**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:**

„Mit üzen neked?” mobil app és weblap fejlesztés

**Záródolgozat készítője:**

Mező Tamás Zsolt

**Konzulens: Dr. Várbíróné Nahaji Anikó**

**Konzultációs időpontok:**

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

**Debrecen, 2023.**

Tartalomjegyzék

[Bevezetés 3](#_Toc32482713)

[Felhasználói dokumentáció 4](#_Toc32482714)

[Feladatspecifikáció 4](#_Toc32482715)

[Rendszerkövetelmények 4](#_Toc32482716)

[Program telepítése, elérése 5](#_Toc32482717)

[Program használatának leírása 7](#_Toc32482718)

[Fejlesztői dokumentáció 17](#_Toc32482719)

[Témaválasztás indoklása 17](#_Toc32482720)

[Alkalmazott fejlesztői eszközök 18](#_Toc32482721)

[Rendszerterv 18](#_Toc32482722)

[Adatmodell leírása 18](#_Toc32482723)

[Program leírása 21](#_Toc32482724)

[Részletes feladatspecifikáció, algoritmusok 25](#_Toc32482725)

[Tesztelés 99](#_Toc32482726)

[Továbbfejlesztési lehetőségek 102](#_Toc32482727)

[Összegzés 102](#_Toc32482728)

[Irodalomjegyzék 102](#_Toc32482729)

[Eredetiségnyilatkozat 103](#_Toc32482730)

# Bevezetés

A záródolgozatom egy vallással kapcsolatos alkalmazás, melyben Bibliai igéket lehet olvasni, leírni, hogy mit üzen számodra. Azért választottam ezt a témát, mert szeretek templomba járni.

Az applikációban és a weblapon is igéket lehet olvasni, és ha rányomtam a szövegre, akkor az ige jelenik meg, és szövegdobozba be kell írni, hogy mit jelent számodra a választott ige és mi az üzenete és el lehet küldeni azt. Mindkettőnek van keresés funkciója. Admin felületen lehet rendezni, ellenőrizni, hogy minden rendben van-e, statisztika nyomon követése, a nem odaillő tartalom törlése, a rossz kategóriában szereplő tartalom áthelyezése a megfelelőbe. Itt is lehet igéket írni. Nem bejelentkezett felhasználók nem törölhetnek. Az igék alatt megjelenik az időpont, a Bibliai igehely (könyv, fejezet és vers) és hogy milyen kategóriában van. Az idézetek és vélemények felett a sorszáma látható. Kategóriákat a legördülő listából lehet kiválasztani (pl: ünnep, tanítások). El lehet mondani, hogy mit üzen számunkra egy-egy idézet, hol hallottuk stb. Új igét fel lehet vinni a rendszerbe a felvitel menüpont segítségével.

Az alkalmazást azoknak ajánlom, akik szeretnek Bibliát olvasni, és meg akarják osztani másokkal a gondolataikat és üzenetüket az igéről.

# Felhasználói dokumentáció

## Feladatspecifikáció

**Mobil applikáció funkciói:**

**Felhasználó:**

* Keresés a vélemények és igék között
* Vélemény írása
* Könyv, fejezet és vers megjelenítése az Igénél
* Statisztika nyomon követése (vélemény, idézet) külön-külön
* Kategória
* Ige felvitele
* Az idézetek és vélemények felett azonosító.

## Rendszerkövetelmények

### Mobil applikáció:

* Internet szükséges(mobilos applikáció)
* Android
* Bármelyik népszerű böngésző (Chrome, Firefox)

### Web:

* A webeshez nem szükséges internet elérés.
* Windows, Linux
* Bármelyik népszerű böngésző
* Legalább 4Gb RAM
* Legalább 2GB szabad lemezterület, de nem árt több is.

## Program telepítése, elérése

A programot a githubon lehet elérni, melynek elérhetőségei:

Mobil:

<https://github.com/mezotamas/mobilfrontend>

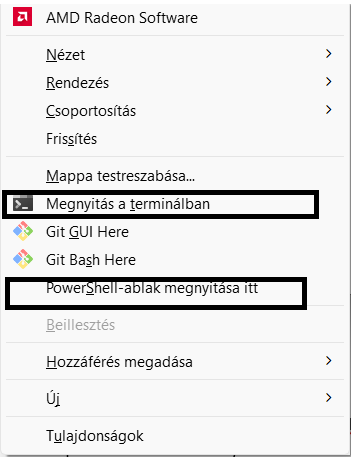
Weblap:

<https://github.com/mezotamas/webfrontend>

<https://github.com/mezotamas/webbackend>

A lépések:

1. Telepíteni kell a node.js programot, ami a frontend és a backend működéséhez kell. <https://nodejs.org/en>, és az lts verziót ajánlott. Ha a node.js-sel probléma adódna akkor ellenőrizni kell a környezeti változókat. pl PATH változóban szerepel-e a „C:\Program Files\nodejs” vagy ahová telepítve van a program.
2. Telepíteni kell az XAMPP nevű programot, ami tartalmazza az Mysql/phpMyAdmin adatbáziskezelőt <https://www.apachefriends.org/hu/download.html>, ebből a 8.1.12 -es verziót.
3. Telepíteni kell a Git programot <https://git-scm.com/downloads>.
4. A programok telepítésének ideje függ a gép teljesítményétől. Az xampp esetén többet kell várni. Ezek után a gépet újra kell indítani.
5. Meg kell nyitni Fájlkezelőt, utána egy tetszőleges könyvtába belépve, vagy egy új mappa. Utána Shiht+jobb egérgomb Megnyitás Terminálban vagy PowerShell-ablak nyitása itt opció kiválasztása.



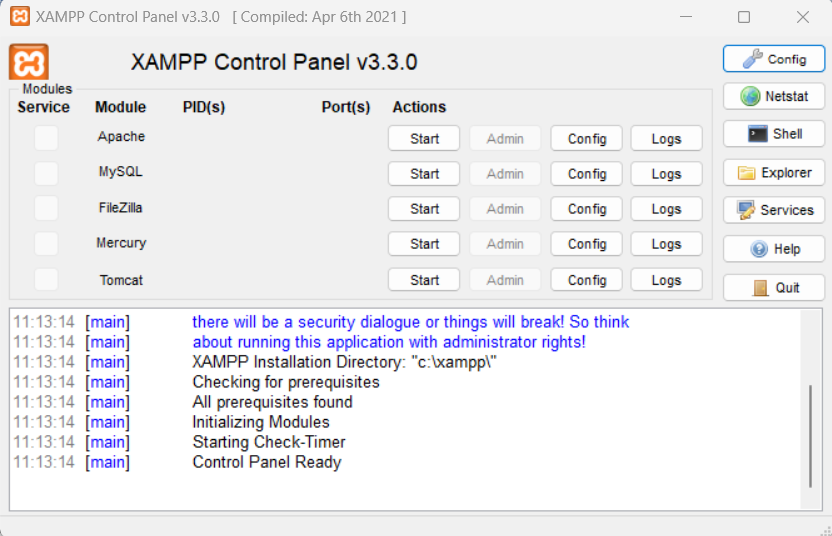
1. A fentebb található Github hivatkozás a git clone https://github.com/felhasznalo/repository parancssal lehet lementeni például git clone <https://github.com/mezotamas/webbackend>, majd a cd paranccsal belépni a mappába pl.: cd webbackend/
2. A webfrontend és a webbackend egy mappában kell lennie pl:

Vallas/webfrontend  
Vallas/webbackend

1. Mindkét esetben az npm i –force parancsot kell kiadni, a függőségek telepítéséhez. Ez igénybe vehet néhány percet (függ a sebességtől és a gép teljesítményétől).

A Windows 11 esetén a terminálban vannak fülek, úgy ahogy a böngészőben is. Itt a + gombbal lehet új terminált nyitni. Az egyikben a backend a másikban a frontend mappa van.

1. XAMPP programot el kell indítani.



Itt az Apache-t és a Mysql-t el indítani a Start gomb segítségével. Majd az aktívvá váló Admin gombra megnyílik a phpMyAdmin a böngészőben.

Az importálás menüben be kell importálni a vallas.sql fájlt.

1. Először a backendet kell utána a frontendet. A backend indításához a node server.js parancssot kell kiadni és a frontendhez pedig a npm start parancsot kell kiadni.

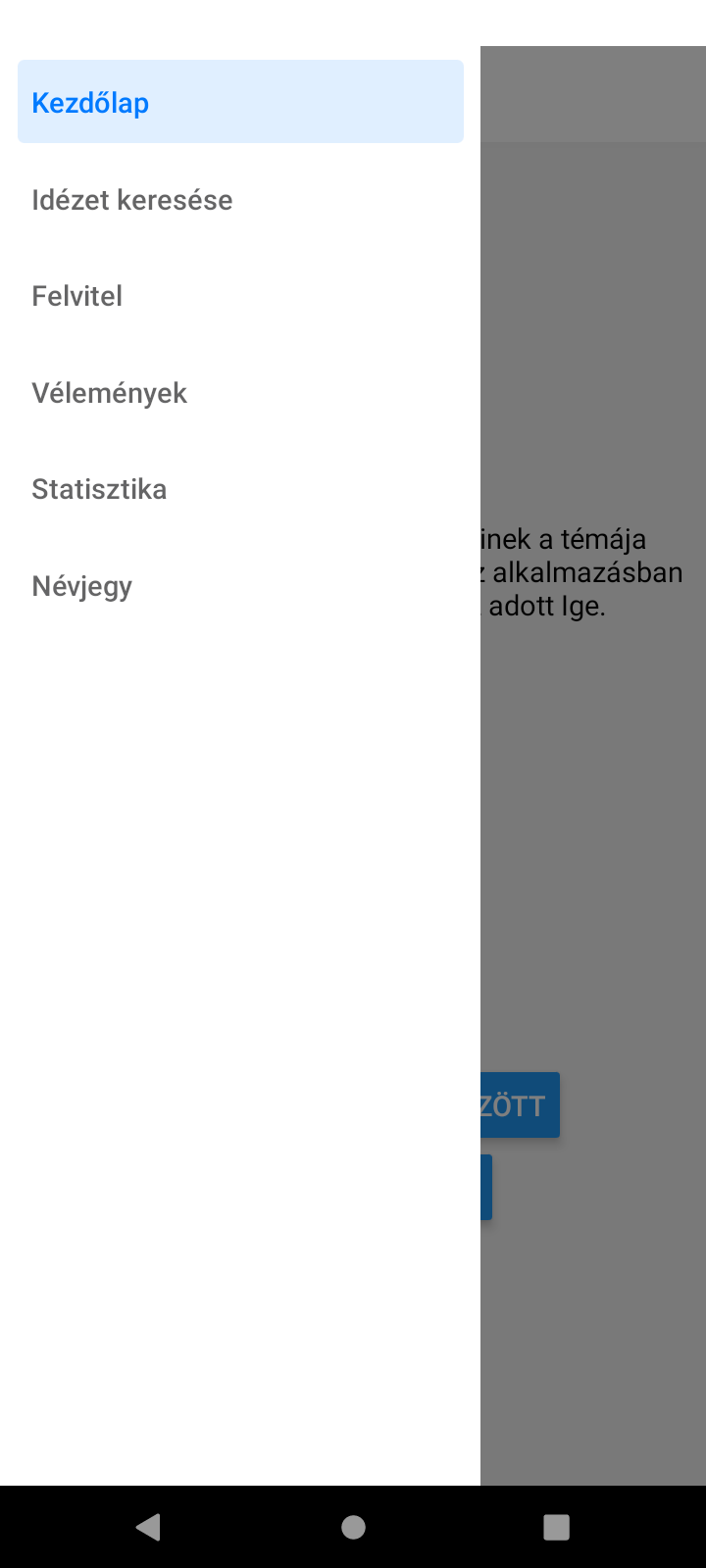
Ha mindent jól csináltunk elindul a webes felület és lehet használni. Ammenyiben nem nyílik meg automatikusan a böngészö, akkor el kell indítani és a címsorba a <http://localhost:8081/> címet kell beírni.

