**Debreceni Szakképzési Centrum**

**Baross Gábor Középiskolája és Kollégiuma**

**Debrecen, 4030**

**Budai Ézsaiás u. 8/A.**

**A szakképesítés azonosító Száma:54 213 05**

**Szakképesítés megnevezése: Szoftverfejlesztő**

**Záródolgozat címe:**

**…. honlap fejlesztése**

**Záródolgozat készítője:**

**…**

**Konzulens: Dr. Várbíróné Nahaji Anikó**

**Konzultációs időpontok:**

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Debrecen, 2021.**

Tartalomjegyzék

[Bevezetés 3](#_Toc98417475)

[Felhasználói dokumentáció 4](#_Toc98417476)

[Feladatspecifikáció 4](#_Toc98417477)

[Rendszerkövetelmények 4](#_Toc98417478)

[Program telepítése, elérése 4](#_Toc98417479)

[Program használatának leírása 4](#_Toc98417480)

[Fejlesztői dokumentáció 9](#_Toc98417481)

[Témaválasztás indoklása 9](#_Toc98417482)

[Alkalmazott fejlesztői eszközök 9](#_Toc98417483)

[Rendszerterv 9](#_Toc98417484)

[Adatmodell leírása 9](#_Toc98417485)

[Program leírása 9](#_Toc98417486)

[Részletes feladatspecifikáció, algoritmusok 9](#_Toc98417487)

[Tesztelés 24](#_Toc98417488)

[Továbbfejlesztési lehetőségek 24](#_Toc98417489)

[Összegzés 25](#_Toc98417490)

[Irodalomjegyzék 25](#_Toc98417491)

[Eredetiségnyilatkozat 26](#_Toc98417492)

# Bevezetés

A záró dolgozatom témáját testvérem tanulmányai alapozták meg. Testvérem jártas a kőműves és a faipari munkákban, anyagokban és eszközökben. Ekkor jött az ötletem, hogy egyszerűbb lenne az ehhez szükséges anyagokat egy helyre összeszednem és elérhetővé tenni a témában kevésbé járatos embereknek.

A fejlesztés során folyamatosan gondolkoztam az új funkciókon mivel bővíthetném a projektem. Készítettem egy mobilaplikációt, mely az adatbázisban tárolt adatokat jeleníti meg, ezen kívül készültek különféle kalkulátorok a számítások megkönnyítéséhez (Pl: beton és parketta kalkulátor). Új funkcióként készítettem egy rendelési felületet is, mellyel a felhasználó kiválaszthatja a megfelelő terméket, fajtáját és a kellő mennyiséget adhatja meg. Az alkalmazás után egy weblapot is szerettem volna készíteni, hogy azok a személyek, akik igazgatják a termékeket és rendeléseket egyszerűen tudjanak dolgozni. Az oldal rendelkezik egy bejelentkezési résszel, ahol a regisztrált felhasználó a számára megfelelő hozzáféréssel érheti el a tartalmakat. Egy átlag felhasználó a web oldalon ugyanazokat láthatja, mint amit mobilon, viszont az irányító személyek hozzáférnek az adatok törléséhez, esetleg új felviteléhez és a rendeléseket tudják elbírálni, hogy hogyan áll a felhasználó által kért összeállítás.

Az oldalam inkább funkcionális, mint látványos, de véleményem szerint ebből a munkából a későbbiekben többet is ki lehetne hozni Pl: a kinézetén, a kezelhetőségén módosítva vagy esetleg újabb funkciókkal hasznosabbá válhat.

# Felhasználói dokumentáció

## Feladatspecifikáció

Mobil applikáció funkciói:

* Anyagok kilistázása
* Rendelés felvitele
* Laminált parketta kalkulátor
* Mixer beton kalkulátor
* Kézi keveréses beton kalkulátor

Weblap funkciói:

* Anyagok kilistázása
* Rendelés felvitele
* Rendelések elbírálása
* Anyagok törlése
* Új anyag felvitele

## Rendszerkövetelmények

milyen hardver igénye van, net elérés, böngésző…reszponzivitás, mobilon, tableten is, mi az ajánlott felbontás

