

## Eötvös Műhely @ Szegedi Tudományegyetem

### Modern webtechnológiák a kutatás szolgálatában

Az oktatás témája a node.js<sup>1</sup> keretrendszer volt, megismerkedtünk azzal, hogy hogyan lehet felépíteni egyszerű node.js szerver oldali alkalmazást, hogy tudunk vele kommunikálni egy frontenddel és egy mongoDb<sup>2</sup> adatbázisban tárolni az adatokat. A nap folyamán létrehoztunk egy amőba játékot frontenden, amivel hogyha klikkeltünk egy mezőre akkor a szerver oldalon kiírtuk a koordinátákat (tömb indexek). Ahhoz, hogy normális jegyzetet tudjunk írni, eldöntöttük az előadóval, hogy egy kis példa projektet<sup>3</sup> csinál mindenki magának, amiről röviden beszámol jelen dokumentációban.

Én egy nagyon egyszerű webshop alkalmazást találtam ki, amiben van pár termék alaphól, az adatbázisban ami egy mongoDb dockerizált környezetben futtattam (korábbi szoftverfejlesztési munkából adódóan ez volt a legegyszerűbb és leggyorsabb módja) amit betudtunk rakni egy kosárba, egy másik page-n pedig megtekinteni és kitörölni a kosárból.

Tehát két collection van = [products, shopping\_cart] minden egyes objektum három adattal rendelkezik = {

\_id,

name,

price

}, ahol az:

- \_id egy 12 byte -os hexadecimális értékekből álló string, ahol:
  - az első 4 byte reprezentálja a másodperceket a Unix időszámítás óta (1970. január 1. éjféli)
  - a következő 5 byte véletlenszerű érték
  - az utolsó 3 byte pedig egy számláló véletlenszerű értéktől indulva
- name: egy string ami a nevet reprezentálja
- price: egy szám ami az árat reprezentálja

Bár célszerű lett volna az adatbázist jobban megtervezni, kulcsokat alkalmazni (\_id), egy közös collection-re hivatkozni és csak a kulcsokat tárolni máshol, de a projekt lényege és a feladat szintje nem ezt kívánta meg.

A frontend amit ezen a napon ismertem meg a Vue.js<sup>4</sup> (Angular frontend fejlesztéssel foglalkozok a Szegedi Tudományegyetem Szoftverfejlesztési Tanszékén) egyetemi képzésekhez képest ez is más volt, de az Angular tapasztalatom miatt volt némi rálátásom, ami miatt gyorsan belerázódtam.

Ubuntu 18.04-en fejlesztettem, Visual Studio Code IDE-vel és mint említettem Docker-Compose-t használtam az adatbázishoz fejlesztés közben.

---

<sup>1</sup> Node.js: <https://nodejs.org/en/docs/>

<sup>2</sup> MongoDB: <https://docs.mongodb.com/>

<sup>3</sup> A projekt elérési útvonala: [https://github.com/eszesnorbi/eotvos\\_beadando](https://github.com/eszesnorbi/eotvos_beadando)