A webeshez nem kell internet a telepítés után, mert ez a localhoston kommunikál, így működik offline, ha minden problémamentesen zajlott le.

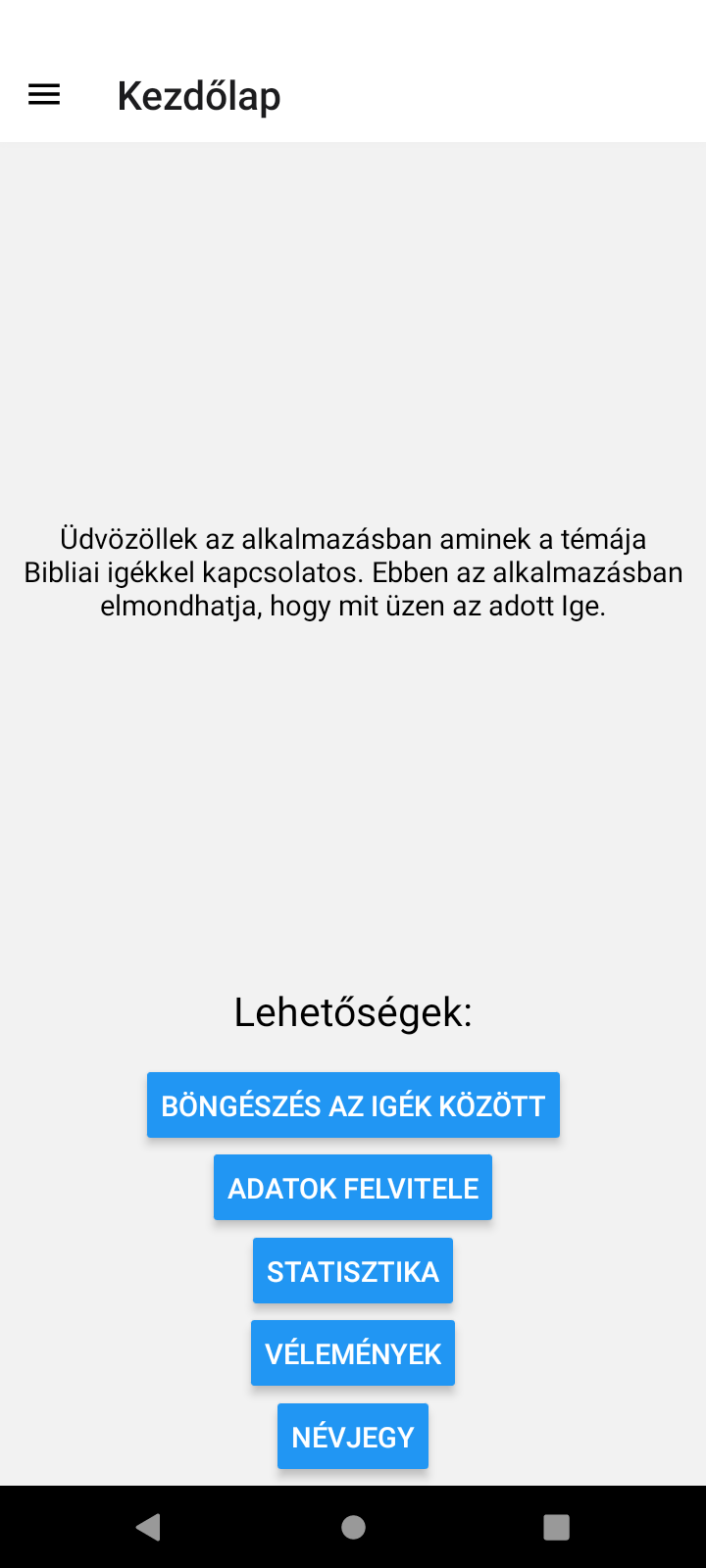
## Program használatának leírása

**Mobil applikáció leírása**

A mobil applikációban 6 menüpont van:

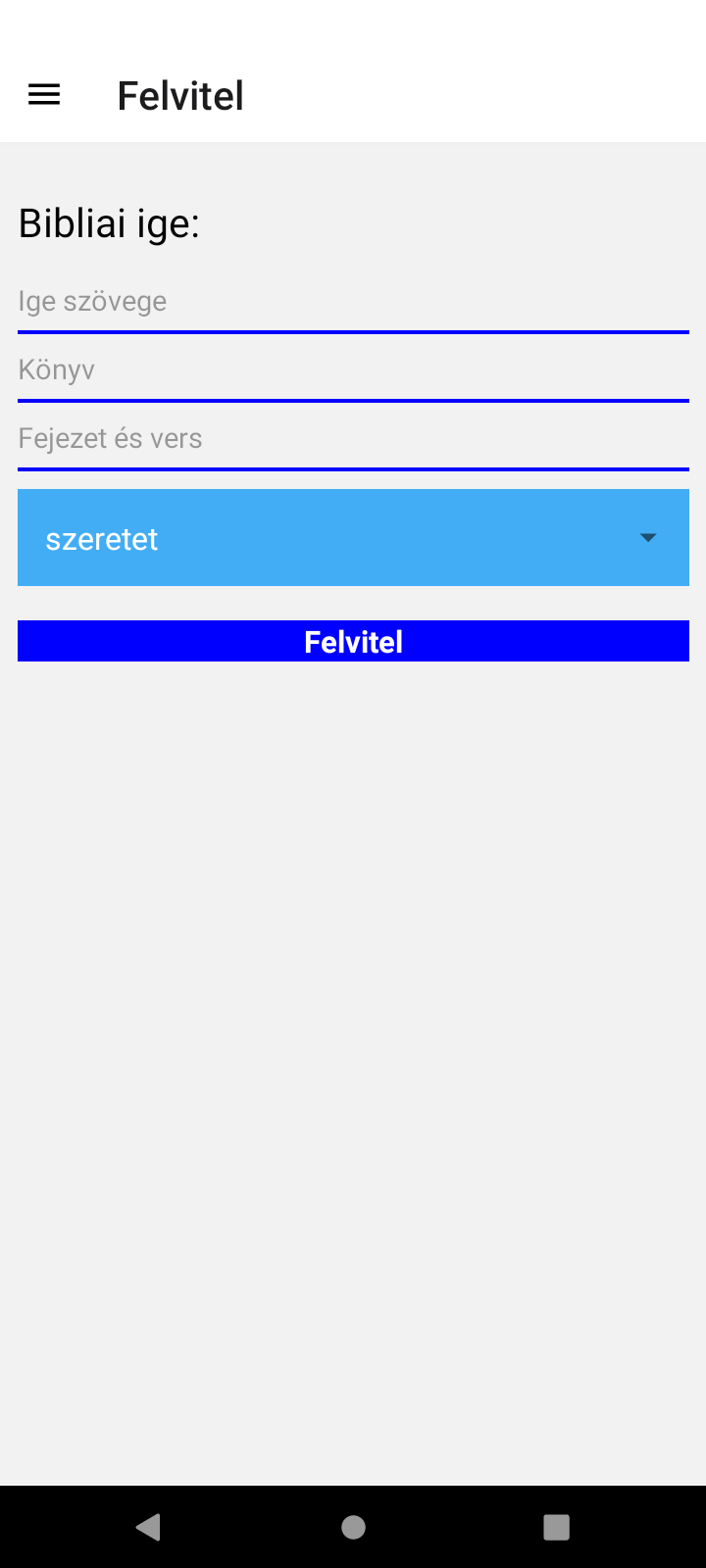


Kezdőlap, Idézet keresése, Felvitel, Vélemények, Statisztika, Névjegy

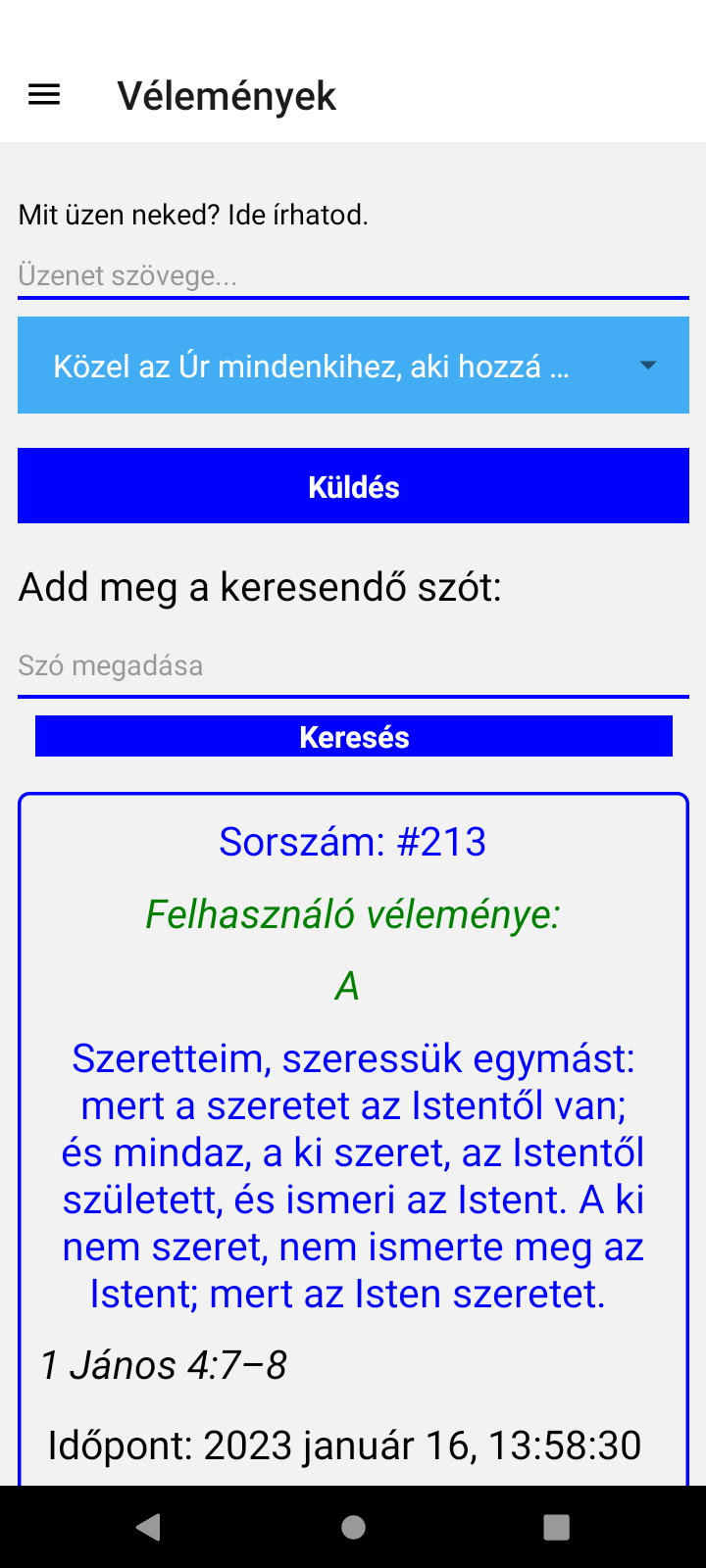


A kezdőlap menüben üdvözlőszöveg, alatta a lehetőségek gombjai találhatóak. Ebben a menüpontban lehet ugrani a menükben, a vissza gombbal lehet a kezdőlapra térni.