szoftver igények, op. rendszer

## Program telepítése, elérése

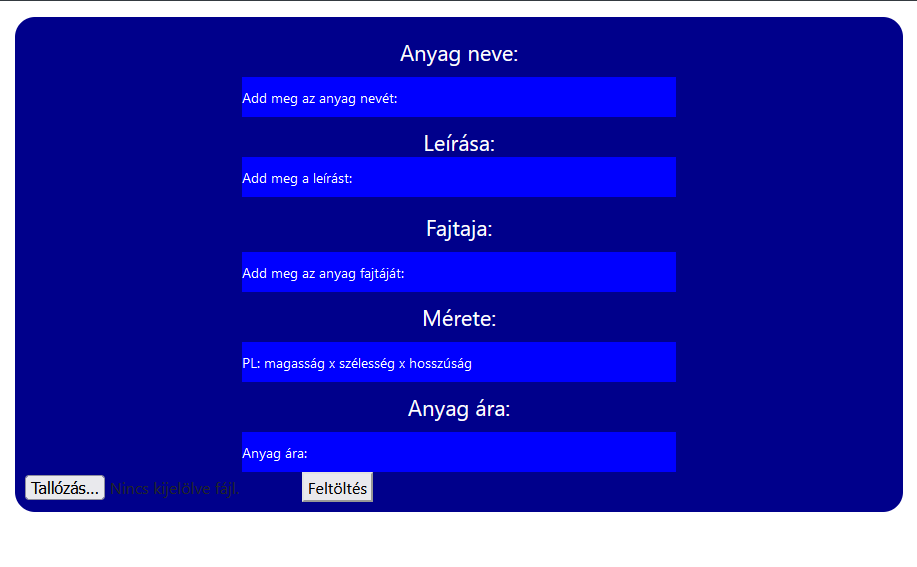
telepítési tudnivalók, ha van

netes elérés, ahova feltetted a lapod, a lap címe, meg kell adni a bejelentkezési adatokat, felhasználónév, jelszó megadása a belépéshez, user-re és admin-ra

## Program használatának leírása

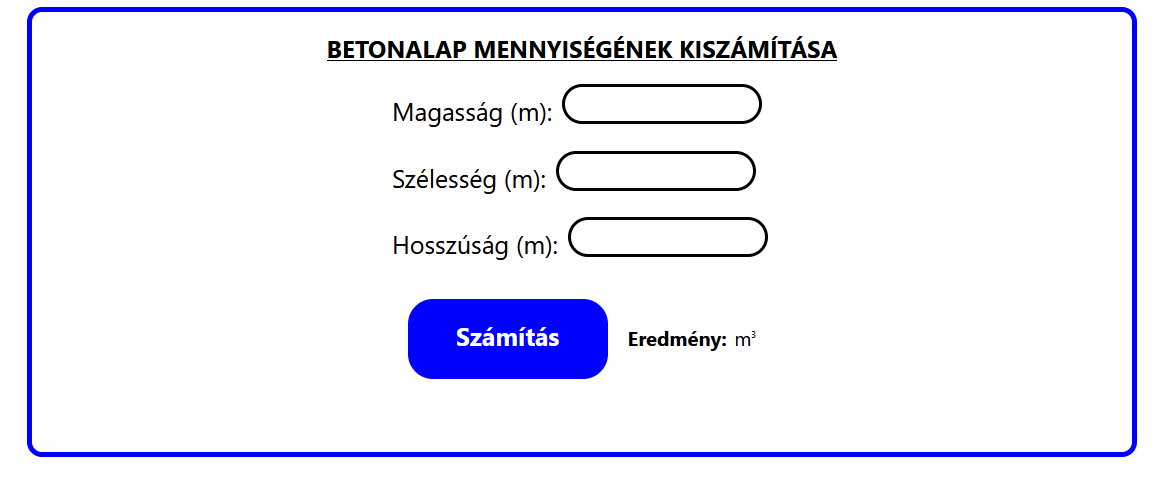
**Anyagok felvitele:**

A mezőkbe megadjuk a kívánt terméket, melyet fel szeretnénk vinni és a hozzátartozó egyéb információkat, a tallózás gombra nyomva elérhetjük a fájlkezelőt a kép feltöltéséhez, majd a feltöltés gombra kattintva az adatokat felviszi az adatbázisba.

****

**Kalkulátor:**

A mezők kitöltése után a számítás gombra kattintva megkapjuk a szükséges anyag mennyiséget.