<sup>4</sup> Vue.js: <https://vuejs.org/v2/guide/>

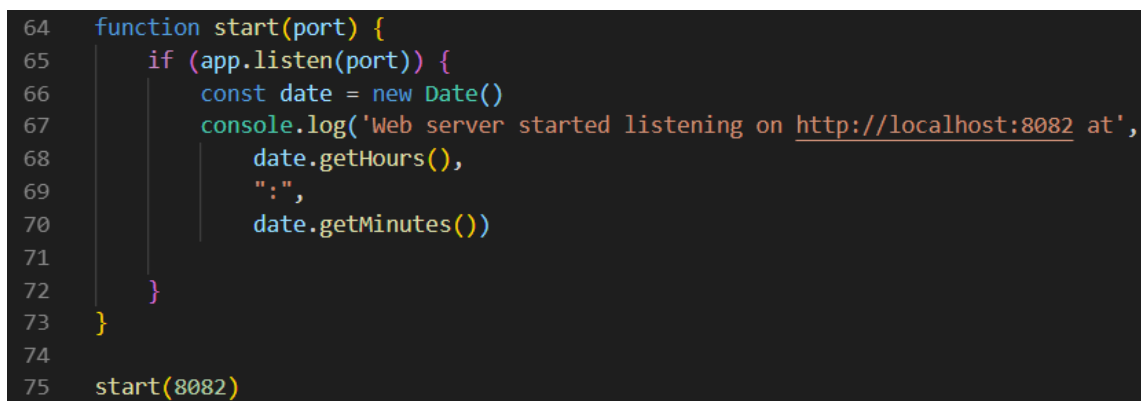
Kezdjük a node.js applikációval. npm init-tel létrehoztuk, index.js fájlt is és be importáltam a szükséges csomagokat, a kommunikációhoz, a http body parsereléshez, a cors-hoz a mongo-db-hez.



```
JS index.js x
1  const express = require('express')
2  const cors = require('cors')
3  const bodyParser = require('body-parser')
4  const app = express()
5  const MongoClient = require('mongodb').MongoClient
6  var db
7
8  app.use(cors())
9  app.use(bodyParser.json())
10
```

1. ábra Node.js applikáció: importok (1. kép)

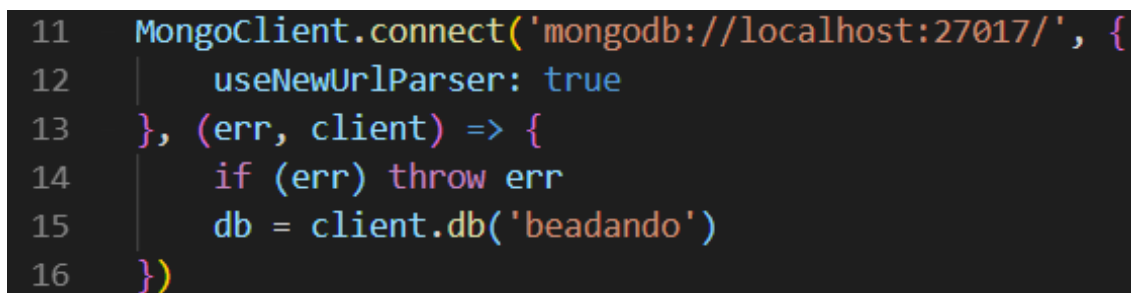
Írtam egy metódust a szerver indításra, hogy stdoutra, a consolera írjon ki egy üzenetet a sikeres elindulás után.



```
64 function start(port) {
65   if (app.listen(port)) {
66     const date = new Date()
67     console.log('Web server started listening on http://localhost:8082 at',
68               date.getHours(),
69               ':',
70               date.getMinutes())
71   }
72 }
73
74
75 start(8082)
```

2. ábra Node.js applikáció: szerver indítás (2. kép)

Adatbázis kapcsolódás a beadandó adatbázishoz.



```
11 MongoClient.connect('mongodb://localhost:27017/', {
12   useNewUrlParser: true
13 }, (err, client) => {
14   if (err) throw err
15   db = client.db('beadando')
16 })
```

3. ábra Node.js applikáció: adatbázis kapcsolódás (3. kép)

Megírtam a post metódust, ami query paraméterben várja a MongoDB collection nevét, hogyha különböző collection-be szeretnénk beszúrni akkor is használható legyen. Body paraméterként kapja az előző oldalon leírt objektumot. A consolera kiírtam, hogy sikeres lett-e a művelet és annak pontos idejét.

```
18 app.post('/:id',
19   (req, res) => {
20     db.collection(req.params.id).insertOne(req.body.item, data => {
21       res.send(data)
22       const date = new Date()
23       console.log(
24         "succesfully post operation at",
25         date.getHours(),
26         ": ",
27         date.getMinutes()
28       )
29     })
30   }
31 )
```

4. ábra Node.js applikáció: post metódus (4. kép)

Teljesen hasonló alapon megírtam a get és delete metódust is.

