# DOKUMENTÁCIÓ

## Webtechnológiák 2 BSc

2023. tavasz féléves feladat

Készítette: **Tőzsér Zétény**
Neptunkód: **QGNLD2**

## Bevezetés

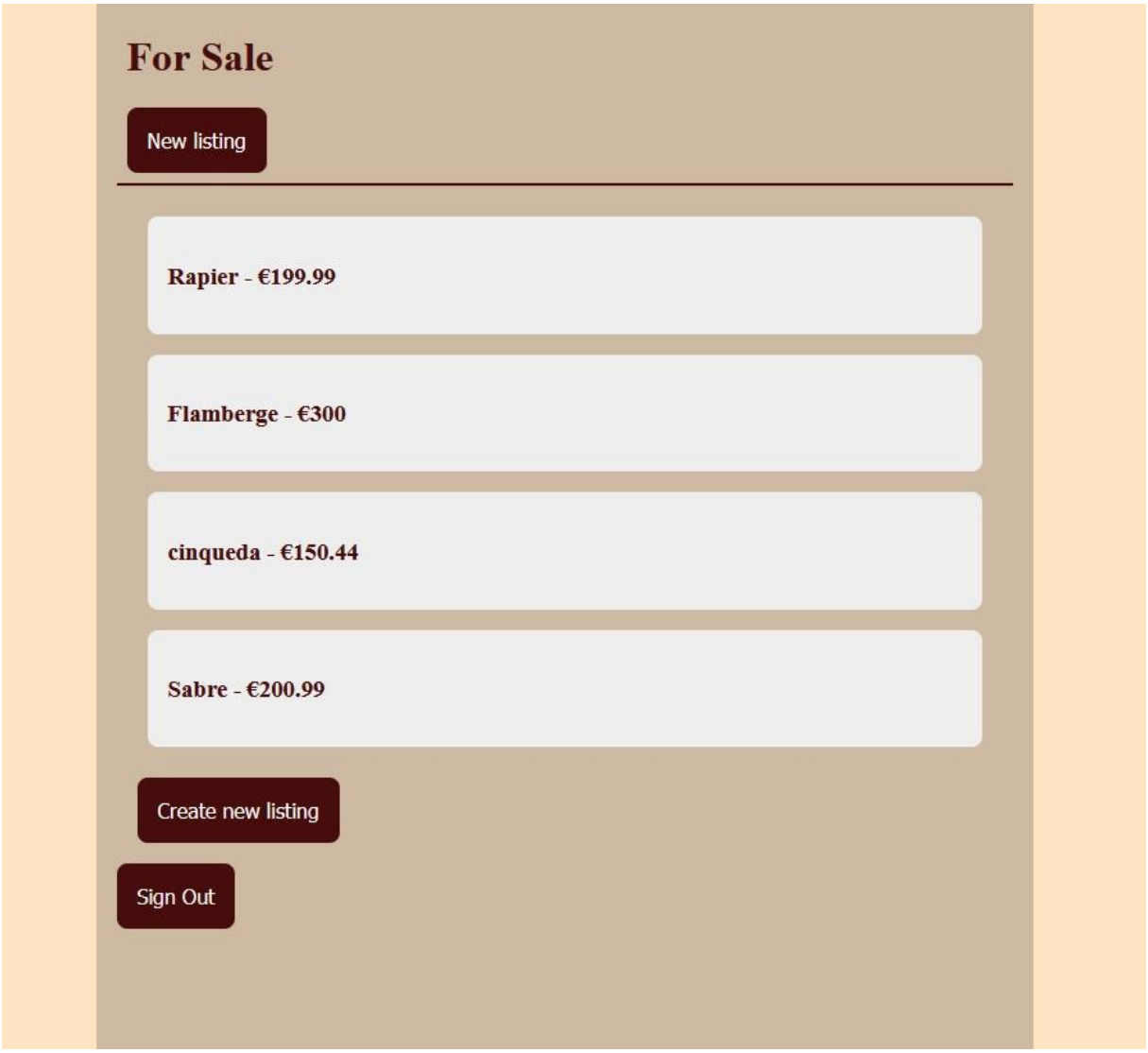
A feladat egy Angular keretrendszerben készült nyilvántartórendszer. Egy kardok adás - vételével foglalkozó weboldal egyszerűsített formája. Google fiókkal való bejelentkezés után lehetőségünk van a listázott tárgyak részletes leírását megtekinteni, újabb tárgyakat létrehozni és törölni a meglévőeket.

Az adatok tárolására az MySQL adatbázis kezelő rendszert használtam. A bejelentkezéshez a Firebase szolgáltatásait vettem igénybe. További felhasznált technológiák: hapi és Node.js.