  
Az Idézet keresésben meg lehet tekinteni az eddigi idézeteket időponttal, kategóriával, Bibliai hellyel ellátva, a szövegbeviteli mezőbe be lehet írni a kulcsszót, például szeretet. Ha megnyomják a gombot, akkor csak azokat az igéket jeleníti meg amiben szerepel a keresett szó.

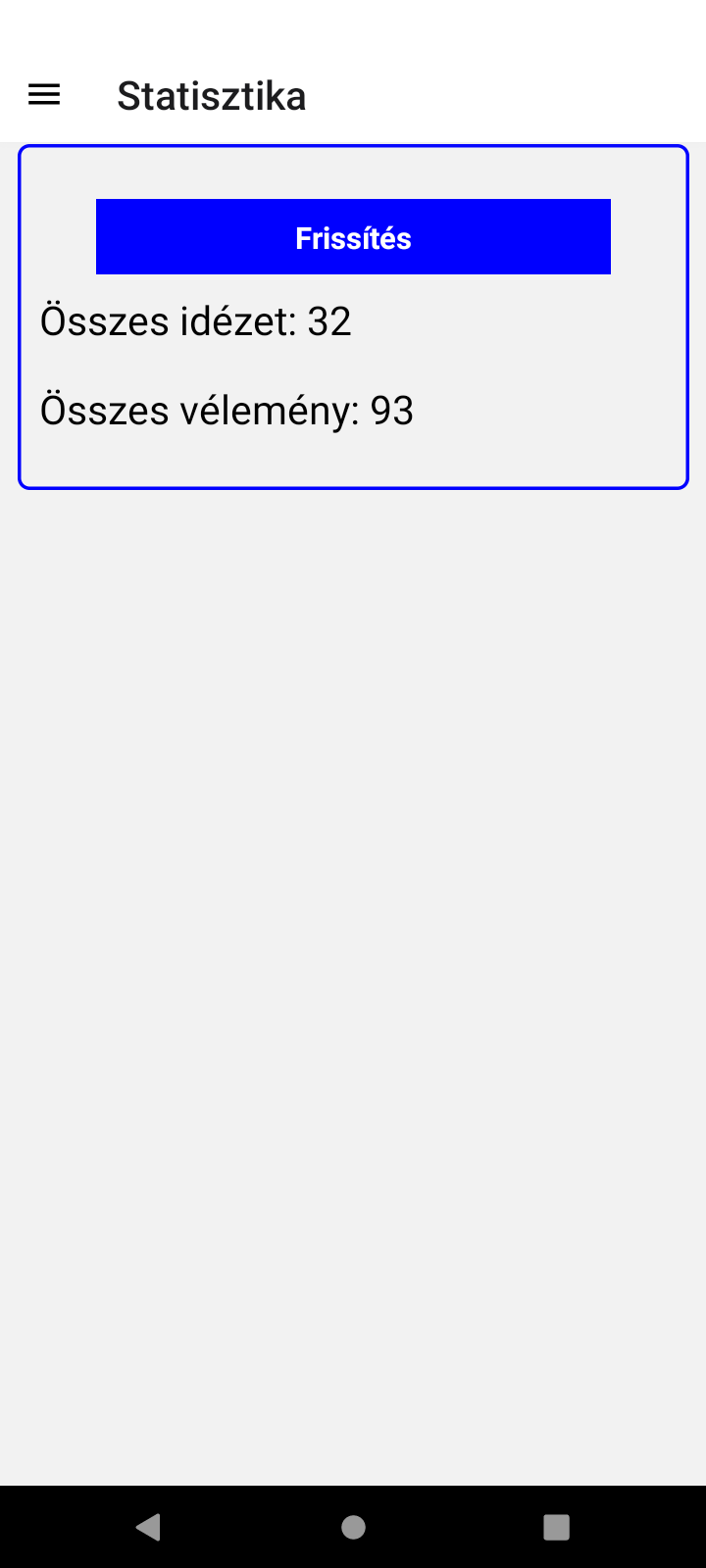


A Felvitel menüpontban adatokat lehet felvinni a rendszerbe a következő sorrendben: idézet szövege könyv, fejezet és vers, a lenyílóban pedig a kategóriát. A Felvitel gomb megnyomásával megtörténik az adatok felvitele.



A Vélemények menüpontban le lehet írni, hogy mit üzen számunkra az ige. A fent található szövegbeviteli mezőben le lehet írni az üzenetet, a lenyílóban pedig a kiválasztott ige.   
A küldés gomb megnyomásával lehet elküldeni az üzenetet. Az üzenet alább jelennek meg.

Ha üresen nyomják meg a gombot a felvitelnél és a vélemény írásánál, akkor egy üzenetet kap a felhasználó, hogy nem maradhat üresen, ha pedig a keresésnél hagyják üresen akkor az összes adatot megjeleníti.

  
A statisztika menüben meg lehet tekinteni a rendszerben található idézetek és vélemények számát külön-külön. A frissítéshez meg kell nyomni a Frissítés gombot, és frissülnek az adatok.

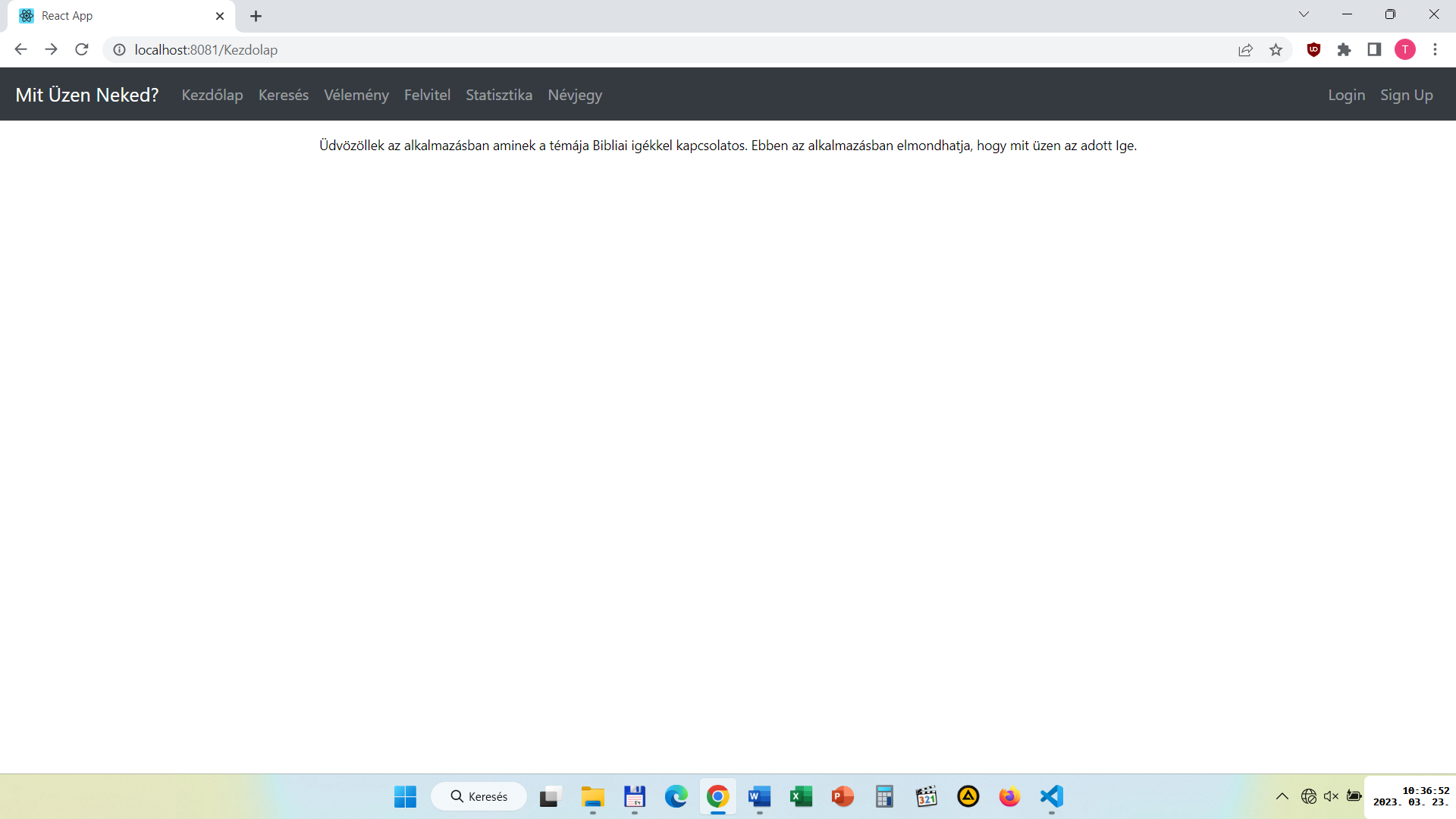


A névjegy menüpontban pedig a készítő és az intézmény neve.