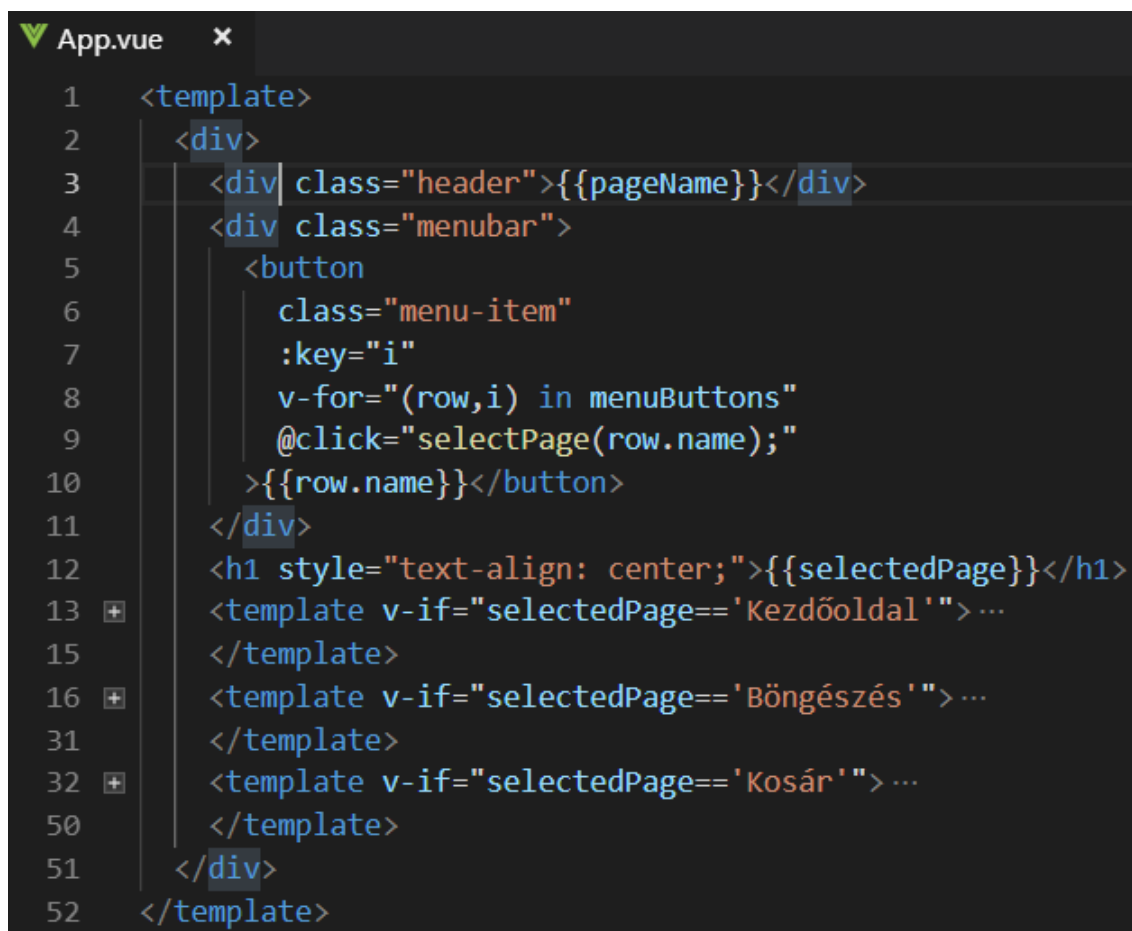
```
33 app.get('/:id',
34   (req, res) => {
35     db.collection(req.params.id).find().toArray((err, data) => {
36       res.send(data)
37       const date = new Date()
38       console.log(
39         "succesfully get operation at",
40         date.getHours(),
41         ": ",
42         date.getMinutes()
43       )
44     })
45   }
46 )
47
48 app.delete('/:id',
49   (req, res) => {
50     db.collection("shopping_cart").deleteOne({_id: req.params.id}), ( () => {
51       const date = new Date()
52       console.log(
53         "succesfully delete operation at",
54         date.getHours(),
55         ": ",
56         date.getMinutes()
57       )
58       db.close()
59     })
60   }
61 )
```