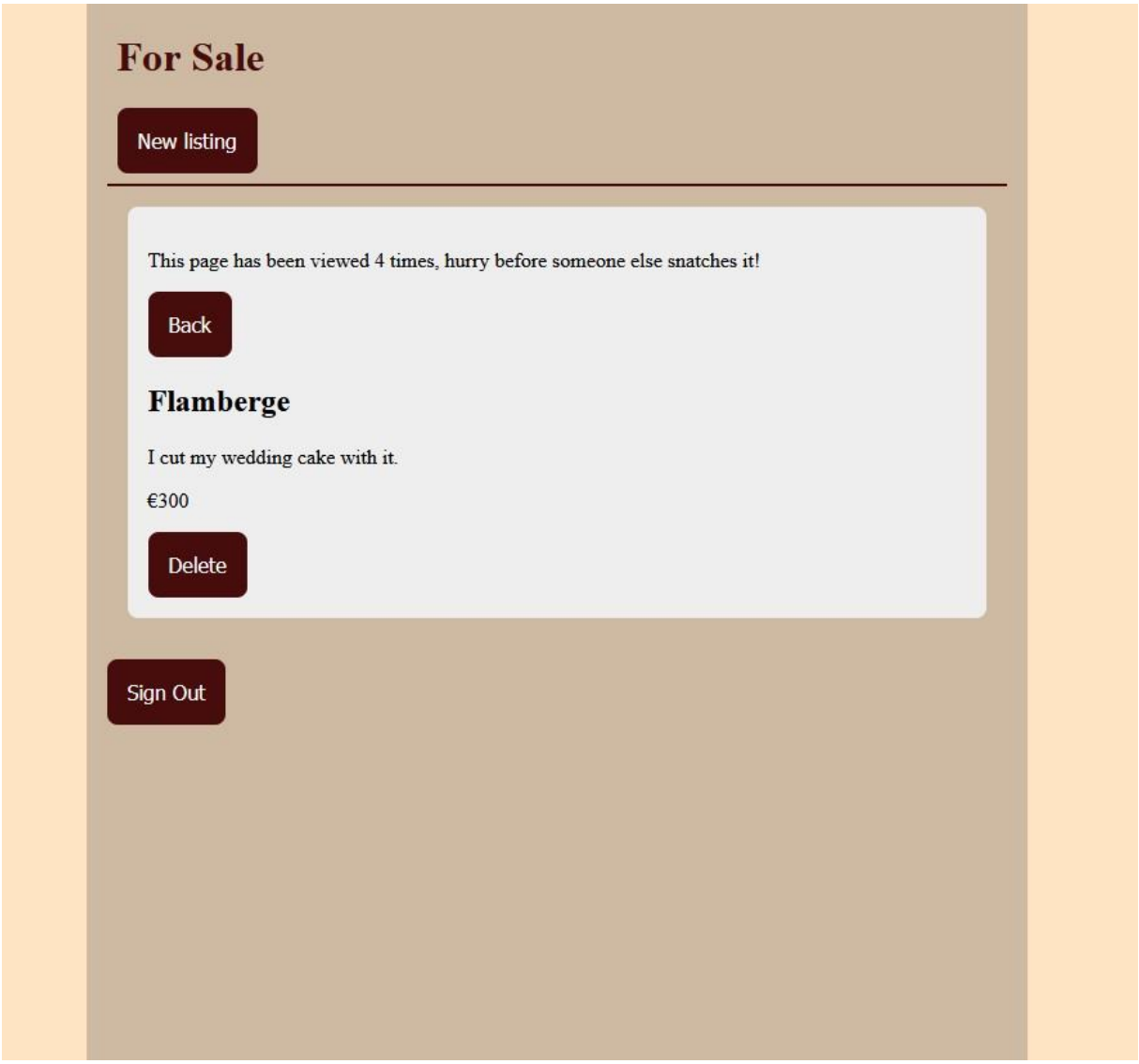
## Az oldal

Amikor először látogatjuk az oldalt, akkor csak a belépési felületet látjuk. Google fiókkal lehet belépni. Ezt a Firebase biztosítja.