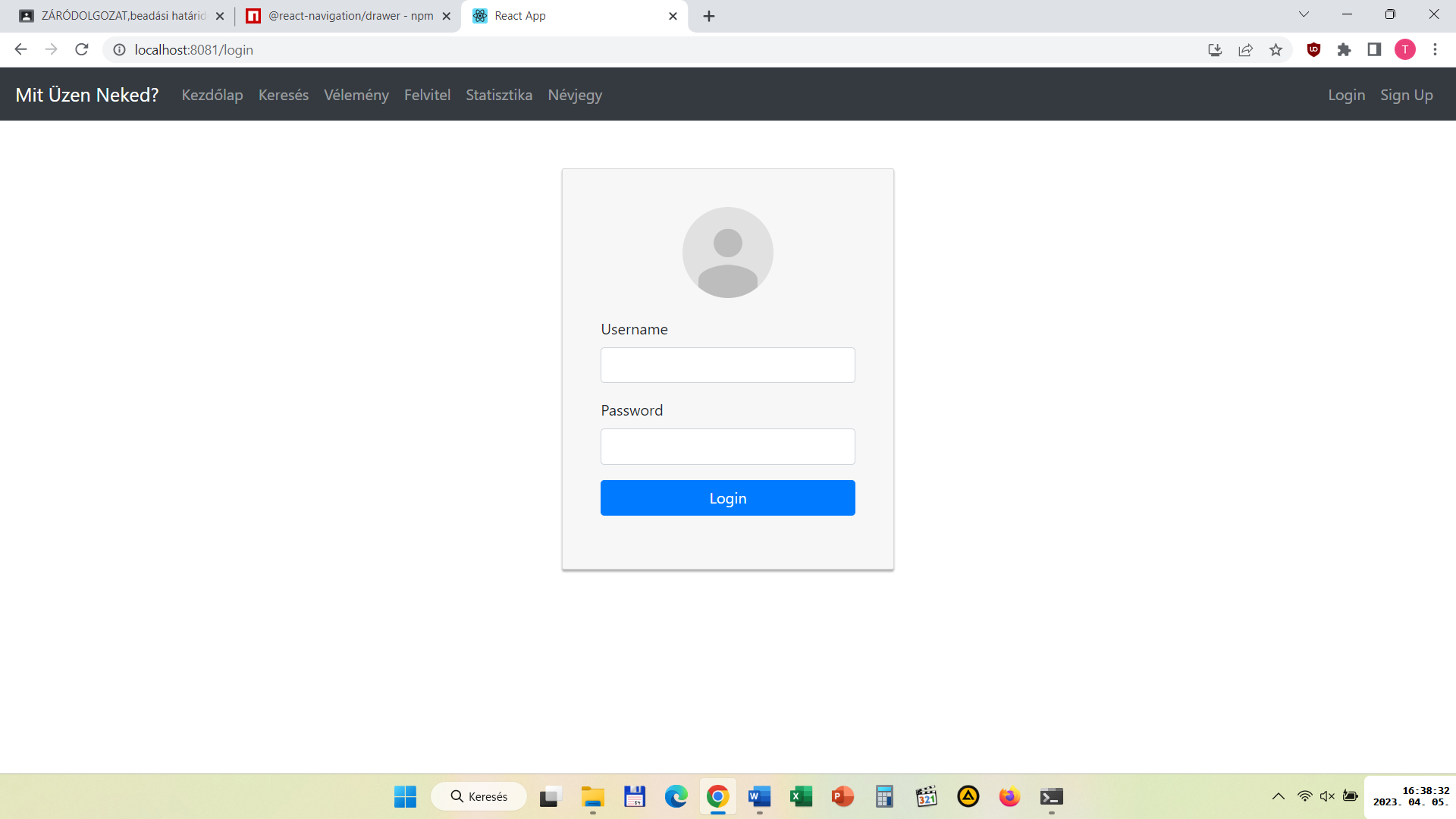
A Weben:

Nem bejelentkezett felhasználónak ezek a menük állnak rendelkezése (szürke felső sáv):

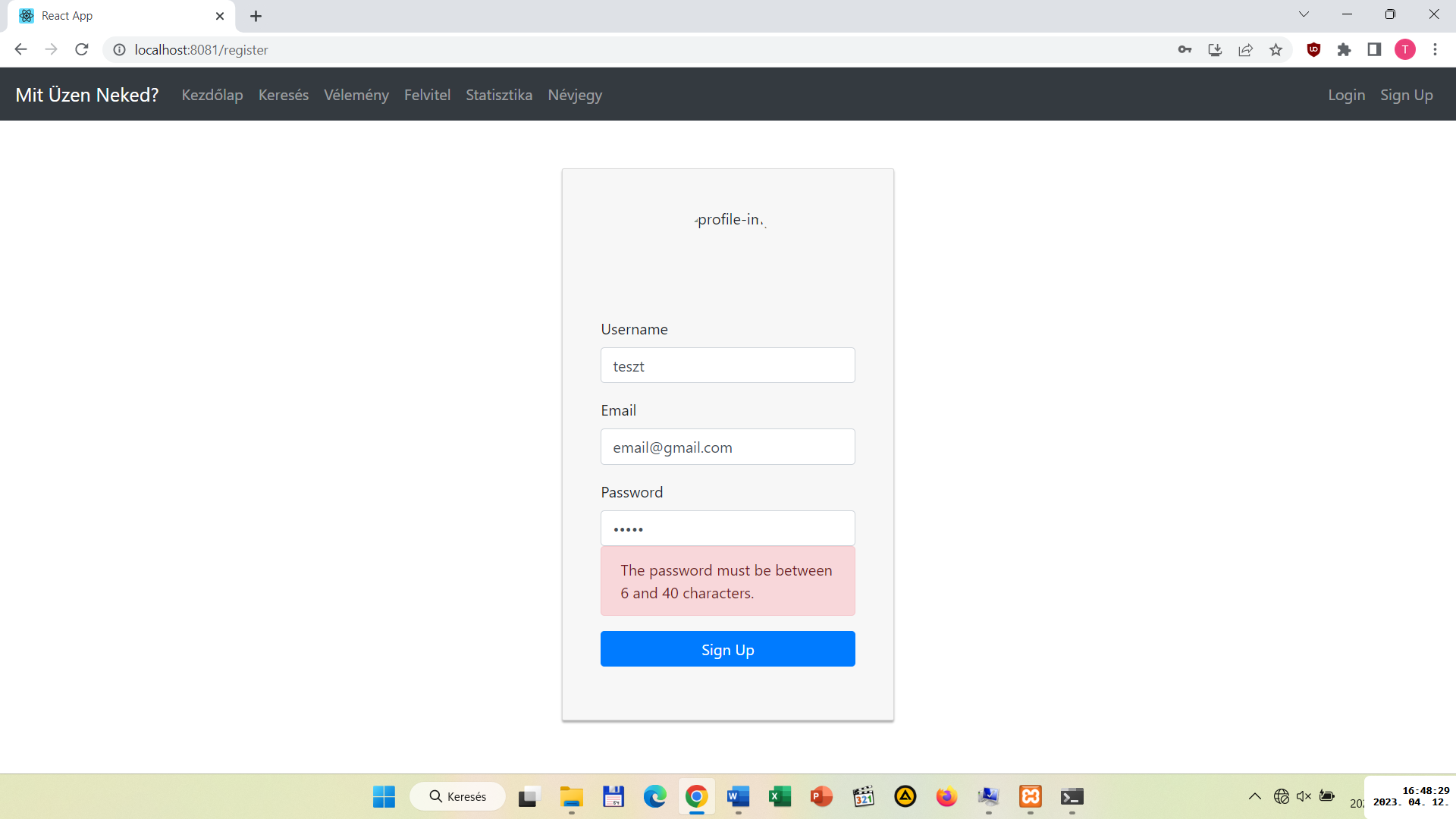
Kezdőlap, Keresés, Vélemény, Felvitel, Statisztika, Névjegy



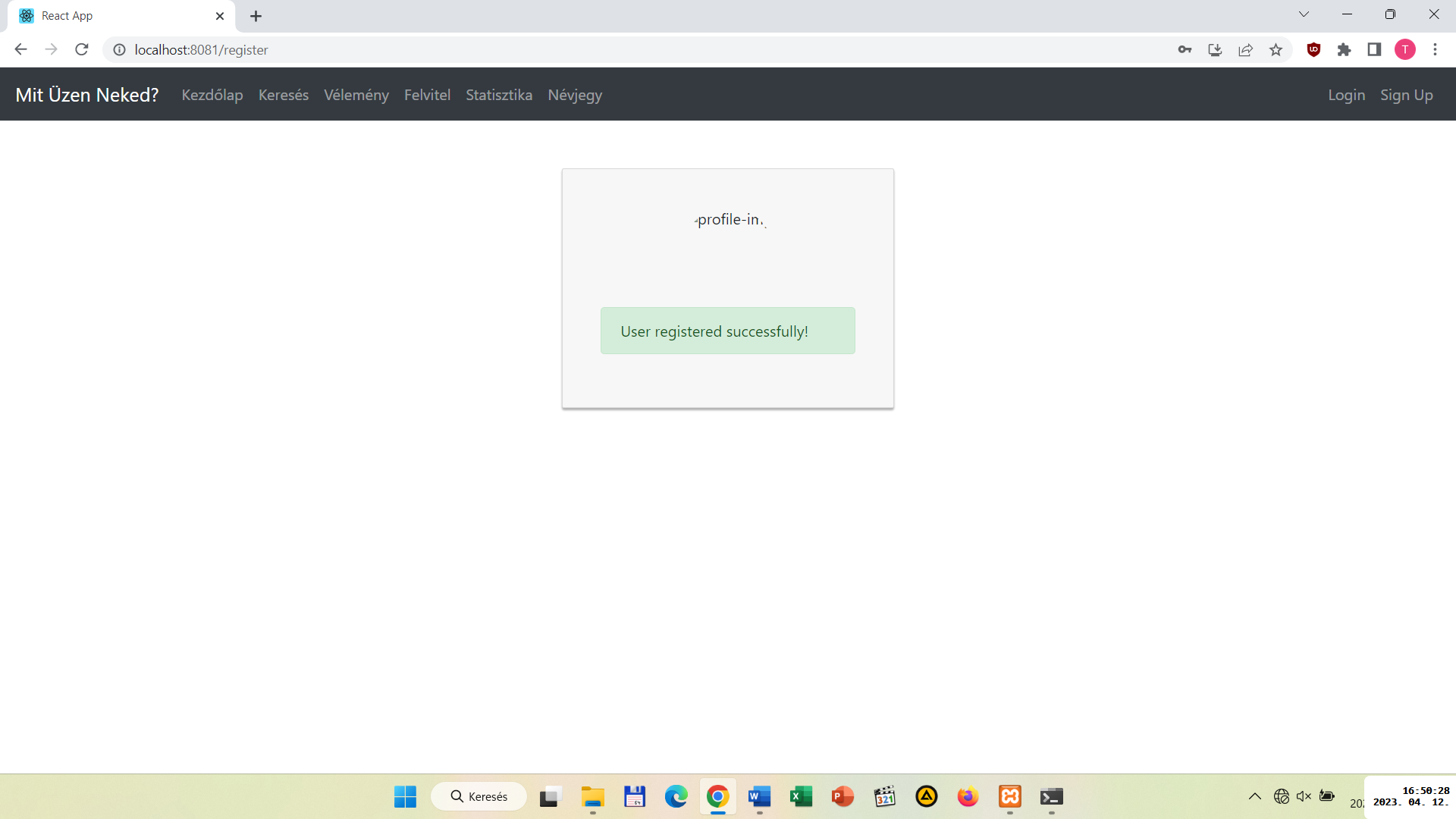
A belépési felület:



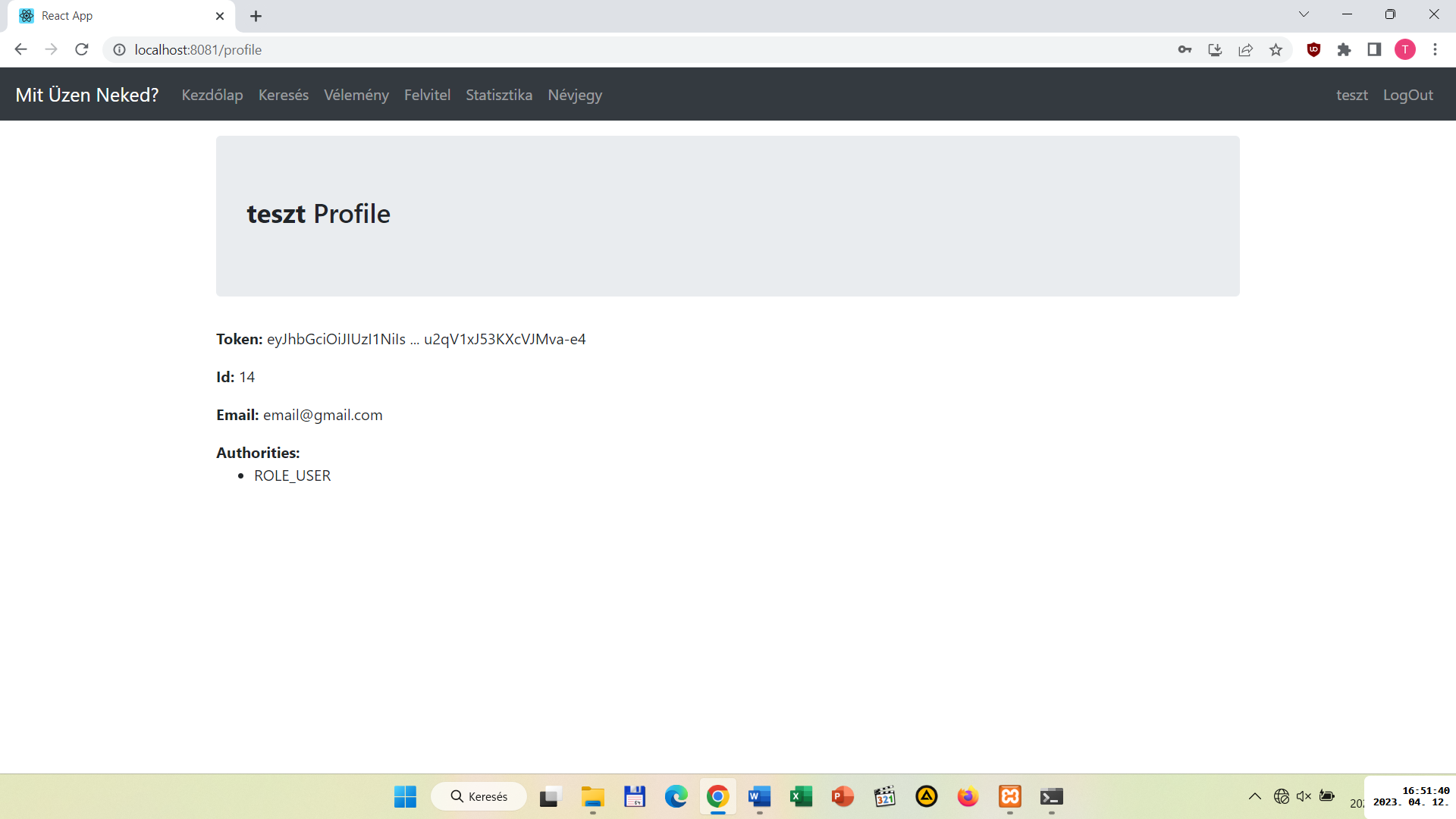
Tul rövid jelszó esetén:



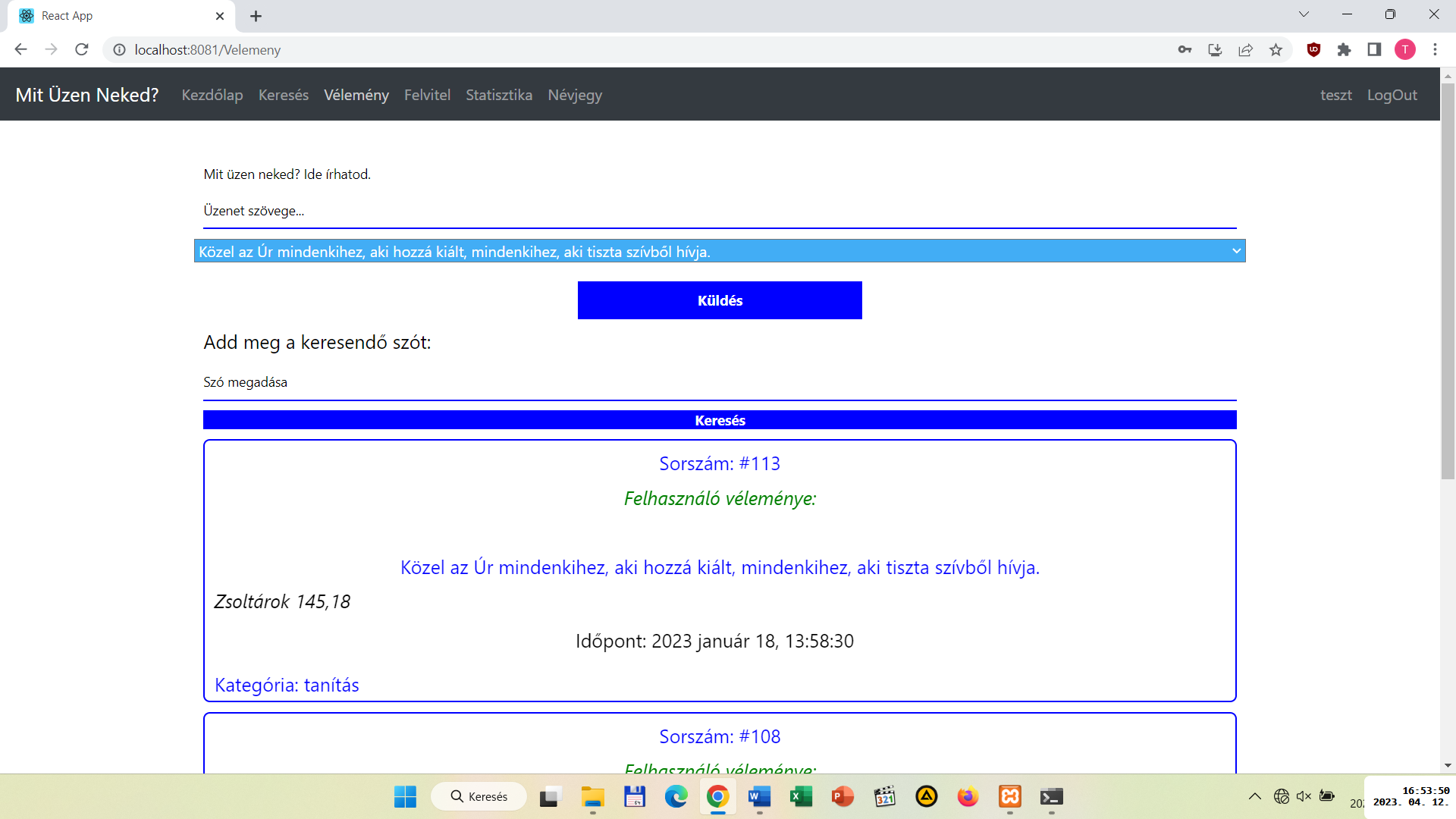
Sikeres regisztráció:



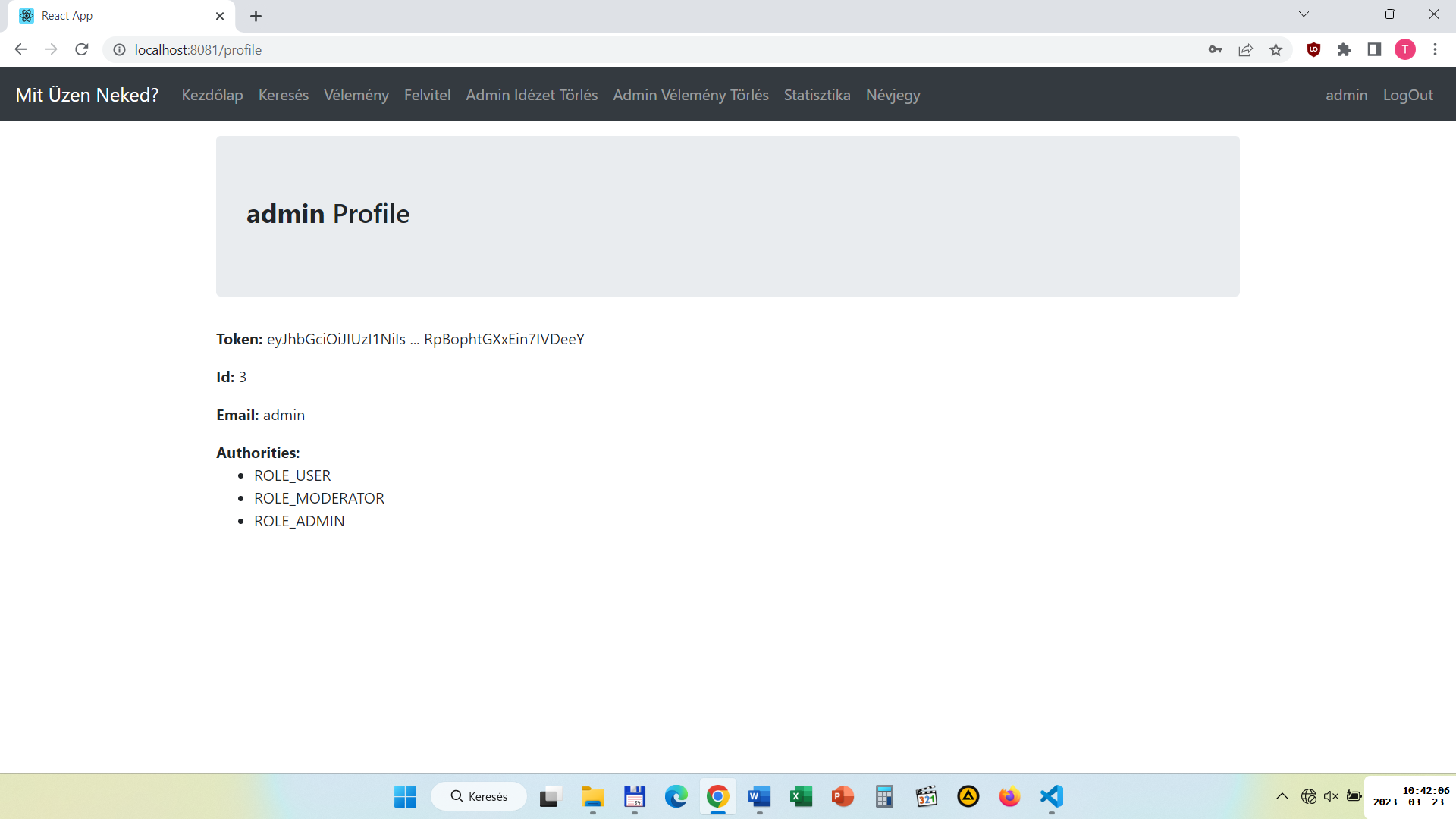
Bejelentkezett felhasználó:

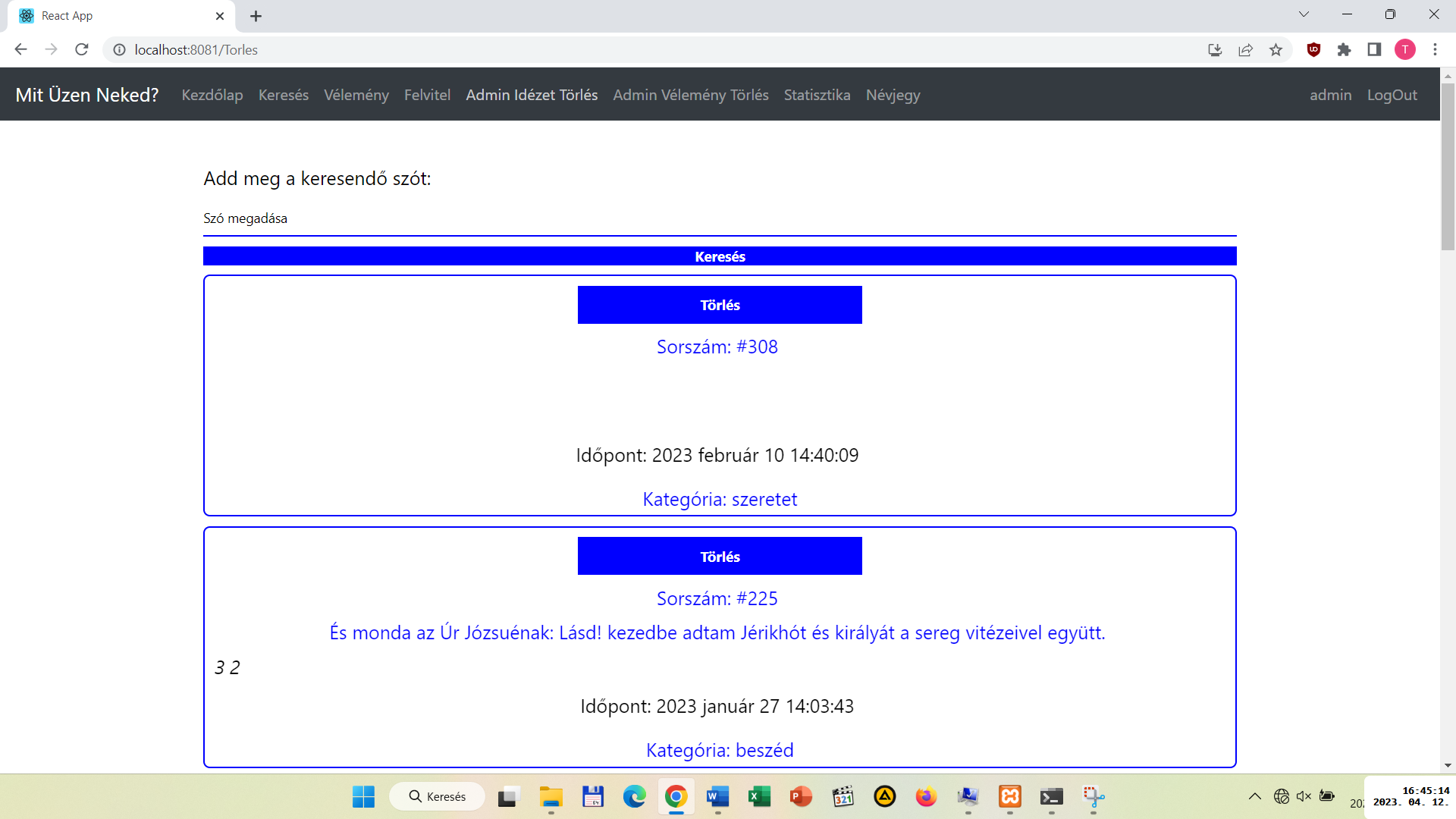


Átlagos felhasználónak nem lehetséges az idézet és vélemény törlése.



Az admin belépés után ez az oldal jön be:





Ez az admin profil. Itt már lehetőség van törölni véleményeket, és idézeteteket.

# Fejlesztői dokumentáció

## Témaválasztás indoklása

Azért választottam ezt a témát, mert sokat járok templomba, sokat olvasom a Bibliát és ezt akartam valamilyen formában összehozni az informatikai tudásommal.

## Alkalmazott fejlesztői eszközök

**Mobil applikáció:**

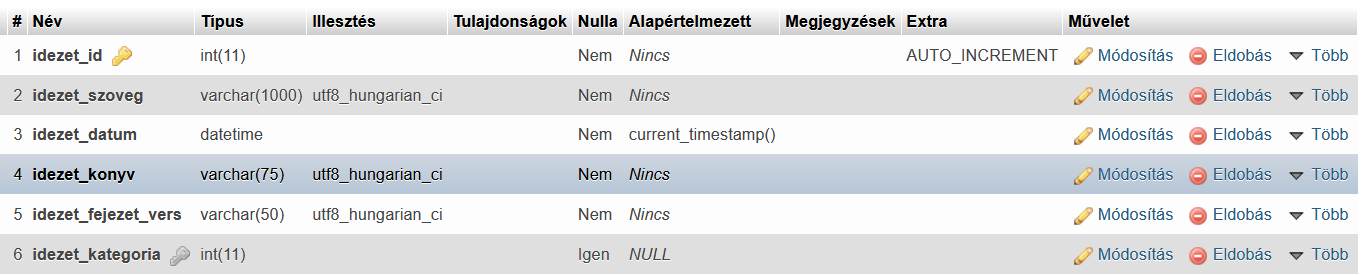
Ezt a programot react native nyelvben írtam (honlapja: https://reactnative.dev/) és a feljesztői környezet Expo Go (honlapja: https://expo.dev/), Javascript alapú. gépen kell hozzá nodejs(honlapja: https://nodejs.org/en/) fordító és terminál, és mobilon kell az Expo Go applikáció. A mobillal le kell olvasni a QR kódot, amit kihoz a gépen a program. Ennek az a feltétele, hogy azonos hálózaton kell lenni. A node-modulok az npmjs.com oldalról lehet telepíteni. Az adatbázist Mysql és PhpMyAdminban készítettem el. A kódokat VSCode nevű programban szerkesztettem. A Backend-et Express-ben írtam node.js alatt, a mobilos alkalmazás backend-je felkerült a <http://nodejs.dszcbaross.edu.hu/> iskolai szerverre, melyhez egy Mysql szerver is kapcsolódik.

## Rendszerterv

### Adatmodell leírása

Adatbázis: vallas

## idezet tábla:



idezet\_id: auto\_increment, egyedi azonosítót kap.

idezet\_szoveg: szöveg (max 50 karakter), ige szövegét tarolja

idezet\_datum: datetime, current\_timestamp(), az idézet felvitelének az időpontját tárolja

idezet\_konyv: szöveg (max 75 karakter), a könyv címét tárolja

idezet\_fejezet\_vers: szöveg (max 50 karakter), a fejezetet és a verset tárolja

idezet\_kategoria: szám típusú, idegen kulcs a kategoria tábla azonosítóját tárolja

## kategoria tábla:



kategoria\_id: auto\_increment, egyedi azonosítót kap

kategoria\_nev: szöveg (max 50 karakter), magát a kategória nevét tárolja

## velemeny tábla:



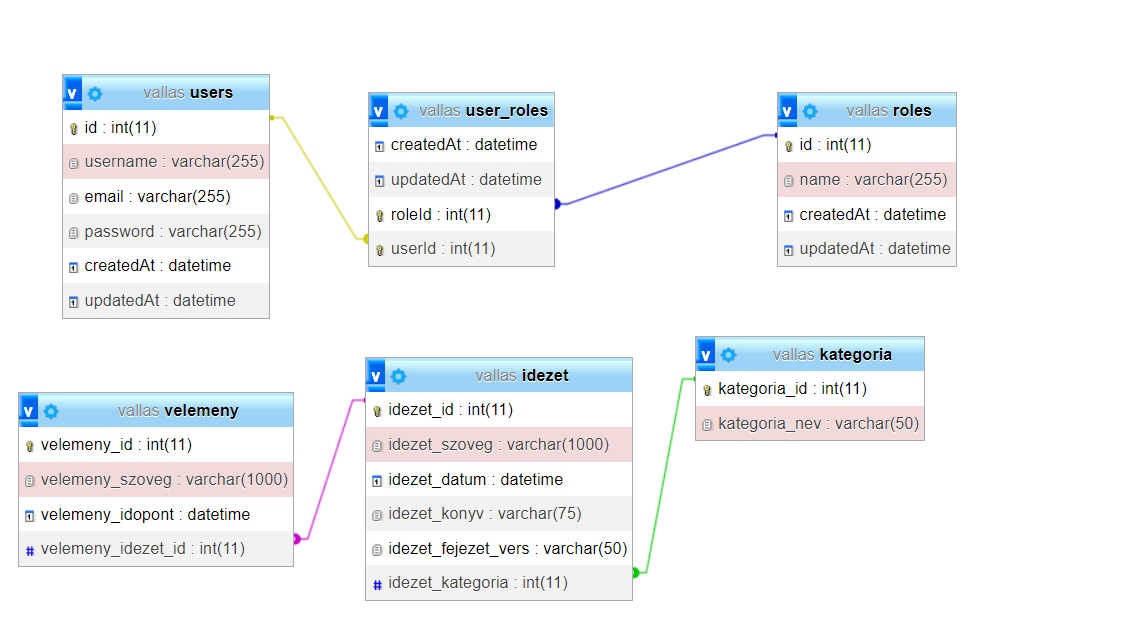
velemeny\_id: auto\_increment: egyedi azonosítót kap

velemeny\_szoveg: szöveg (max 50 karakter), a vélemény szövegét tárolja

velemeny\_idopont: datetime current\_timestamp(), a vélemény felvitelének az időpontját tárolja

velemeny\_idezet\_id: szám tipusú, idegen kulcs, az idezet táblának az azonosítóját tárolja

## kapcsolatok:

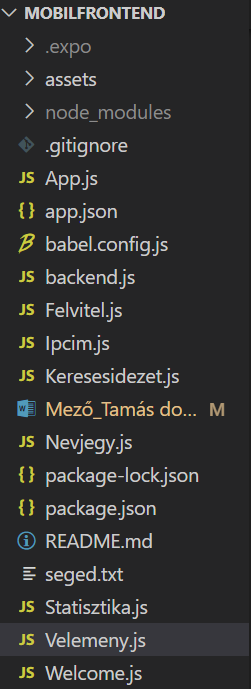


## 

## Program leírása

## Mobil applikáció:

Az mappában 9 darab .js fájl található:



Fájlok:

backend.js: A mobilfrontend backend végpontokat tartalmazza (iskolai szerver)

App.js: Ez az indítófájl, ebben vannak a menük.

Felvitel.js: Ez a fájl az idézetek felvitelét végzi el.

Ipcim.js: Az távoli backend szerver címét tárolja. Kapcsolatot létesít.

Keresesidezet.js: Az idézetek között való keresést oldja meg.

Nevjegy.js: Ez csak a névjegy.

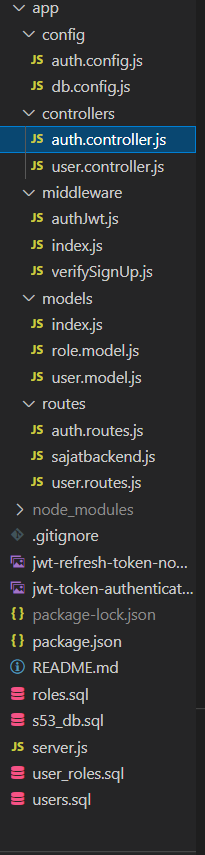
Statisztika.js: Statisztikai adatok lekérése.

Velemeny.js: Vélemények írását és keresését teszi lehetővé.

Welcome.js: Kezdőlap, itt van az üdvözlőszöveg és a gombok a menübe.