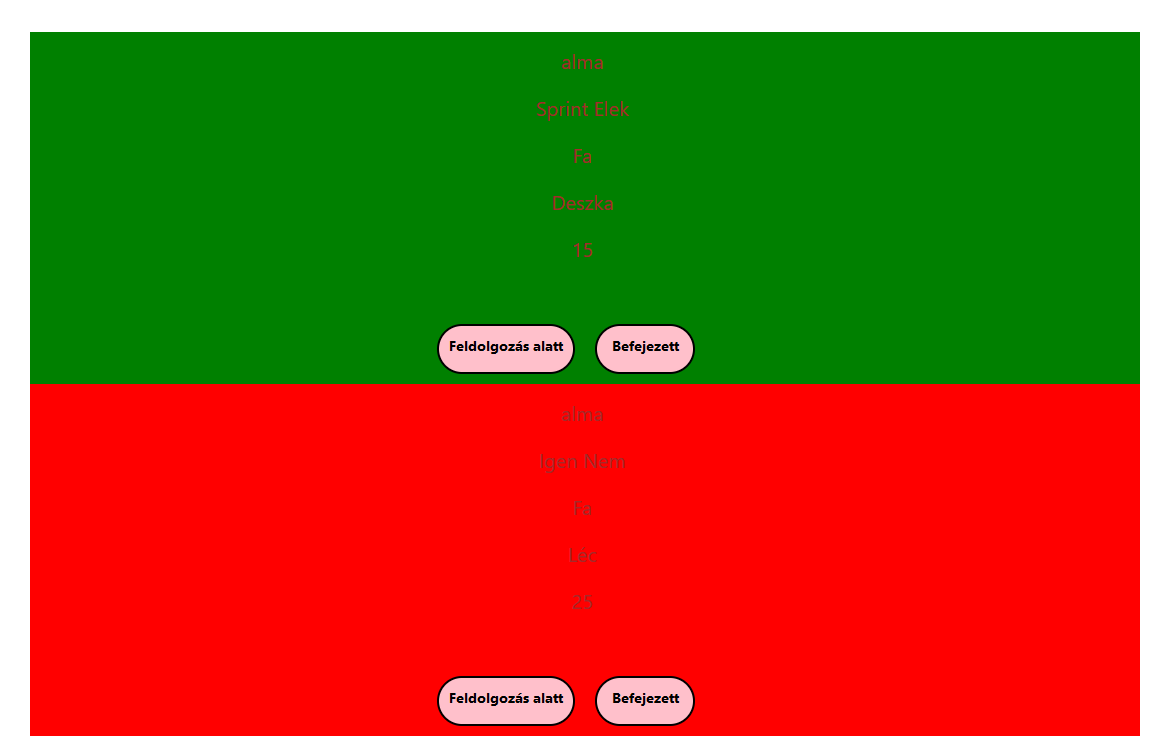
****

**Rendelés leadás:**

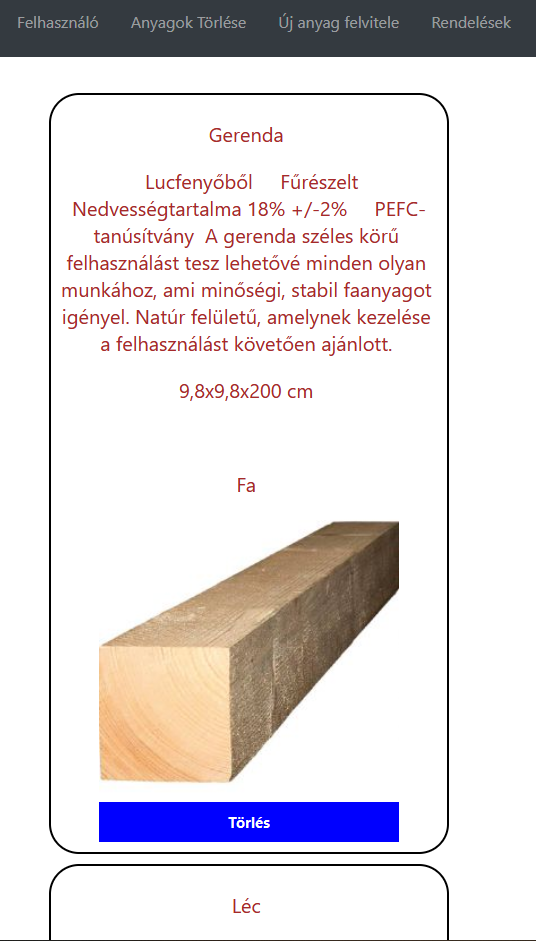
A mezők kitöltése után a gombra kattintva az adatokat feltöltődnek az adatbázisba.

**Rendelések:**

A leadott rendelések láthatóak a feldolgozás alatt, és a befejezett gombokkal jelölhetjük a rendelések állapotát.



**Anyagok törlése:**

A törlés gomb megnyomásával a kiválasztott terméket törölhetjük a termék listáról. 

# Fejlesztői dokumentáció

## Témaválasztás indoklása

mit csinál a program röviden, miért pont ezt választottad, biztonsági kérdések ( almenüre hivatkozásnál le van védve például-az a session-ös cucc, amit beleraktunk)

## Alkalmazott fejlesztői eszközök

Visual Studio Code,

XXAMP, PHP Myadmin, MYSQL

React Native Expo

Backend: Express.js

A backend és a frontend a programozás során javascript nyelven történt.

## Rendszerterv

### Adatmodell leírása

**Roles tábla:**

A roles tábla 4 oszlopból áll. Az első mező az id. Ez a mező int típusú és elsődleges kulcsot tartalmaz más táblákhoz való csatolás érdekében. A name mező a felhasználók között használt rangokat tartalmazza varchar típusban magyar nyelv kódolással. A createdAt és updatedAt datetime típusú mezők. Azt a célt szolgálják, hogy az új felhasználó pontos dátumát és idejét jelenítsék meg vagy a módosított felhasználóét.