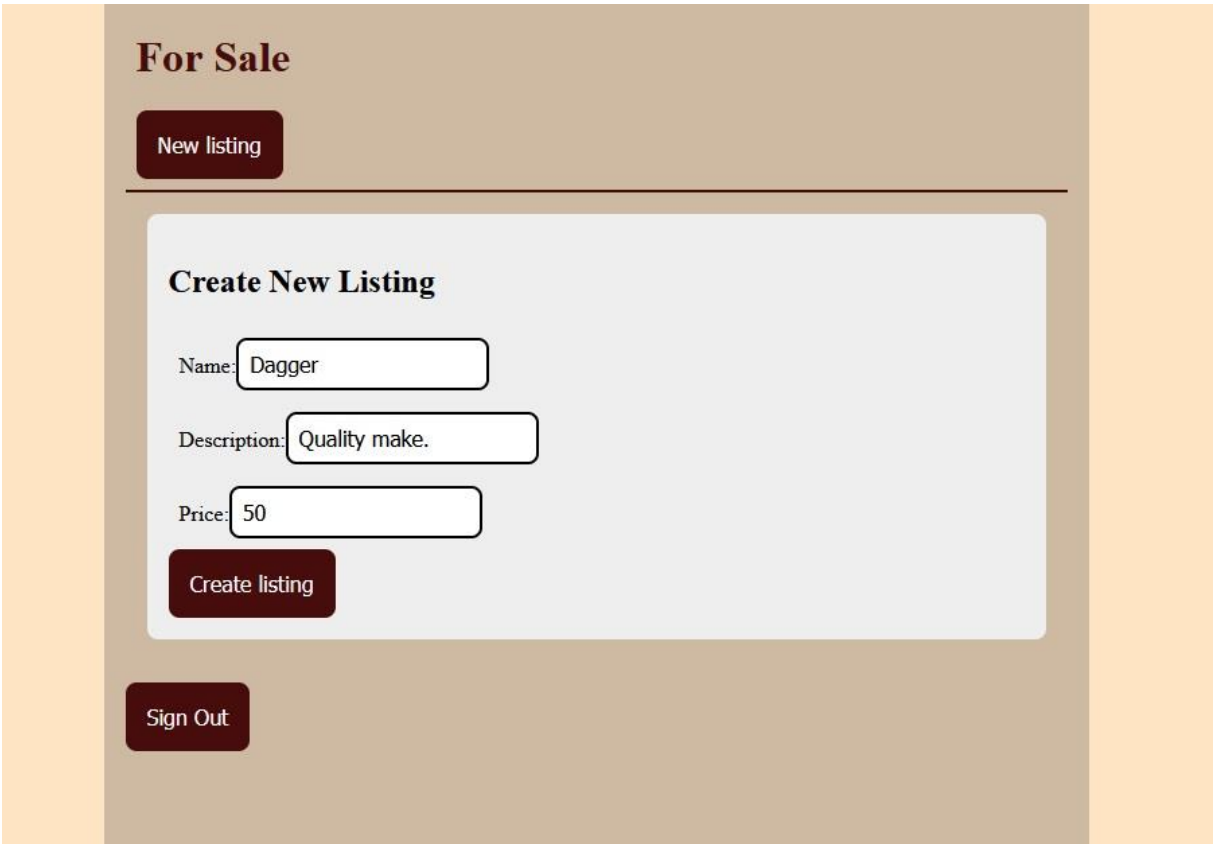
Bejelentkezés után a főoldalra kerülünk. Itt láthatjuk az elérhető tárgyakat. Ezekre kattintva megtekinthetjük a részletes leírásukat. Felül található a navigációs sáv. Nincsen sok haszna, hiszen csak a főoldal és az új tárgy létrehozása érhető el rajta.



A részletek oldalon a tárgy leírása és a megtekintések száma látható. Itt lehet kitörölni a tárgyat.



Új tárgy hozzáadásához a nevet, leírást és árat kell megadni. Az azonosító egy generált uuid. A megtekintések száma 0-ra állítódik be. Az ár lehet egész, vagy tört szám, tizedespontot használva. Mást beütve az ár mezőbe, nem engedi az oldal az új tárgy létrehozását.



Látható, hogy a „Dagger” tárgy hozzá lett adva az oldalhoz.


## A kód

Frontenden mindegyik oldal egy komponens, amihez tartozik egy html, egy css és két typescript fájl. A html és a css az oldal megjelenéséért felelnek. Az alkalmazás kis mérete és html elemek újrahasználása miatt, a styles.css fájlban van összegíűjtve minden stílus elem.

Részlet a styles.css fájlból:

Swords > src > # styles.css > #max-width-column

```css
button {
    background-color: rgb(71, 13, 13);
    border: none;
    border-radius: 8px;
    color: white;
    cursor: pointer;
    outline: none;
    padding: 16px;
    font-size: 16px;
}

input, textarea {
    border: 2px solid black;
    border-radius: 8px;
    font-size: 16px;
    padding: 8px;
}
```

Minden komponenshez tartozik egy specs.ts-re végződő fájl. Ez az egyes komponensek tesztelésére szolgál. A komponens-neve.ts fájl a fontosabbik. Ez működteti a komponenst. Meghatározza, hogy mi történjen gombokra kattintáskor, kommunikál a backenddel.

A részletek oldal html és typescript fájljai:

Swords > src > app > listing-detail-page > listing-detail-page.component.html > ...