## 5. ábra Node.js applikáció: get és delete metódus (5. kép)

Ezzel ki is merült a Node.js applikáció, a következőkben a Vue.js applikációt fogom bemutatni.

Axios-t használtunk az API-hoz, ami egy promise alapú http kliens, ezt hozzá kellett adni a projekthez. A Vue applikáció elég érdekes épül fel, egy fájlban aminek kiterjesztése a .vue. 3 tag van alából, a <template>, <script>, <style> rendre a html, js és css kódok helyének. Lehet komponenseket létrehozni, importálni és megjeleníteni, ezáltal komplex alkalmazások létrehozására is alkalmas, azonban a projekt szintje miatt, itt az egész alkalmazást egy komponensben, a fő App komponensben írtam meg. Természetesen itt is lehetett volna még jobban szét bontani komponensekre, de ezt nem tettem meg.

Elsősorban a html részt szeretném kifejtetni, a <script> részben láthatóan lesz egy adattag ami a kiválasztott page-t határozza meg és a menubar-ban lévő 3 gomb amire, ha kattintunk az aktuális oldal nevét állítja be az előbb említett adattagnak. A gombok egy tömbből vannak for ciklussal bejárva megjelenítve és css: flex megjelenítés miatt reszponzív.



```
1 <template>
2   <div>
3     <div class="header">{{pageName}}</div>
4     <div class="menubar">
5       <button
6         class="menu-item"
7         :key="i"
8         v-for="(row,i) in menuButtons"
9         @click="selectPage(row.name);"
10      >{{row.name}}</button>
11     </div>
12     <h1 style="text-align: center;">{{selectedPage}}</h1>
13     <template v-if="selectedPage=='Kezdőoldal'">...
14   </template>
15   <template v-if="selectedPage=='Böngészés'">...
16   </template>
17   <template v-if="selectedPage=='Kosár'">...
18   </template>
19 </div>
20 </template>
```

6. ábra Vue.js applikáció: html szerkezet (1. kép)

A kezdőoldalon csak a „Üdvözöllek” üzenet fogad minket, ennek a kreatív elképzelésével most nem foglalkoztam, nem ezen volt a hangsúly. De viszont a böngészés oldalon szerves részének a fele. Ott jelenítjük meg a get metódus által kapott termékeket (products) egy táblázatban, és tudjuk hozzá adni a kosarunkhoz az éppen kilistázott rekordot.

A menüsoron megjelenített gombokhoz hasonlóan itt is for ciklussal van bejárva egy tömb, ahol kiíratjuk a tábla fejlécének tartalmait, majd a termékeket (products).

```

pageName: "Web$$$hop Jah",
menuButtons: [
  {
    name: "Kezdőoldal"
  },
  {
    name: "Böngészés",
    tableRowNames: ["Termék megnevezése", "Ára", "Kosárba rakás"]
  },
  {
    name: "Kosár",
    tableRowNames: ["Termék megnevezése", "Ára", "Törlés"]
  }
],
currency: "Ft",
selectedPage: null,
products: [],
shoppingCart: []

```

7. ábra Vue.js applikáció: adattagok (1. kép)