**User\_roles tábla:**

A users tábla 4 oszlopból áll. A createdAt és updatedAt datetime típusú mezők. Azt a célt szolgálják, hogy az új felhasználó pontos dátumát és idejét jelenítsék meg vagy a módosított felhasználóét. A roleId mező int típusú egész számok felvételére képes mely magában hordozza a felhasználó rangját. A userId a felhasználó azonosítóját tartalmazza az int típusú mezőben, egyéni kulcsot is kapott.



**Users tábla:**

A users tábla 6 oszlopot tartalmaz. Az id mező a felhasználó azonosítója és a tábla elsődleges kulcsa, valamint int típusú egész számokat tárol és auto\_increment-tel rendelkezik. A username mező a felhasználó nevét tartalmazza varchar típusban magyar nyelv kódolással. Az email mező a felhasználó email-jét tartalmazza varchar típusban magyar nyelv kódolással. A password mező a felhasználó jelszavát tartalmazza varchar típusban magyar nyelv kódolással. A createdAt és updatedAt datetime típusú mezők. Azt a célt szolgálják, hogy az új felhasználó pontos dátumát és idejét jelenítsék meg vagy a módosított felhasználóét.



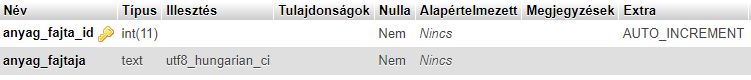
**Anyag tábla:**

A tábla 7 oszlopban különíti el az adatokat. Az anyag\_id auto\_incrementet kapott a gyorsabb megkülönböztethetőség érdekében, int típusú. Az anyag\_neve és az anyag\_kep mezői varchar típusúak, az egyik 100, a másik 255 karakter tárolására alkalmas. Az anyag\_leiras, anyag\_fajtaja és az anyag\_merete text típusú oszlopok. Az anyag\_ar int típusú oszlop.

### 

### Anyagfajták tábla:

A tábla 2 oszloppal rendelkezik. Az első oszlop auto\_increment-et kapott és elsődleges kulccsal bír, int típusú. Az anyag\_fajtaja oszlop text típusú.

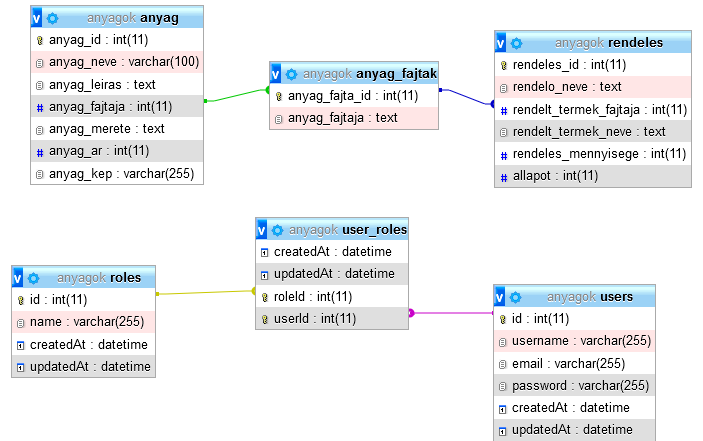


**Rendelések tábla:**

A rendelések tábla 6 oszlopának mezői tárolják el a rendelések adatait. Az első oszlop a rendeles\_id mely auto\_increment-et kapott és elsődleges kulcsot. A rendelo\_neve, a rendelt\_termek\_fajtaja és a rendelt\_termek\_neve text típusúak. A rendeles\_mennyisege és az allapot oszlop mezői int típusúak.



**A táblák közötti kapcsolatok:**

****

### Program leírása

használt php és egyéb js fájlok és feladataik leírása, hierarchia bemutatása rajzzal, kikopizhatod pl. commanderből, vscode-ból, hogy egy-egy mappádon belül milyen fájlok vannak, rövid leírás.

|  |  |
| --- | --- |
| mobilback.PNG | mobilfront.PNG |
| webback.PNG | webfront.PNG |

## Részletes feladatspecifikáció, algoritmusok

**Kalkulátor forrása**

Elérési út: mobilfrontend/feladat/Kalkulator.js

webanyagokfrontend/src/components/sajatosztalyok/Kalkulator.js

export default class PizzaTranslator extends Component {

constructor(props) {

super(props);

this.state = {text: ''};

}

state = {

hosszusag: '',

magassag: '',

szelesseg: '',

ered: '',

}

magassagkezel = (text) => {

this.setState({ magassag: text })

}

szelessegkezel = (text) => {

this.setState({ szelesseg: text })

}

hosszusagkezel = (text) => {

this.setState({ hosszusag: text })

}

szamitas = (hosszusag, szelesseg, magassag) =>{

var eredmeny = parseInt(hosszusag)\*parseInt(szelesseg)\*parseInt(magassag);

this.setState({ered: eredmeny})