## Weblap:

### Backend:

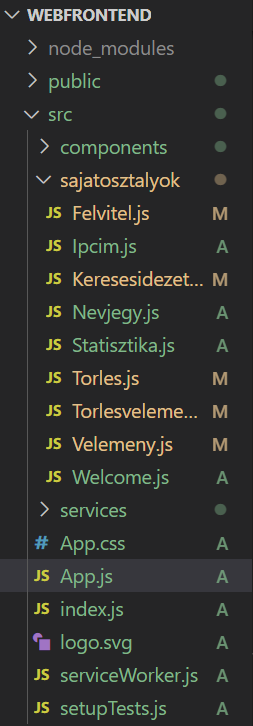


routes/sajatbackend.js: Itt vannak a backend-végpontok (lokális).

server.js: A web backend indítófájlja.

### Frontend:

A src/sajatosztalyok mappában 10 darab .js fájl található.



App.js: a web frontend indítófájlja.

src/sajatosztalyok mappa:

Felvitel.js: Az idézetek felvitelét teszi lehetővé.

Ipcim.js: A szerver címét tárolja.

Keresesidezet.js: Az idézetek között való keresést teszi lehetővé.

Nevjegy.js: Névjegyet tartalmazza.

Statisztika.js: Statisztikai adatok lekérését teszi lehetővé.

Torles.js: Az idézetek törlését és írását teszi lehetővé weben keresztül csak admin számára.

Torlesvelemeny.js: Az vélemények törlését és írását teszi lehetővé weben keresztül csak admin számára.

Velemeny.js: A vélemények írását és keresését teszi lehetővé.

Welvome.js: Kezdőlap, és üdvözlés.

## Részletes feladatspecifikáció, algoritmusok

## Mobil applikáció:

### backend.js:

const express = require('express')

var cors = require('cors')

var mysql = require('mysql')

const app = express()

const port = 22007

var connection

function dbconn() {

connection = mysql.createConnection({

host: '192.168.0.200',

user: 'u53\_jsVtWFL7iE',

password: 'Gxi8wTYeGqm+76r7+=I45OKO',

database: 's53\_db'

})

connection.connect()

}

app.use(cors())

app.use(express.json())

app.get('/', (req, res) => {

res.send('Hello World!')

})

app.post('/keresidezet', (req, res) => {

dbconn()

let parancs='SELECT \* from idezet inner join kategoria on idezet\_kategoria=kategoria\_id where idezet.idezet\_szoveg like "%'+req.body.bevitel1+'%" order by idezet\_id desc'

connection.query(parancs, function (err, rows, fields) {

if (err) throw err

console.log(rows)

res.send(rows)

})

connection.end()

})

app.post('/keresvelemeny', (req, res) => {

dbconn()

let f='"%'+req.body.bevitel1+'%"'

let parancs='SELECT \* from velemeny inner join idezet on idezet\_id=velemeny\_idezet\_id inner join kategoria on idezet\_kategoria=kategoria\_id where velemeny\_szoveg like '+f+' order by velemeny\_id desc'

connection.query(parancs, function (err, rows, fields) {

if (err) throw err

console.log(rows)

res.send(rows)

})

connection.end()

})

app.get('/idezet', (req, res) => {

dbconn()

connection.query('SELECT \* from idezet inner join kategoria on idezet\_kategoria=kategoria\_id order by idezet\_id desc', function (err, rows, fields) {

if (err) throw err

console.log(rows)

res.send(rows)

})

connection.end()

})

app.get('/kategoria', (req, res) => {

dbconn()

connection.query('SELECT \* from kategoria', function (err, rows, fields) {

if (err) throw err

console.log(rows)

res.send(rows)

})

connection.end()

})

app.post('/felvitel', (req, res) => {

dbconn()

connection.query("INSERT INTO idezet VALUES (NULL, '"+req.body.bevitel1+"',NOW(),'"+req.body.bevitel2+"' ,'"+req.body.bevitel3+"' ,'"+req.body.bevitel4+"')", function (err, rows, fields) {

if (err)

console.log( err)

else{

console.log("Sikeres felvitel!")

res.send("Sikeres felvitel!")}

})

connection.end()

})

app.get('/velemenyek', (req, res) => {

dbconn()

connection.query("select \* from velemeny inner join idezet on idezet\_id=velemeny\_idezet\_id inner join kategoria on idezet\_kategoria=kategoria\_id order by velemeny\_id desc", function (err, rows, fields) {

if (err)

console.log( err)

else{

console.log(rows)

res.send(rows)}

})

connection.end()

})

app.get('/statisztika1', (req, res) => {

dbconn()

connection.query("SELECT count(idezet\_id) as idezetek\_szama FROM idezet", function (err, rows, fields)

{

if (err)

console.log( err)

else{

console.log("Sikeres felvitel!")

res.send(rows)}

})

connection.end()

})

app.get('/statisztika2', (req, res) => {

dbconn()

connection.query("SELECT count(velemeny\_id) as velemenyek\_szama FROM velemeny", function (err, rows, fields) {

if (err)

console.log( err)

else{

console.log("Sikeres felvitel!")

res.send(rows)}

})

connection.end()

})

app.post('/velemeny', (req, res) => {

dbconn()

connection.query("INSERT INTO velemeny VALUES (NULL, '"+req.body.bevitel1+"',NOW(),'"+req.body.bevitelvelemenyid+"')", function (err, rows, fields) {

if (err)

console.log( err)

else{

console.log("Sikeres felvitel!")

res.send("Sikeres felvitel!")}

})

connection.end()

})

app.listen(port, () => {

console.log(`Example app listening at http://localhost:${port}`)

})

A backenden 9 végpont van (ebből 4 post 5 get)

get (5):

* idezet: Végrehajt egy get utasítást, amely lekérdezi az összes adatot az idezet táblából.
* kategoria: Végrehajt egy get utasítást, amely lekérdezi az összes adatot a kategoria táblából.
* velemenyek: Végrehajt egy get utasítást, amely lekérdezi az összes adatot a velemeny táblából.
* statisztika1: Végrehajt egy get utasítást, amely lekérdezi a idézetek számát az idezet táblából.
* statisztika2: Végrehajt egy get utasítást, amely lekérdezi a vélemények számát a velemeny táblából.

post (4):

* keresidezet: Keresést végez az idézet táblában.
* keresvelemeny: Keresést végez a velemeny táblában.
* felvitel: Végrehajt egy post utasítást, amely a frontendről érkező adatokat felviszi az idezet táblába.
* velemeny: Végrehajt egy post utasítást, amely a frontendről érkező adatokat felviszi az velemeny táblába.

### App.js:

import \* as React from 'react';

import { Button, View, Text } from 'react-native';

import { createDrawerNavigator } from '@react-navigation/drawer';

import { NavigationContainer } from '@react-navigation/native';

import Kezdolap from './Welcome';

import Keresidezet from './Keresesidezet';

import Felvitel from './Felvitel';

import Velemeny from './Velemeny';

import Stats from './Statisztika';

import Nevjegy from './Nevjegy';

function Kezdo\_lap({ navigation }) {

return (

<View style={{ flex: 1, alignItems: 'center', justifyContent: 'center' , marginBottom:10}}>

<Kezdolap/>

<Text style={{bottom:10, fontSize:20}}>Lehetőségek:</Text>

<View style={{marginTop:10}}>

<Button

onPress={() => navigation.navigate('Idézet keresése')}

title="Böngészés az igék között"

/>

</View>

<View style={{marginTop:10}}>

<Button

onPress={() => navigation.navigate('Felvitel')}

title="Adatok felvitele"

/>

</View>

<View style={{marginTop:10}}>

<Button

onPress={() => navigation.navigate('Statisztika')}

title="Statisztika"

/>

</View>

<View style={{marginTop:10}}>

<Button

onPress={() => navigation.navigate('Vélemények')}

title="Vélemények"

/>

</View>

<View style={{marginTop:10}}>

<Button

onPress={() => navigation.navigate('Névjegy')}

title="Névjegy"

/>

</View>

</View>

);

}

function Keresidezet\_lap({ navigation }) {

return (

<Keresidezet/>

);

}

function Felvitel\_lap({ navigation }) {

return (

<Felvitel/>

);

}

function Velemeny\_lap({ navigation }) {

return (

<Velemeny/>

);

}

function Stats\_lap({ navigation }) {

return (

<Stats/>

);

}

function Nevjegy\_lap({ navigation }) {

return (

<Nevjegy/>

);

}

const Drawer = createDrawerNavigator();

export default function App() {

return (

<NavigationContainer>

<Drawer.Navigator useLegacyImplementation initialRouteName="Home">

<Drawer.Screen name="Kezdőlap" component={Kezdo\_lap} />

<Drawer.Screen name="Idézet keresése" component={Keresidezet\_lap} />

<Drawer.Screen name="Felvitel" component={Felvitel\_lap} />

<Drawer.Screen name="Vélemények" component={Velemeny\_lap} />

<Drawer.Screen name="Statisztika" component={Stats\_lap} />

<Drawer.Screen name="Névjegy" component={Nevjegy\_lap} />

</Drawer.Navigator>

</NavigationContainer>

);

}

Az App.js fájlban találhatók a menük. A menük ebben az esetben egy-egy js fájlra hivatkoznak. Itt lehet a Kezdőlapon gombok segítségével navigálni, lehet a bal oldalról előhúzható menüvel is navigálni a menük között. A

### Felvitel.js:

import React from 'react';

import {StyleSheet, FlatList, ActivityIndicator, Text, View, TouchableOpacity, TextInput, Button, Menu} from 'react-native';

import {Picker} from '@react-native-picker/picker';

const IP=require("./Ipcim")

export default class FetchExample extends React.Component {

constructor(props){

super(props);

this.state ={

isLoading: true,

bevitel1:"",

bevitel2:"",

bevitel3:"",

bevitel4:"",

valaszto:1,

dataSource:[]

}

}

componentDidMount(){

return fetch(IP.ipcim+"kategoria")

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

.then((responseJson) => {

this.setState({

isLoading: false,

dataSource: responseJson,

}, function(){

});

})

.catch((error) =>{

console.error(error);

});

}

felvitel=()=>{

//alert("Hello")

if(this.state.bevitel1=="" || this.state.bevitel2==""||this.state.bevitel3=="")

alert("Egyetlen sor sem maradhat üresen!")

else{

var bemenet={

bevitel1:this.state.bevitel1,

bevitel2:this.state.bevitel2,

bevitel3:this.state.bevitel3,

bevitel4:this.state.valaszto

}

fetch(IP.ipcim+"felvitel", {

method: "POST",

body: JSON.stringify(bemenet),

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

}

)

.then(x => x.json())

.then(y => {

(JSON.stringify (y))

this.setState({dataSource: y})

}

);

}}

render(){

if(this.state.isLoading){

return(

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

<ActivityIndicator/>

</View>

)

}

return(

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

{/\*---------------------------------------------------kereses \*/}

<Text style={{marginLeft:10, marginTop:10, marginRight:10, marginBottom:10, fontSize:20}}>Bibliai ige:</Text>