Az addToShoppingCart() metódus a post hívással a bevásárló kosár collection-höz adja, illetve a gomb csak akkor jelenik meg, ha még nincs a kosárban az adott termék, ez egy if-ben van és egy metódus fut le ami bejárja a tömböt és a paraméterben lévő (aktuális sora a táblázatnak amit a for ciklus bejárása ad) ellenőrzi az id-t, hogy ugyanaz-e. Tehát nem tudunk újra kosárba rakni a terméket.

```

13 <template v-if="selectedPage=='Kezdőoldal'">
14   <div class="centralizedMessage">Üdvözöllek!</div>
15 </template>
16 <template v-if="selectedPage=='Böngészés'">
17   <table>
18     <tr>
19       <td :key="i" v-for="(row,i) in menuButtons[1].tableRowNames">{{row}}</td>
20     </tr>
21     <tr :key="i" v-for="(row,i) in products">
22       <td>{{row.name}}</td>
23       <td>{{row.price}} {{currency}}</td>
24       <td>
25         <button @click="addToShoopingCart(row)" v-if="shoppingCartChecker(row)">
26           
27         </button>
28       </td>
29     </tr>
30   </table>
31 </template>
32 <template v-if="selectedPage=='Kosár'">

```

8. ábra Vue.js applikáció: Böngészés page (2. kép)

A kosár page-n le van kezelve, ha abszolút nincs semmi a kosárban akkor írja ki, hogy „Nincs adat!” ezen kívül előző page-el azon tudása van, csak törlés metódus hívódik meg és más a gomb-ban lévő kép.

```

32     <template v-if="selectedPage=='Kosár'">
33         <div v-if="shoppingCart[0]">
34             <table>
35                 <tr>
36                     <td :key="i" v-for="(row,i) in menuButtons[2].tableRowNames">{{row}}</td>
37                 </tr>
38                 <tr :key="i" v-for="(row,i) in shoppingCart">
39                     <td>{{row.name}}</td>
40                     <td>{{row.price}} {{currency}}</td>
41                     <td>
42                         <button @click="deleteFromShoopingCart(row)">
43                             
44                         </button>
45                     </td>
46                 </tr>
47             </table>
48         </div>
49         <div v-else class="centralizedMessage">Nincs adat!</div>
50     </template>

```

9. ábra Vue.js applikáció: Kosár page (3. kép)

```

methods: {
  selectPage(pageName) {
    this.selectedPage = pageName;
    if (this.selectedPage == "Böngészés") {
      this.getProducts();
      this.getShoppingCart();
    } else if (this.selectedPage == "Kosár") {
      this.getShoppingCart();
    }
  },
  getProducts() { ... },
  getShoppingCart() { ... },
  addToShoopingCart(item) { ... },
  deleteFromShoopingCart(item) { ... },
  shoppingCartChecker(item) {
    for (let i = 0; i < this.shoppingCart.length; i++) {
      if (this.shoppingCart[i]._id == item._id) {
        return false;
      }
    }
    return true;
  }
}

```

10. ábra Vue.js applikáció: Oldalt működtető metódusok (4. kép)

A html rész körülbelül ennyi volt, még hátra vannak a metódusok, amiket két képben jeleníték, először az oldal működéséhez használatosokat másodjára pedig az server hívásokat. Ahhoz, hogy a Böngészés oldalon tudjunk ellenőrizni arra, hogy mi van a kosarunkban, ahhoz ott is le kell kérdezni, ez bele van építve a selectPage metódusba, így mindig, ha az adott page-re navigálunk up-to-date-k vagyunk az adatbázisban lévő adatokkal. A shoppingCartChecker() metódust az előzőekben kifejtettem.

