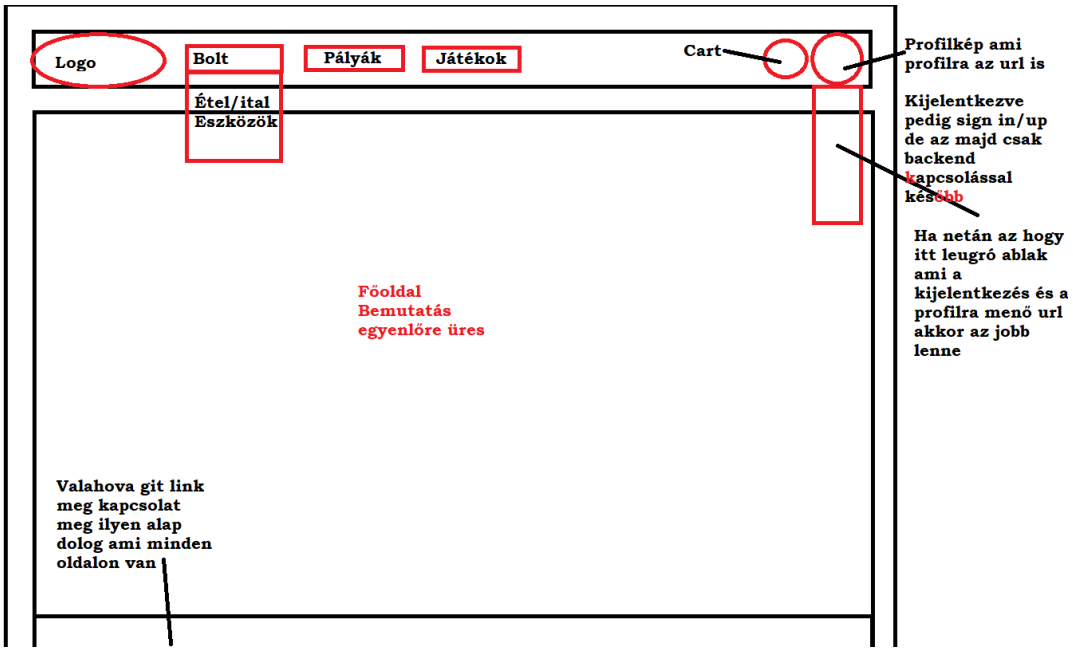
A frontendet az elejétől fogva Vueban terveztük megoldani, hiszen nem tűnt nehéznek a használata Visual Studio Codeon belül. A hozzá ajánlott programok letöltését, Vue VSCode Snippets és a Vue Language, követően a meglévő HTML tudásunkat felhasználva készítettük el a weboldal kinézetét. A backendel való kommunikálásra Axios promise-based HTTP klienst használtunk.