}

render() {

return (

<View style={{padding: 20, marginLeft:"auto", marginRight: "auto", borderRadius: 15, borderColor: "blue", borderWidth: 5, marginTop: 20, minWidth: 450, minHeight: 450}}>

<Text style={{fontSize: 24, fontWeight: "bold", textAlign: "center", marginBottom: 20, textDecorationLine:"underline", textTransform:"uppercase"}}>Betonalap mennyiségének kiszámítása</Text>

<View style={{minHeight: 200, minWidth: 400, marginLeft:"auto", marginRight: "auto"}}>

<View style={{flex: 1, flexDirection: "row",}}>

<Text style={{padding: 10,marginBottom: 5 , fontSize: 25}}>

Magasság (m):

</Text>

<TextInput

style={{ height: 40, borderColor: 'black', borderWidth: 3, borderRadius: 25, width: 200, marginRight: "auto", textAlign:"center", fontSize: 20}}

onChangeText={this.magassagkezel}

value={this.state.magassag}

/>

</View>

<View style={{flex: 1, flexDirection: "row",}}>

<Text style={{padding: 10, marginBottom: 5 , fontSize: 25}}>

Szélesség (m):

</Text>

<TextInput

style={{ height: 40, borderColor: 'black', borderWidth: 3, borderRadius: 25, width: 200, marginRight: "auto", textAlign:"center", fontSize: 20}}

onChangeText={this.szelessegkezel}

/>

</View>

<View style={{flex: 1, flexDirection: "row",}}>

<Text style={{padding: 10,marginBottom: 5 , fontSize: 25}}>

Hosszúság (m):

</Text>

<TextInput

style={{ height: 40, borderColor: 'black', borderWidth: 3, borderRadius: 25, width: 200, marginRight: "auto", textAlign:"center", fontSize: 20 }}

onChangeText={this.hosszusagkezel}

/>

</View>

</View>

<View style={{marginLeft:"auto", marginRight: "auto", flex: 1, flexDirection: "row"}}>

<TouchableOpacity style={{marginTop: 15, fontSize: 25, backgroundColor: "blue", borderRadius: 25, width: 200, height: 80, padding: 20}}

onPress={()=> this.szamitas(this.state.magassag, this.state.szelesseg, this.state.hosszusag)}>

<Text style={{textAlign: "center", color: "white", fontWeight: "bold", fontSize: 25}}> Számítás </Text>

</TouchableOpacity>

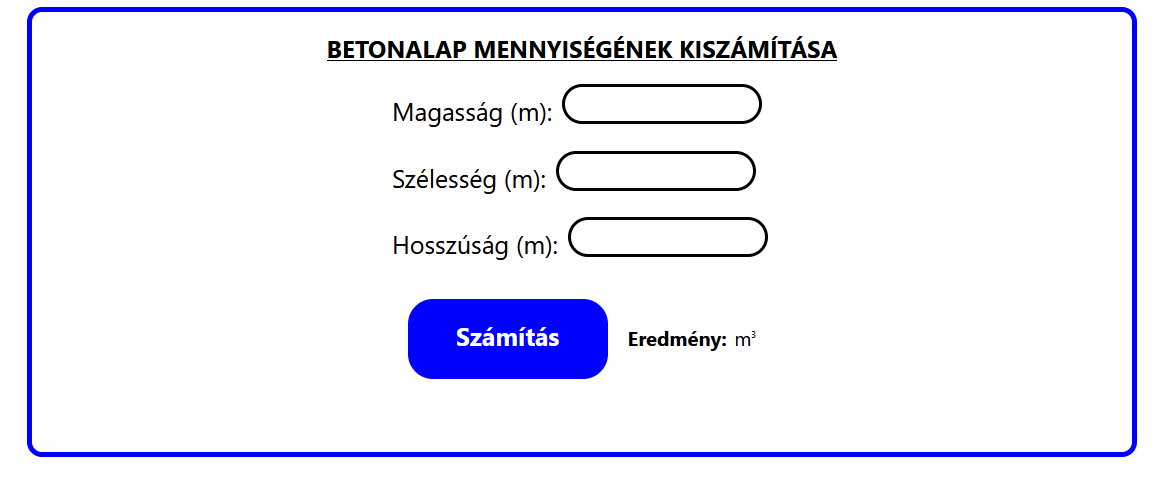
<Text style={{fontSize: 20, fontWeight: "bold", marginTop: 40, marginLeft: 20}}>Eredmény: </Text>

<Text style={{fontSize: 20, marginTop: 40,}}>{this.state.ered}</Text>

<Text style={{fontSize: 20, marginTop: 40,marginLeft: 2}}>m</Text>

<Text style={{fontSize:10, marginTop: 40,lineHeight: 20}}>3</Text>

</View>

</View>);}}

A kód feladata, hogy a beviteli mezőkbe a felhasználó által írt adatokat szövegként a statekbe töltse mivel ezek változó értékek. Az értékeket kezelés után a szamolas nevű függvényben konvertálja számmá és elvégzi a terület kiszámítását. A számítás akkor megy végbe ha a felhasználó megnyomja a ,,Számítás” felíratú TouchableOpacity-t (nyomható elem) mely során az onPress funkcíó meghívja a szamitas függvényt gombnyomásra. A szorzás eredményét egy Text objektumba helyezi hivatkozás által (this.state.ered).