Eszes Norbert a Szegedi Tudományegyetem végzős hallgatója | Neptun kód: DOE61D | email: h652018@stud.u-szeged.hu

Itt vannak a hívások metódusai, paraméter adások amiket a másik oldalon veszünk ki a query paraméterek ( .../:id) és minden body paraméter egyértelműen látszik.

```
90  getProducts() {
91    this.axios.get("http://localhost:8082/products").then(resp => {
92      this.products = resp.data;
93    });
94  },
95  getShoppingCart() {
96    this.axios.get("http://localhost:8082/shopping_cart").then(resp => {
97      this.shoppingCart = resp.data;
98    });
99  },
100 addToShoopingCart(item) {
101   this.axios
102     .post("http://localhost:8082/shopping_cart", { item: item })
103     .then(() => {
104       this.selectPage("Böngészés");
105     });
106  },
107 deleteFromShoopingCart(item) {
108   this.axios.delete("http://localhost:8082/" + item._id).then(() => {});
109   setTimeout(() => {
110     this.getShoppingCart();
111   }, 500);
112 }
```

11. ábra Vue.js applikáció: Szervert hívó metódusok (5. kép)

Kód review végén pedig a beadandó által abszolút nem releváns css blokk.

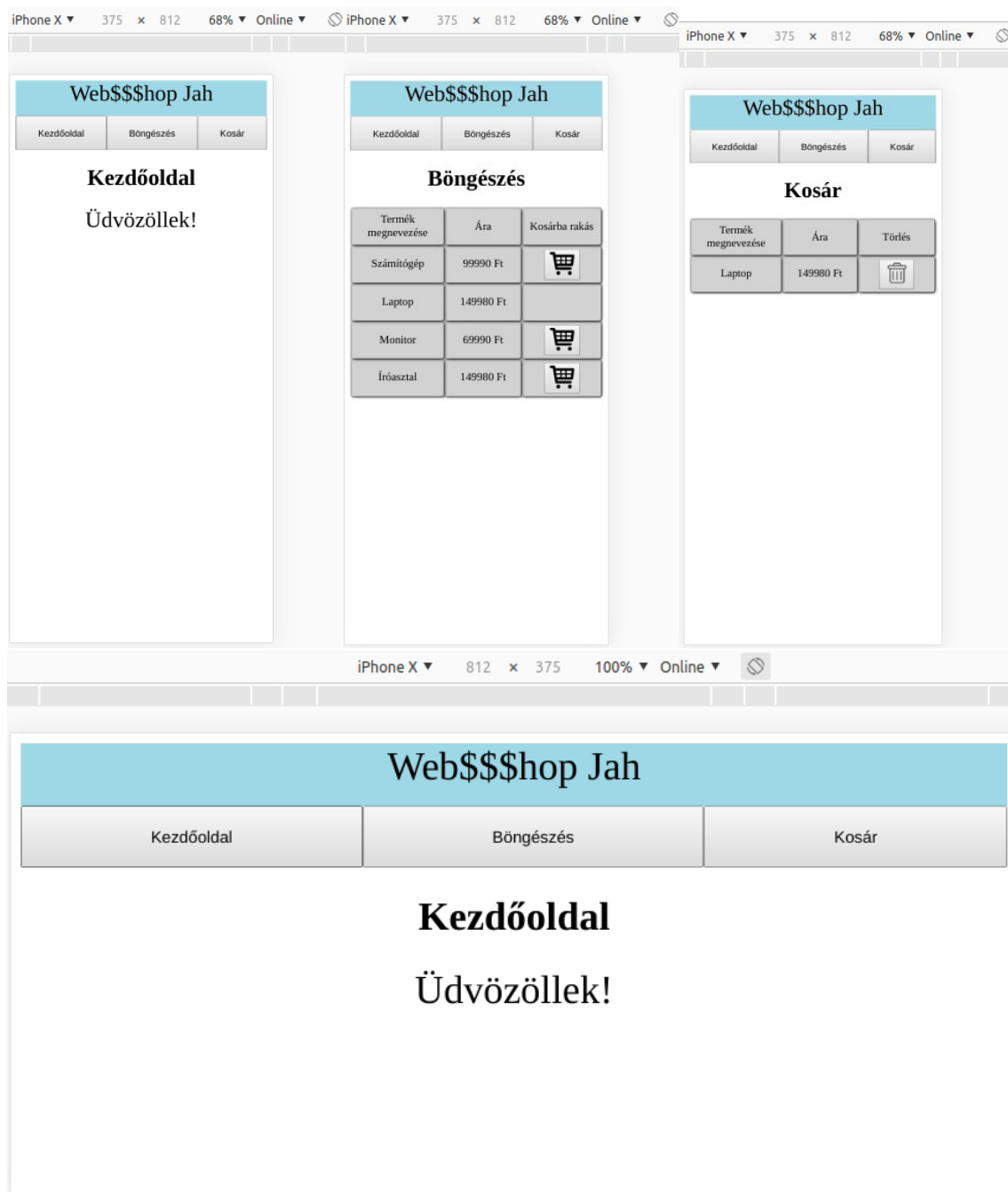
```
125 <style>
126   .header {
127     width: 100%;
128     height: 50px;
129     text-align: center;
130     background-color: lightblue;
131     font-size: 200%;
132   }
133
134   .menubar {
135     width: 100%;
136     height: 50px;
137     display: flex;
138     flex-direction: row;
139     text-align: center;
140     font-size: 150%;
141   }
142
143   .menu-item {
144     flex-grow: 1;
145   }
146
147   table {
148     margin: auto;
149     border: 2px;
150     border-radius: 5px;
151     background-color: lightgray;
152   }
153   table td {
154     text-align: center;
155     width: 200px;
156     height: 50px;
157     border-radius: 3px;
158     box-shadow: 1px 1px 4px black;
159     color: black;
160   }
161
162   .centralizedMessage {
163     text-align: center;
164     font-size: 200%;
165   }
166
167   img {
168     width: 35px;
169     height: 35px;
170   }
171 </style>
```