5.2. A fejlesztés menetrendje, mérföldkövek

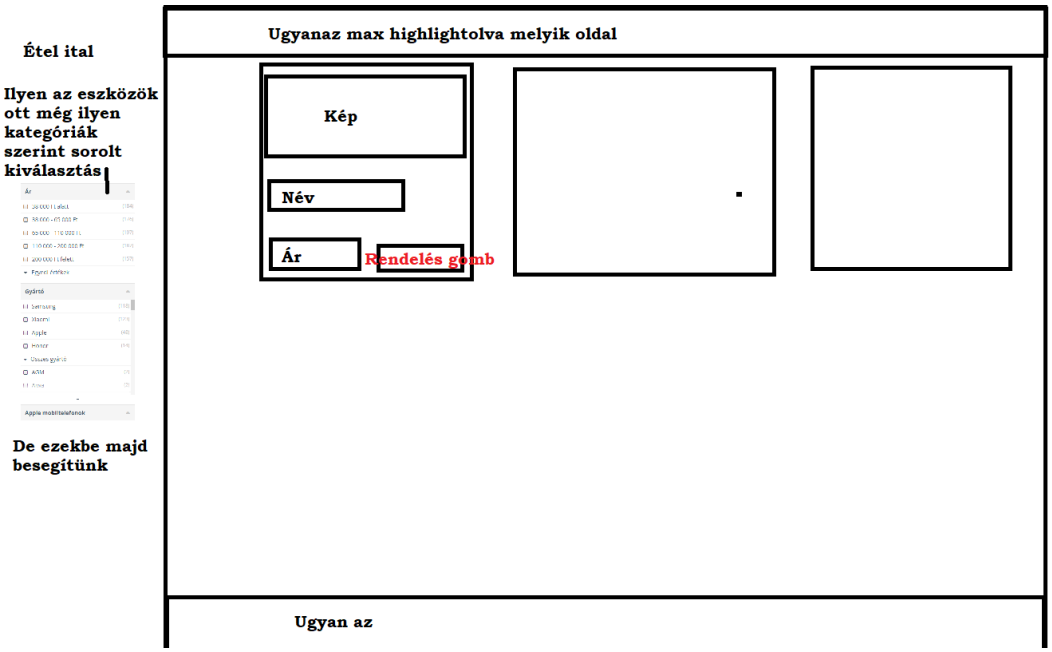
A fejlesztés a weboldal navigációs felső részével kezdődött, amit TailWind CSS segítségével megfelelő kinézetre formáltunk és ezzel elkezdve az oldal kinézetének megvalósítását. Először elkezdve a bejelentkezés részével, hogy biztosra menjünk, hogy a backend sikeresen tud kommunikálni a frontendel és egy kiindulási pontot adjon a továbbiakban. Viszont a TailWind nem igazán volt jó választás, mert nagyban ütközik a másik CSS kiegészítővel, a Bootstrappel, úgyhogy úgy döntöttünk, hogy elhadjuk a TailWindet és csakis Boostrappet használva haladunk tovább, így előről kezdve az egész programot. Szerencsére az elején voltunk és nem vesztettünk vele sok időt. A weboldalak előre tervezett rajzok alapján lettek tervezve, napról napra több és több elkészülve.

A backendel való kommunikálást a bolti részen sikerült először megoldani, hogy le tudja kérni az adatbázisban eltárolt kollekciókat. A POST kérés után, amit nehéz volt megoldani, szinte egyről a kettőre könnyen meg lehetett oldani a backendel való összes kérést. Ezzel egyhuzamba mindenki kivette a részét a frontend kinézetének és mechanikájának jobbá tevésében és egy jól megbeszélt munka körforgás lépett fel, ahol mindenki a magának kiírt

feladatot időben megoldott, így összehangolva a csapatmunkát a kevés idő, amit hagytunk a frontendre mondhatni majdnem elég is lett arra, hogy egy működő alkalmazást hozzunk létre.



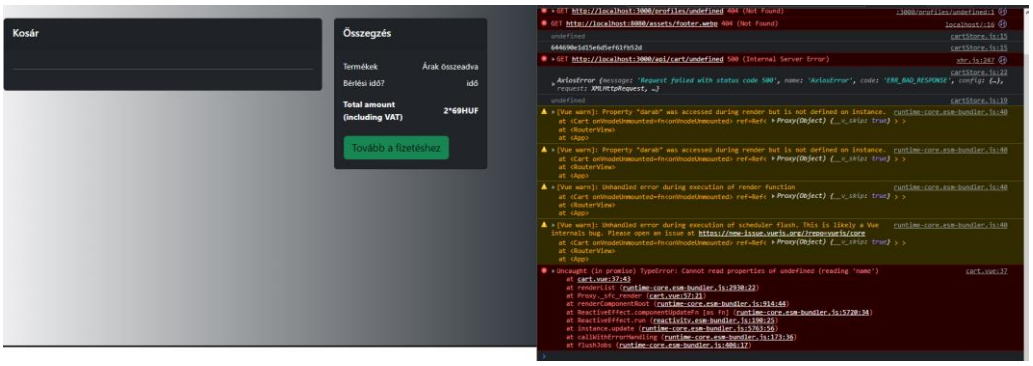
A frontend tervezésekor kitalált kinézetek:



5.3. A fejlesztés fontosabb megoldandó problémái

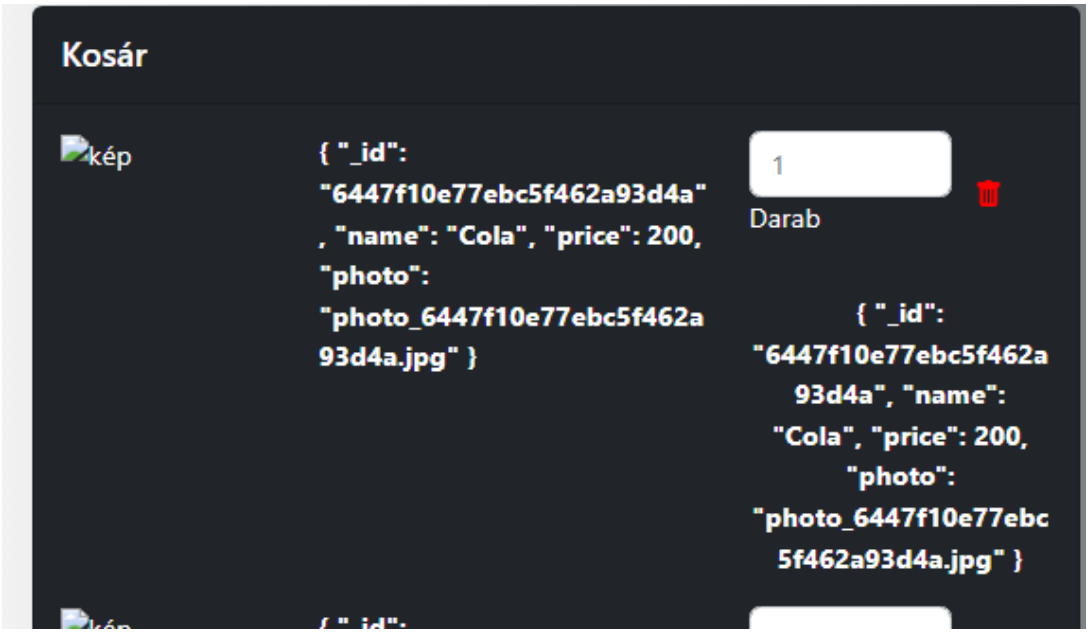
Ami a legnagyobb problémát jelentette az a POST kérésen való átjutás, valamiért az input mezőkbe írt értékek nem akartak tovább küldődni, de végül tanári segítséggel kiderült, hogy

csak rossz féle képpen lettek betöltve a kérésbe. A nem megoldott probléma az még mindig a kosár része a programnak, amely nem rendesen kapja az adatot és maga a tömböt megjeleníti, de amikor egy a tömbbe beírt értéket kérjük, hogy írjon ki valamiért undefinednak veszi a kérés értékét, emiatt nem lett az befejezve, de volt rá terv ami nem is tűnt nehéznek, hiszen v-model segítségével és egy kis számolással meg lehetett volna oldani, de az idő rohamos fogyásával és ennek a hibának nem megoldásával kimaradt a végső részéből a programnak.



10. ábra Kosár hiba

Viszont a tömb megjelenik a kosaron belül



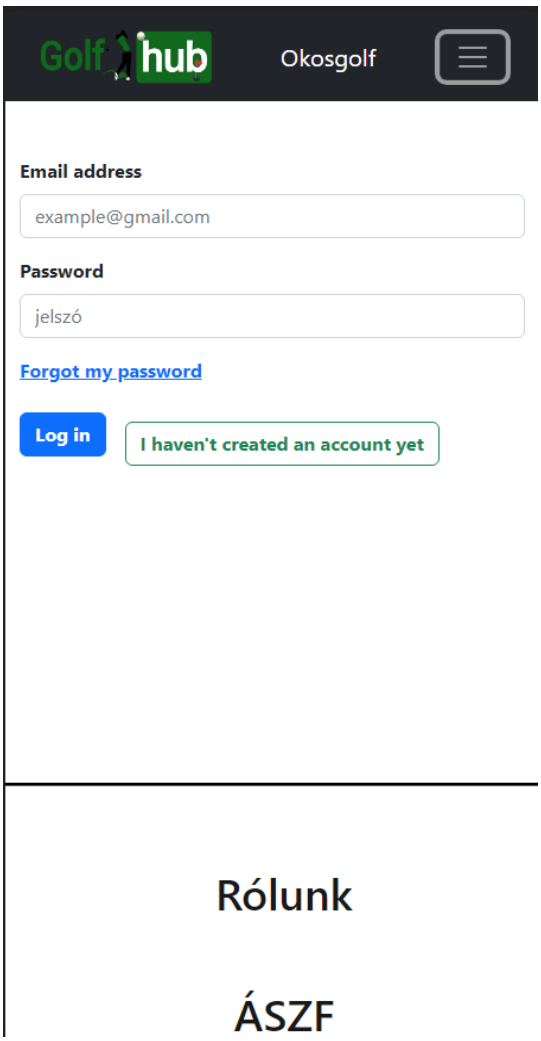
Ezt sajnálatos módon nem sikerült megoldani időben a leadásra.