**Anyagok törlésének forrása**

Elérési út: webanyagokfrontend/components/sajatosztályok/Anyagtorles.js

export default class FetchExample extends React.Component {

constructor(props){

super(props);

this.state ={ isLoading: true}

}

szavazat=(szam)=>{

//alert(szam)

var bemenet={

bevitel1:szam

}

fetch("http://localhost:8080/anyagtorles", {

method: "POST",

body: JSON.stringify(bemenet),

headers: {"Content-type": "application/json; charset=UTF-8"}

}

)

.then(x => x.text())

.then(y => alert(y));

window.location.reload();

this.setState({});

}

componentDidMount(){

return fetch('http://localhost:8080/anyagok')

.then((response) => response.json())

.then((responseJson) => {

this.setState({

isLoading: false,

dataSource: responseJson,

}, function(){

});

})

.catch((error) =>{

console.error(error);

});

}

render(){

if(this.state.isLoading){

return(

<View style={{flex: 1, padding: 20}}>

<ActivityIndicator/>

</View>

)

}

return(

<View style={{flex: 1, paddingTop:20}}>

<FlatList

data={this.state.dataSource}

renderItem={({item}) =>

<View style={{width: 400, marginBottom: 10, borderColor: "black", borderRadius: 30, borderWidth: 2, marginLeft: "auto", marginRight: "auto", padding: 10}}>

<Text style={{color:"brown",fontSize:20,textAlign:"center",marginTop:15,marginBottom:5}} >{item.anyag\_neve} </Text>

<Text style={{color:"brown",fontSize:20,textAlign:"center",marginTop:15,marginBottom:5}} >{item.anyag\_leiras} </Text>

<Text style={{color:"brown",fontSize:20,textAlign:"center",marginTop:15,marginBottom:5}} >{item.anyag\_merete} </Text>

<Text style={{color:"brown",fontSize:20,textAlign:"center",marginTop:15,marginBottom:5}} >{item.anyag\_ar} </Text>

<Text style={{color:"brown",fontSize:20,textAlign:"center",marginTop:15,marginBottom:5}} >{item.anyag\_fajtaja} </Text>

<Image source={{uri: 'http://localhost:8080/'+item.anyag\_kep}} style={{width:300,height:300,marginLeft:"auto",marginRight:"auto"}} />

<TouchableOpacity

style={styles.kekgomb}

onPress={async ()=>this.szavazat(item.anyag\_id)}

>

<Text style={{color:"white",fontWeight:"bold",fontSize:15}} >Törlés</Text>

</TouchableOpacity>

</View>

}

keyExtractor={({anyag\_id}, index) => anyag\_id}

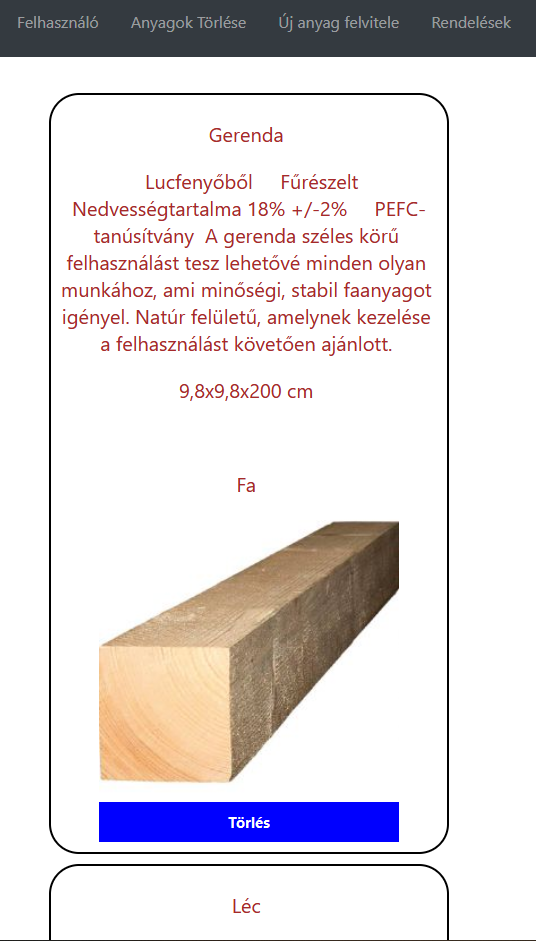
/>

</View>

);

}

}



Az oldal működése fetcheken alapszik melyek azt teszik lehetővé, hogy az adatokat külső helyről érhessük el (itt kettőre volt szükség mivel lekérdezzük az adatbázisból az információkat (GET) és a törlés által módosítjuk (POST) is őket). A törlés gomb megnyomásakor meghívjuk a szavazat nevű függvényt ami tartalmazza a törlést végrehajtó fetchet. A gomb megnyomáskor a kiválasztott áru adatbázis szerinti id-t tárolja el majd adja át a bevitel változónak. A művelet végrehajtása után pedig az oldal újratöltésre kerül a window.location.reload() segítségével.