Go to component

```html
<div class="content-box" *ngIf="!isLoading">
    <p>This page has been viewed {{listing.views}} times, hurry before someone else
    <a routerLink="/listings">
        <button>Back</button>
    </a>
    <h2>{{ listing.name }}</h2>
    <p>{{ listing.description }}</p>
    <p>€{{ listing.price }}</p>
    <button (click)="onDeleteClicked(listing.id)">Delete</button>
</div>
<div class="content-box" *ngIf="isLoading">
    <h2>Loading...</h2>
</div>
```

Swords > src > app > listing-detail-page > TS listing-detail-page.component.ts > ...

```ts
import { Component, OnInit } from '@angular/core';
import { ActivatedRoute } from '@angular/router';
import { ListingsService } from '../listings.service';
import { Listing } from '../types';

@Component({
  selector: 'app-listing-detail-page',
  templateUrl: './listing-detail-page.component.html',
  styleUrls: ['./listing-detail-page.component.css']
})
export class ListingDetailPageComponent implements OnInit {
  isLoading: boolean = true;
  listing: Listing;
  listings: Listing[] = [];

  constructor(
    private route: ActivatedRoute,
    private listingsService: ListingsService,
  ) { }

  ngOnInit(): void {
    const id = this.route.snapshot.paramMap.get('id');
    this.listingsService.getListingById(id!).subscribe(listing => {
      this.listing = listing;
      this.isLoading = false;
    })!;

    this.listingsService.addViewToListing(id!).subscribe(listing =>
      console.log('Views updated.'));
  }

  onDeleteClicked(listingId: string): void {
    this.listingsService.deleteListing(listingId).subscribe(() => {
      this.listings = this.listings.filter(listing => listing.id !== listingId);
    });
  }
}
```

A proxy.config.json file lokális tesztelésnél használt. A front- és backend nem azonos helyen futnak, így CORS hibát kapnánk nélküle.

Swords > {} proxy.config.json > ...

```json
{
    "/api/*": {
        "target": "http://localhost:8000",
        "secure": false,
        "logLevel": "debug",
        "changeOrigin": true
    }
}
```

Backenden a .babelrc lehetővé teszi a legfrissebb Javascript szabványok használatát, amikor ha egyik eszközünk nem támogatná azokat.

A server.js fájl hozza létre a hapi szervert. Elindítja, hibakezelést határoz meg és csatlkaozik az adatbázisra.

Swords-backend > src > JS server.js > ...

```
1   import Hapi from '@hapi/hapi';
2   import routes from './routes';
3   import { db } from './database';
4   import * as admin from 'firebase-admin';
5   import credentials from '../credentials.json';
6
7   admin.initializeApp({
8       credential: admin.credential.cert(credentials),
9   });
10
11  let server;
12
13  const start = async () => {
14      server = Hapi.server({
15          port: 8000,
16          host: 'localhost'
17      });
18
19      routes.forEach(route => server.route(route));
20
21      db.connect();
22      await server.start();
23      console.log(`Server is listening on ${server.info.uri}`);
24
25  }
26
27  process.on('unhandledRejection', err => {
28      console.log(err);
29      process.exit(1);
30  });
31
32  process.on('SIGINT', async () => {
33      console.log('Stopping server...');
34
35      await server.stop({ timeout: 10000 });
36
37      db.end();
38      console.log('Server stopped');
39      process.exit(0);
40  });
41
42  start();
```

A database.js összeköti az alkalmazást az adatbázissal, továbbítja a lekéredzéseket az adatbázis felé.

Swords-backend > src > JS database.js > ...

```
import mysql, { escape } from 'mysql';

const connection = mysql.createConnection({
    host: 'localhost',
    user: 'hapi-server',
    //password: Asdfg1234
    //Workbench wants one with mixed case letters, numbers and 8 chars min.
    password: 'Asdfg1234',
    database: 'swords'
});

export const db = {
    connect: () => connection.connect(),
    query: (querySrtring, escapedValues) =>
    new Promise((resolve, reject) => {
        connection.query(querySrtring, escapedValues, (error, results, fields) =>{
            if (error) reject(error);
            resolve({ results, fields });
        })
    }),
    end: () => connection.end(),
}
```

A routes mappában találhatók az egyes oldalakhoz tartozó elérési útvonalak és adatbázis lekérdezések. Például a részletek oldal az adozt azonosítójú terméket kérdezi le.

Swords-backend > src > routes > JS getListing.js > ...

```
import Boom from '@hapi/boom';
import { db } from '../database';

export const getListingRoute = {
    method: 'GET',
    path: '/api/listings/{id}',
    handler: async (req, h) => {
        const id = req.params.id;
        const { results } = await db.query(
            'SELECT * FROM listings WHERE id=?',
            [id],
        );
        const listing = results[0];
        if(!listing) throw Boom.notFound(`Listing does not exist with id ${id}`);
        return listing;
    }
}
```