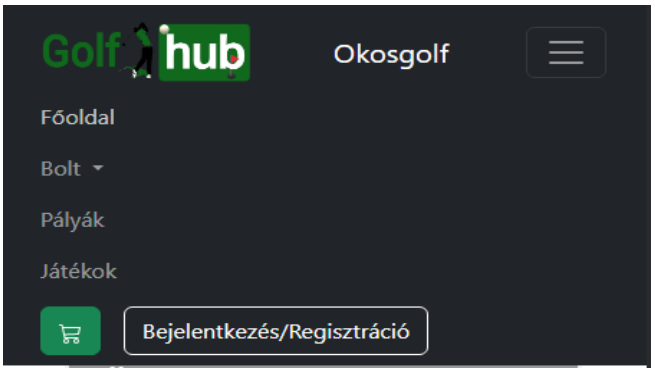
Viszont a program további bővítésére megfelelő ez az oldal.

6. Mobil alkalmazás, mobil nézet

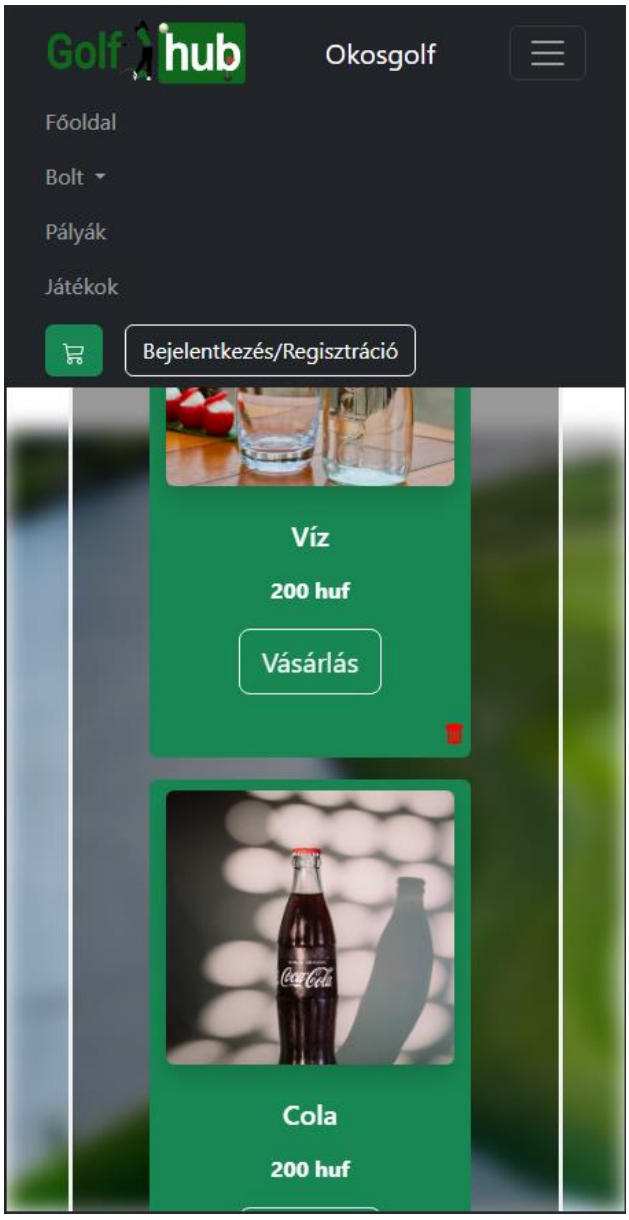
Kimondott mobil alkalmazás nem készült, de viszont a weboldalunk minden oldala reszponzívan működik, úgyhogy bárki, aki golfozás közben szeretné felvinni az éppen elért eredményét, vagy mondjuk szeretne rendelni valamit, amit fel tud venni a fő épületben, azt megteheti az oldalról. Ezzel is kényelmesebbé téve a golfozók kikapcsolódását az elérhető 15 pályáinkon. Ha mást nem nyomon tudja követni golftársai elért eredményeit.

A bootstrap segítségével egyszerű volt oldalainkat és a rajta lévő képeket, szövegeket reszponzívvá tenni.