**Rendelések forrása**

Elérési út: webanyagokfrontend/components/sajatosztalyok/Rendelesek.js

export default class FetchExample extends React.Component {

constructor(props){

super(props);

this.state ={ isLoading: true}

}

allapotvaltfel=async (allapot\_id)=>{

alert(allapot\_id)

let bemenet={

bev1:allapot\_id

}

fetch('http://localhost:8080/allapot\_valtoztat\_oke',{

method: "POST",

body: JSON.stringify(bemenet),

headers: {"Content-type": "application/json; charset=UTF-8"}

}

)

.then((response) => response.text())

.then((szoveg) => {

alert(szoveg)

window.location.reload();

this.setState({});

})

}

allapotvaltle=async (allapot\_id)=>{

alert(allapot\_id)

let bemenet={

bev1:allapot\_id

}

fetch('http://localhost:8080/allapot\_valtoztat\_nemoke',{

method: "POST",

body: JSON.stringify(bemenet),

headers: {"Content-type": "application/json; charset=UTF-8"}

}

)

.then((response) => response.text())

.then((szoveg) => {

alert(szoveg)

window.location.reload();

this.setState({});

})

}

componentDidMount(){

return fetch('http://localhost:8080/rendelesek')

.then((response) => response.json())

.then((responseJson) => {

this.setState({

isLoading: false,

dataSource: responseJson,

}, function(){

});

})

.catch((error) =>{

console.error(error);

});

}

render(){

if(this.state.isLoading){

return(

<View style={{flex: 1, padding: 20}}>

<ActivityIndicator/>

</View>

)

}

return(

<View style={{flex: 1, paddingTop:20}}>

<FlatList

data={this.state.dataSource}

renderItem={({item}) =>

<View>

{ item.allapot == 0

?

( <View><Text style={{color:"brown",fontSize:20,textAlign:"center",marginTop:15,marginBottom:5}} >alma </Text>

<Text style={{color:"brown",fontSize:20,textAlign:"center",marginTop:15,marginBottom:5}} >{item.rendelo\_neve} </Text>

<Text style={{color:"brown",fontSize:20,textAlign:"center",marginTop:15,marginBottom:5}} >{item.rendelt\_termek\_fajtaja} </Text>

<Text style={{color:"brown",fontSize:20,textAlign:"center",marginTop:15,marginBottom:5}} >{item.rendelt\_termek\_neve} </Text>

<Text style={{color:"brown",fontSize:20,textAlign:"center",marginTop:15,marginBottom:5}} >{item.rendeles\_mennyisege} </Text>

<Text style={{color:"brown",fontSize:20,textAlign:"center",marginTop:15,marginBottom:5}} >{item.anyag\_fajtaja} </Text>

<View style={{flexDirection:"row"}}>

<View style={{flex: 1, padding: 10}}>

<TouchableOpacity

onPress={async ()=>this.allapotvaltfel(item.rendeles\_id)}>

<View style={styles.gomb1}>

<Text style={styles.gombSzoveg}>Feldolgozás alatt</Text>

</View>

</TouchableOpacity>

</View>

<View style={{flex: 1, padding: 10}}>

<TouchableOpacity

onPress={async ()=>this.allapotvaltle(item.rendeles\_id)}>

<View style={styles.gomb}>

<Text style={styles.gombSzoveg}>Befejezett</Text>

</View>

</TouchableOpacity>

</View>

</View>

</View>

)

:

null

}

{ item.allapot == 1

?

( <View style={{backgroundColor:"green"}}><Text style={{color:"brown",fontSize:20,textAlign:"center",marginTop:15,marginBottom:5}} >alma </Text>

<Text style={{color:"brown",fontSize:20,textAlign:"center",marginTop:15,marginBottom:5}} >{item.rendelo\_neve} </Text>

<Text style={{color:"brown",fontSize:20,textAlign:"center",marginTop:15,marginBottom:5}} >{item.rendelt\_termek\_fajtaja} </Text>

<Text style={{color:"brown",fontSize:20,textAlign:"center",marginTop:15,marginBottom:5}} >{item.rendelt\_termek\_neve} </Text>

<Text style={{color:"brown",fontSize:20,textAlign:"center",marginTop:15,marginBottom:5}} >{item.rendeles\_mennyisege} </Text>