12. ábra Vue.js applikáció: CSS rész (6. kép)

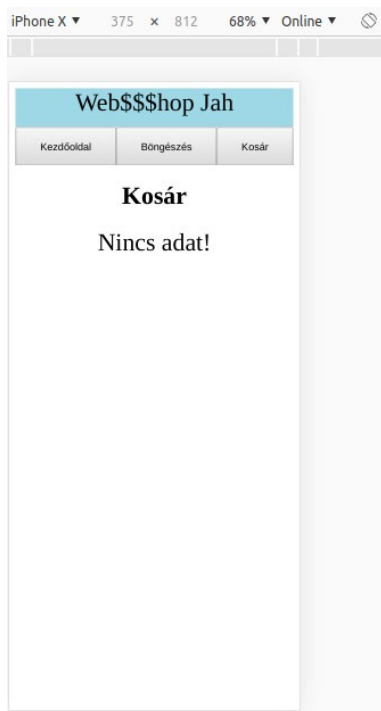
Eszes Norbert a Szegedi Tudományegyetem végzős hallgatója | Neptun kód: DOE61D | email: h652018@stud.u-szeged.hu

Végezetül néhány screenshot a működő projektről. A frontendet mobil nézetben mutatom, mert így a leg hely takarékosabb illetve legalább a reszponzivitás is látszik.





Eszes Norbert a Szegedi Tudományegyetem végzős hallgatója | Neptun kód: DOE61D | email: h652018@stud.u-szeged.hu



```
eszesnorbi@eszesnorbi-TUF-GAMING-FX504GD-FX80GD:~/eotvos_muhely/eotvos_beadand
p/backend$ node .
Web server started listening on http://localhost:8082 at 19 : 15
succesfully get operation at 19 : 15
succesfully get operation at 19 : 15
succesfully post operation at 19 : 15
succesfully get operation at 19 : 15
succesfully get operation at 19 : 15
succesfully get operation at 19 : 15
succesfully get operation at 19 : 15
succesfully get operation at 19 : 15
```