Bár egy pár dologgal voltak problémák, de semmi olyasmi, amit pár Google kereséssel meg nem lehetett volna oldani. A Navigációs résszel az elején probléma akadt azzal, hogy nem rendesen esik le és az alatta lévő elemek bele lógnak, de ezt azóta ki javítottuk és minden megfelelően működik mobilos használatra. A képen látva, hogy megoldódott a navbar probléma és funkcionális.



7. Tesztelés

A teszteléseket úgy próbálgattuk, ahogy haladtunk. A backend teszteléséhez egy Visual Studio Code kiegészítőt használtunk ThunderClient néven, amivel biztosra tudtunk menni, hogy a backend része precízen és folyamatosan jól működik. A frontend részen a böngészőbe beépített konzol segítségével oldottuk meg a problémákat.

Backendben a tesztelés során nehezen tudtuk megoldani, hogy a játékok controllerbe ha valaki hoz létre egy játékot, akkor annak az embernek a nevét jelenítse meg, aki létrehozta azt. Ezt a lekérési kód más-más kombinációjával való tesztelés során sikerült megoldani.

```
exports.createJatek = async (req, res, next) => {
    try {
        const { szam, playedAt, score } = req.body

        const jatek = await Jatek.create({
            user : await User.findById(req.user.id),
            szam,
            playedAt,
            score
        })
      res.status(201).json({ success: true, data: jatek });
    } catch (error) {
      next(error)
    }
};
```

Így amikor valaki egy játékot hoz létre, akkor a bejelentkezés segítségével lekéri a user ObjectId-ját a MongoDb adatbázisból, és azt menti el a kollekción belül. Majd ahhoz, hogy megkapjuk a hozzá rendelt felhasználó nevét a GET kérésnél populate függvényt használva

```
exports.getJatekok = async (req,res,next)=>{
    try {
        const jatekok = await Jatek.find().populate({
            path: 'user szam',
            select: 'name szam dif',

        })
        res.status(200).json({ success: true , count: jatekok.length, data: jatekok });
      } catch (error) {
        next(error)
      }
}
```

kiírjuk.

A backend és frontendnek való összekötése közben is hibába ütköztünk, mivel a kéréseinket nem akarta a backend felé küldeni a frontenden használt Axios HTTP kliens. Ezt úgymond gyorsan meg tudtuk oldani annak szerencséjére, hogy másoknak is ez a probléma jött elő egy-két nappal a miénk előtt.

```
router.use(function(req, res, next) {
    res.header("Access-Control-Allow-Origin", "*");
    res.header("Access-Control-Allow-Methods", "GET,HEAD,OPTIONS,POST,PUT,DELETE");
    res.header("Access-Control-Allow-Headers", "Origin, X-Requested-With, Content-Type, Accept, Authorization, X-Api-Key");
    next();
})
```

Minden átengedési műveletet megengedtünk a bejövő kéréseknek a Server.Js-ben és az elérhető útvonalakon is.

```
app.use(function(req, res, next) {
    res.header("Access-Control-Allow-Origin", "*");
    res.header("Access-Control-Allow-Methods", "GET,HEAD,OPTIONS,POST,PUT");
    res.header("Access-Control-Allow-Headers", "Origin, X-Requested-With, Content-Type, Accept, Authorization, X-Api-Key");
    next();
  });
```

A frontenden is megjelent tesztelés közben pár hiba. Elsők közt nem sikerült rendesen a GET kérésünkből kiolvasni az adatokat, amire a hiba az volt, hogy a visszaküldött adaton belül van még egy adat és azt kell beolvasnunk a számunkra kívánt tömbben. Ezt egy egyszerű kód hozzácsatolással meg lehetett oldani és mindenhol, ahol rosszul lett írva ki lett javítva.