<Text style={{color:"brown",fontSize:20,textAlign:"center",marginTop:15,marginBottom:5}} >{item.anyag\_fajtaja} </Text>

<View style={{flexDirection:"row"}}>

<View style={{flex: 1, padding: 10}}>

<TouchableOpacity

onPress={async ()=>this.allapotvaltfel(item.rendeles\_id)}>

<View style={styles.gomb1}>

<Text style={styles.gombSzoveg}>Feldolgozás alatt</Text>

</View>

</TouchableOpacity>

</View>

<View style={{flex: 1, padding: 10}}>

<TouchableOpacity

onPress={async ()=>this.allapotvaltle(item.rendeles\_id)}>

<View style={styles.gomb}>

<Text style={styles.gombSzoveg}>Befejezett</Text>

</View>

</TouchableOpacity>

</View>

</View>

</View>

)

:

null

}

{ item.allapot == 2

?

( <View style={{backgroundColor:"red"}}><Text style={{color:"brown",fontSize:20,textAlign:"center",marginTop:15,marginBottom:5}} >alma </Text>

<Text style={{color:"brown",fontSize:20,textAlign:"center",marginTop:15,marginBottom:5}} >{item.rendelo\_neve} </Text>

<Text style={{color:"brown",fontSize:20,textAlign:"center",marginTop:15,marginBottom:5}} >{item.rendelt\_termek\_fajtaja} </Text>

<Text style={{color:"brown",fontSize:20,textAlign:"center",marginTop:15,marginBottom:5}} >{item.rendelt\_termek\_neve} </Text>

<Text style={{color:"brown",fontSize:20,textAlign:"center",marginTop:15,marginBottom:5}} >{item.rendeles\_mennyisege} </Text>

<Text style={{color:"brown",fontSize:20,textAlign:"center",marginTop:15,marginBottom:5}} >{item.anyag\_fajtaja} </Text>

<View style={{flexDirection:"row"}}>

<View style={{flex: 1, padding: 10}}>

<TouchableOpacity

onPress={async ()=>this.allapotvaltfel(item.rendeles\_id)}>

<View style={styles.gomb1}>

<Text style={styles.gombSzoveg}>Feldolgozás alatt</Text>

</View>

</TouchableOpacity>

</View>

<View style={{flex: 1, padding: 10}}>

<TouchableOpacity

onPress={async ()=>this.allapotvaltle(item.rendeles\_id)}>

<View style={styles.gomb}>

<Text style={styles.gombSzoveg}>Befejezett</Text>

</View>

</TouchableOpacity>

</View>

</View>

</View>

)

:

null

}

</View>

}

keyExtractor={({anyag\_id}, index) => anyag\_id}

/>

</View>

);

}

}

A kód azt a célt szolgálja, hogy a két fetch által kapott adatokat először megjeleníti majd gombnyomásra változtatja a rendelések állapotát és hogy ez látványos legyen az oldal automatikusan újratölt. Az oldal tartalma egy if –es feltételes utasításban található. A feltétel az allapot\_id-t figyeli. A ,,Befejezett” és ,,Feldolgozás alatt” nevű gombok megnyomásakor meghívjuk az allapotvaltle/fel függvényt melyek a gombok által eltárolt id-ket megkapják és a nekik megfelelő fetch által megváltoztatják a rendelések állapotát, hogy ez látványos legyen a View-k stílusában különböző tulajdonságokat kaptak, hogy a felhasználó megbizonyosodhasson a változásokról.

## Tesztelés

A mobilalkalmazást Huawei P20, Huawei MediaPad M3 és Redmi 9T készülékeken teszteltem, a reszponzivitás látványosságáért. A web alkalmazás Google Chrome, Mozilla Firefox és Operában teszteltek.

## Továbbfejlesztési lehetőségek

A következő tovább fejlesztési lehetőségek merültek fel bennem a záródolgozatommal kapcsolatban:

* termék nézegető egy külön kiugró ablakban
* árak változásáról egy külön lap a felhasználóknak
* felhasználónként kezelt rendelések mobil és web oldalról

# Összegzés

A szakdolgozatom készítése közben rengeteg újat tanultam a REACT-ról, mivel ebben készítettem a munkám. Igyekeztem minél hasznosabbá tenni a mobil applikációmat és a weblapomat a hétköznapi emberek számára, akik kevésbé tájékozottak, hogy minél egyszerűbben és gyorsabban oldhassák meg ebben a témában a felmerülő problémájukat.

# Irodalomjegyzék

tankönyvek php, css, html, sql, javascript

netesforrasok, lap címe es megnevezése, linkeket nyugodtan kopizz ide!!!

# Eredetiségnyilatkozat

AlulírottSaját Név nyilatkozom, hogy záró dolgozatom a saját szellemi termékem.

Debrecen, 2018. márc. 28

.

aláírás