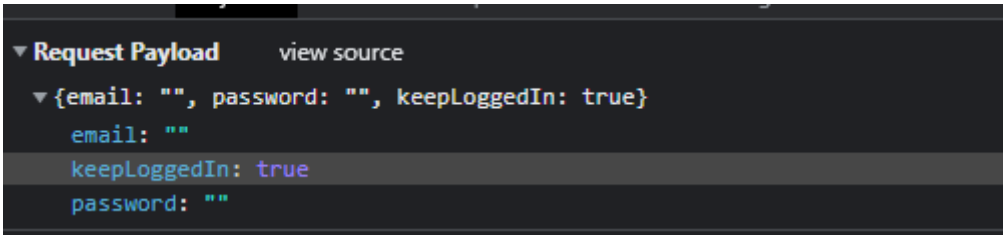
```
state: () => ({
    eszkoz: [],
}),
getters: {},
actions: {
    getEszkozok() {
        Axios.get(`/eszkozok/`)
            .then(resp => {
                this.eszkoz = resp.data.data
            })
    },
```

A másik nagyobb hiba, ami a frontenden került elő, hogy nem jó változókkal küldtük a POST kérésre az adatokat, így több napot is megakadva a projektmunkával, ami közben csak a

kinézetét tudtuk a programnak változtatni, hiszen a tervezett funkciókat a POST nélkül nem tudtuk megoldani.

```
const email = '';
const password = '';
let keepLoggedIn = true;

  postUserLogIn(email, password, keepLoggedIn);
```



Az elején kigondolt kóddal a kérésünk semmit nem küldött tovább, amit az input mezőkbe írtunk. Szerencsére hosszas keresgélés után megtaláltuk mi volt a probléma és onnantól kezdve megoldódott a POST-olási problémánk. A segítségével a másik két főbb kérést is sikerült megoldani, így a PUT és a DELETE is működött.

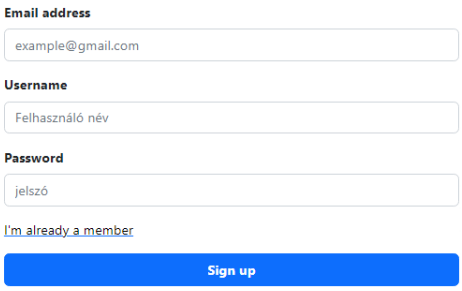
```
const email = ref();
const password = ref();
function userlogin(){
  console.log(email.value, password.value);
  postUserLogIn(email.value, password.value);
  push({path:'/'})
}
```

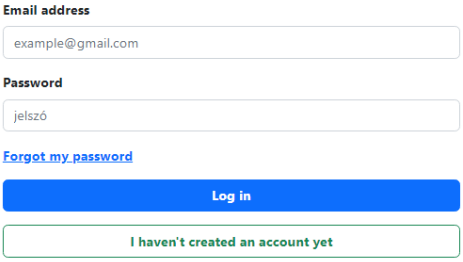
Egy külön functionön belül kellett rakni a kérést, és a Vueba alapból beépített Ref importálása után meg tudtuk kapni az inputmezőbe írt értékeket.

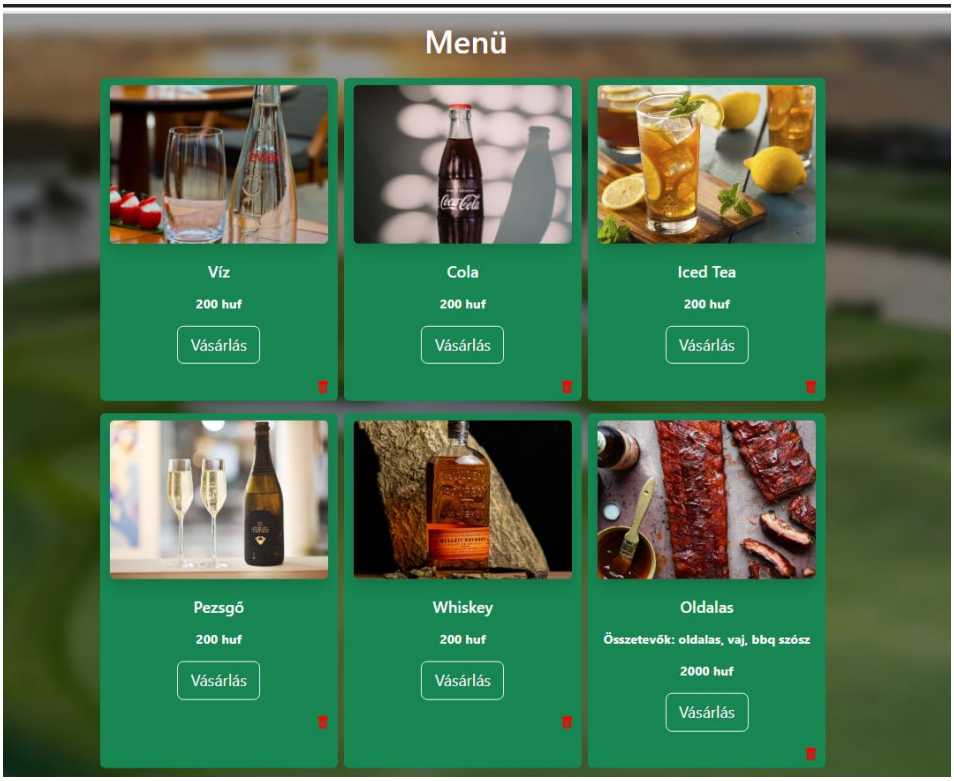
Ami azóta sem oldódott meg az a fent megemlített kosár problémája, aminek többféle megoldás próbálgatása után sem sikerült működésre bírni.

8. Felhasználói útmutató

Regisztráció után a regisztrált Email címmel és jelszóval be kell lépni a Bejelentkezés fülön keresztül, sikeres bejelentkezés után az profilba be is sikerült lépni, onnan kezdve a webshopon belül tetszése szerint vásárolhat a vevő az elérhető termékek között, vagy rendelhet a helyszíni konyháról. Ha a golfpályán elkezd játszani, akkor a játékok menüpontban az azon pálya azonosítóját megadja, ahol éppen játszott, majd beírja ütéseinek számát, amivel a golflabdát a lyukba ütötte azt helyben tudja rögzíteni. A profilon keresztül tudja adatait frissíteni, viszont új profilkép kérése esetén a személyzetnek kell szólnia, hogy csináljanak róla egy képet és változtassák meg az adatbázisban.







+Új játék mentése

Felhasználó	Játszott pálya száma	Ütések száma	Játék dátuma	
poliiiiiiiiii	1	6	2023-04-26T10:50:40.887Z	
poliiiiiiiiii	2	1	2023-04-26T11:47:16.768Z	
poliiiiiiiiii	14	8	2023-04-27T13:58:56.773Z	

A személyzetnek a weboldalon belül elérhető új termékek és menü elemek létrehozása, és azok törlése. Ahhoz, hogy valaki nagyobb staff vagy admin rankot érjen el manuálisan be kell azt állítani az ott dolgozóknak, hogy biztosra menjenek abban, hogy megbízható embereknek adják ki a roleokat.

9. Összegzés

Véleményünk szerint szerint sajnáljuk, hogy az utolsó pillanatra lett hagyva majdnem az egész beadandó. Visszagondolva a szeptemberben kiszabott terveinkre, ha netán elég időt szántunk volna rá nagy részét meg is valósíthattuk volna. De nagy részbe foglalva ebben a programban alapokul benne van az, amit még az iskola kezdeténél kitűztünk magunknak céloknak. A feladatok készítése közben rájöttem, hogy mégis lehet élvezni a programozást, ha nem egy szabott kereten belül kell mozogni, hanem a saját probléma megoldó képességeidre vagy hagyatkozva. A csapat bizonytalansága miatt az elején semmi sem volt biztos a projekttel kapcsolatban, de örülök, hogy a végére mindenki kivette a saját részét annak elkészítésében, és mindenki szorgosan próbálta összerakni a saját feladatát. Ami még mindig zavar, hogy a kosarat nem sikerült megoldani. Egy ideig mindenképp zavarni fog mindig, amikor visszaemlékszem, hogy az az egy dolog, ami nem sikerült, mondhatni az egész weblap legfontosabb része. A jövőre tekintve tanultunk ebből és más hozzáállással állunk neki projekteknek.

Források:

- Kód:
    - https://getbootstrap.com/
    - https://icons.getbootstrap.com/
    - https://stackoverflow.com/
    - https://www.w3schools.com/
    - https://vuejs.org/
    - https://www.youtube.com/
    - https://nodejs.org/en
    - plusz az órai anyag: Soós Gábor Tanár Úr (Nodejs), Bólya Gábor Tanár Úr (Vuejs)

Kép:

    - https://images.google.com/
        - ezen belül minden ami golffal kapcsolatos
    - Paint (pályák + logó összeszerkesztése)
    - https://photoscissors.com/
    - https://www.kennyflowers.com/
    - https://www.pingeurope.com/
    - https://www.eurekagolf.co.uk/

- Dokumentáció:
    - sikeres vizsga dokumentáció példa

10. Tartalomjegyzék