<TextInput

style={{height: 40,marginLeft:10,marginRight:10, borderBottomColor: "blue", borderBottomWidth:2}}

placeholder="Ige szövege"

onChangeText={(beirtszoveg)=>this.setState({bevitel1:beirtszoveg})}

value={this.state.bevitel1}

/>

<TextInput

style={{height: 40,marginLeft:10,marginRight:10, borderBottomColor: "blue", borderBottomWidth:2}}

placeholder="Könyv"

onChangeText={(beirtszoveg)=>this.setState({bevitel2:beirtszoveg})}

value={this.state.bevitel2}

/>

<TextInput

style={{height: 40,marginLeft:10,marginRight:10, borderBottomColor: "blue", borderBottomWidth:2}}

placeholder="Fejezet és vers"

onChangeText={(beirtszoveg)=>this.setState({bevitel3:beirtszoveg})}

value={this.state.bevitel3}

/>

<Picker

style={{backgroundColor:"#42adf5",color:"white",marginTop:10, marginBottom:10, marginLeft:10, marginRight:10}}

selectedValue={this.state.valaszto}

onValueChange={(ertek) =>

this.setState({valaszto:ertek})

}>

{this.state.dataSource.map(item=>

<Picker.Item label={item.kategoria\_nev} value={item.kategoria\_id} />

)}

</Picker>

<TouchableOpacity style={{marginTop:10}}

onPress={()=>this.felvitel()}

>

<Text style={{textAlign:"center",color:"white",fontWeight:"bold",fontSize:15,marginLeft:10,marginRight:10, backgroundColor:"blue"}} >Felvitel</Text>

</TouchableOpacity>

{/\*---------------------------------------------------talalatok \*/}

</View>

);

}

}

const styles = StyleSheet.create({

kekgomb: {

alignItems: "center",

backgroundColor: "blue",

padding: 10,

width:300,

marginLeft:"auto",

marginRight:"auto",

marginTop:10

}

});

A Felvitel.js fájl az adatok felvitelét teszti lehetővé. A Felvitel gomb megnyomására lefut a felvitel nevű függvény, amely küldi a felhasználó által bevitt adatokat a backendnek. A moment egy olyan modul, amely lehetővé teszi, hogy a dátumot egyedi, olvasható formátumban való megjelenítését. Telepítése az npm i moment paranccsal lehetséges.

Ez szerepel a Keresesidezet.js fájban meg a weben, ahol a dátumot meg kell jeleníteni.

### Ipcim.js

module.exports = {

    ipcim:'http://nodejs.dszcbaross.edu.hu:22007/'

};

A távoli backend szerver címet tárolja, ahonnan az adatok érkeznek.

### Keresesidezet.js:

import React from 'react';

import {StyleSheet, FlatList, ActivityIndicator, Text, View, TouchableOpacity, TextInput, Button, Menu } from 'react-native';

import moment from 'moment';

import 'moment/locale/hu'

moment.locale('hu')

const IP=require("./Ipcim")

export default class FetchExample extends React.Component {

constructor(props){

super(props);

this.state ={

isLoading: true,

szo:"",

bevitel1:"",

dataSource:[],

dataSource2:[]

}

}

componentDidMount(){

fetch(IP.ipcim+"idezet")

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

.then((responseJson) => {

this.setState({

isLoading: false,

dataSource: responseJson,

}, function(){

});

})

.catch((error) =>{

console.error(error);

});

fetch(IP.ipcim+"kategoria")

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

.then((responseJson) => {

this.setState({

isLoading: false,

dataSource2: responseJson,

}, function(){

});

})

.catch((error) =>{

console.error(error);

});

}

keres=()=>{

//alert("Hello")

var bemenet={

bevitel1:this.state.szo

}

fetch(IP.ipcim+"keresidezet", {

method: "POST",

body: JSON.stringify(bemenet),

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

}

)

.then(x => x.json())

.then(y => {

(JSON.stringify (y))

this.setState({dataSource: y})

}

);

}

render(){

if(this.state.isLoading){

return(

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

<ActivityIndicator/>

</View>

)

}

return(

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

{/\*---------------------------------------------------kereses \*/}

<Text style={{marginLeft:10, marginTop:10, marginRight:10, marginBottom:10, fontSize:20}}>Add meg a keresendő szót:</Text>

<TextInput

style={{height: 40,marginLeft:10,marginRight:10, borderBottomColor: "blue", borderBottomWidth:2}}

placeholder="Szó megadása"

onChangeText={(beirtszoveg)=>this.setState({szo:beirtszoveg})}

value={this.state.szo}

/>

<TouchableOpacity style={{marginTop:10}}

onPress={()=>this.keres()}

>

<Text style={{textAlign:"center",color:"white",fontWeight:"bold",fontSize:15,marginLeft:10,marginRight:10, backgroundColor:"blue"}} >Keresés</Text>

</TouchableOpacity>

{/\*---------------------------------------------------talalatok \*/}

<FlatList

data={this.state.dataSource}

renderItem={({item}) =>

<View style={{borderWidth:2,borderColor:"blue", borderRadius:7, marginLeft:10, marginRight:10, marginTop:10, marginBottom:10}}>

<Text style={{marginRight:"auto",marginLeft:"auto",color:"blue",fontSize:20,textAlign:"center",marginLeft:10, marginRight:10, marginTop:10}} >Sorszám: #{item.idezet\_id} </Text>

<Text style={{marginRight:"auto",marginLeft:"auto",color:"blue",fontSize:20,textAlign:"center",marginLeft:10, marginRight:10, marginTop:10}} >{item.idezet\_szoveg} </Text>

<Text style={{fontStyle:"italic", fontSize:20,textAlign:"left",marginLeft:10, marginRight:10, marginTop:10}} >{item.idezet\_konyv} {item.idezet\_fejezet\_vers} </Text>

<Text style={{color:"black",fontSize:20,textAlign:"center",marginTop:15,marginBottom:5}} >Időpont: {moment(item.idezet\_datum).format('YYYY MMMM DD H:mm:ss')} </Text>

<Text style={{color:"blue",fontSize:20,textAlign:"center",marginTop:15,marginBottom:5}}>Kategória: {item.kategoria\_nev}</Text>

</View> }

//keyExtractor={({film\_id}, index) => film\_id}

/>

</View>

);

}

}

const styles = StyleSheet.create({

kekgomb: {

alignItems: "center",

backgroundColor: "blue",

padding: 10,

width:300,

marginLeft:"auto",

marginRight:"auto",

marginTop:10

}

});

A Keresés gomb megnyomására lefut a kereses nevű függvény, amely lekérdezi az idézeteket a backendről, amennyiben üresen hagyjuk a keresés mezőt akkor az összeset lekéri, ha valamilyen szót írtunk be akkor arra megfelelő találatokat adja ki. A FlatList for ciklusként funkcionál, amely végigmegy a DataSource-on. Az item.valami hivatkozik az adatra amely az adatbázis oszlopának a nevével azonos. Ez felelős az összes idézet megjelenítéséhez.

### Névjegy.js

import React, {Component} from 'react';

import {Alert, Text, StyleSheet, View} from 'react-native';

export default class ButtonBasics extends Component {

render() {

return (

<View style={styles.container}>

<Text>© Copyright: Mező Tamás 13/5, DSZC Baross Gábor Technikum</Text>

</View>

);

}

}

const styles = StyleSheet.create({

container: {

flex: 1,

justifyContent: 'center',

},

buttonContainer: {

margin: 20,

},

alternativeLayoutButtonContainer: {

margin: 20,

flexDirection: 'row',

justifyContent: 'space-between',

},

});

A Névjegy.js fájl egyszerű szöveget tartalmaz, amely a készítő nevét és az intézményét tartalmazza.

### Statisztika.js

import React from 'react';

import {StyleSheet, FlatList, ActivityIndicator, Text, View, TouchableOpacity, TextInput, Button, Menu } from 'react-native';

const IP=require("./Ipcim")

export default class FetchExample extends React.Component {

constructor(props){

super(props);

this.state ={

isLoading: true,

dataSource:[],

dataSource2:[]

}

}

componentDidMount(){

fetch(IP.ipcim+"statisztika1" )

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

.then((responseJson) => {

this.setState({

isLoading: false,

dataSource: responseJson,

}, function(){

});

})

.catch((error) =>{

console.error(error);

});

fetch(IP.ipcim+"statisztika2" )

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

.then((responseJson) => {

this.setState({

isLoading: false,

dataSource2: responseJson,

}, function(){

});

})

.catch((error) =>{

console.error(error);

});

}

frissit=()=>{

fetch(IP.ipcim+"statisztika1" )

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

.then((responseJson) => {

this.setState({

isLoading: false,

dataSource: responseJson,

}, function(){

});

})

.catch((error) =>{

console.error(error);

});

fetch(IP.ipcim+"statisztika2" )

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

.then((responseJson) => {

this.setState({

isLoading: false,

dataSource2: responseJson,

}, function(){

});

})

.catch((error) =>{

console.error(error);

});

}

render(){

if(this.state.isLoading){

return(

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

<ActivityIndicator/>

</View>

)

}

return(

<View style={{ paddingBottom:20 , paddingTop:20,borderWidth:2,borderColor:"blue", borderRadius:7, marginLeft:10, marginRight:10, marginTop:1}}>

{/\*---------------------------------------------------kereses \*/}

<TouchableOpacity

style={styles.kekgomb}

onPress={async ()=>this.frissit()}

>

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

</TouchableOpacity>

<FlatList

data={this.state.dataSource}

renderItem={({item}) =>

<View >

<Text style={{marginLeft:10, marginTop:10, marginRight:10, marginBottom:10, fontSize:20}}>Összes idézet: {item.idezetek\_szama}</Text>

</View>

}

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

/>

<FlatList

data={this.state.dataSource2}

renderItem={({item}) =>

<View >

<Text style={{marginLeft:10, marginTop:10, marginRight:10, marginBottom:10, fontSize:20}}>Összes vélemény: {item.velemenyek\_szama}</Text>

</View>

}

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

/>

{/\*---------------------------------------------------talalatok \*/}

</View>

);

}

}

const styles = StyleSheet.create({

kekgomb: {

alignItems: "center",

backgroundColor: "blue",

padding: 10,

width:300,

marginLeft:"auto",

marginRight:"auto",

marginTop:10

}

});

A Statisztika.js fájl az idézetek és a vélemények számát írja ki. A Frissít gomb megnyomására lefut a frissit nevű függvény, amely lekérdezi újból az adatokat, így már a statisztika is frissül, ha változás történt a két gombnyomás között.

### Velemeny.js

import React from 'react';

import {StyleSheet, FlatList, ActivityIndicator, Text, View, TouchableOpacity, TextInput, Button, Menu } from 'react-native';

import {Picker} from '@react-native-picker/picker';

import moment from 'moment';

import 'moment/locale/hu'

moment.locale('hu')

const IP=require("./Ipcim")

export default class FetchExample extends React.Component {

constructor(props){

super(props);

this.state ={

isLoading: true,

szo:"",

bevitel1:"",

bevitelvelemenyid:"",

valaszto2:2,

dataSource:[],

dataSource2:[]

}

}

componentDidMount(){

fetch(IP.ipcim+"velemenyek")

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

.then((responseJson) => {

this.setState({

isLoading: false,

dataSource: responseJson,

}, function(){

});

})

.catch((error) =>{

console.error(error);

});

fetch(IP.ipcim+"idezet")

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

.then((responseJson) => {

this.setState({

isLoading: false,

dataSource2: responseJson,

}, function(){

});

})

.catch((error) =>{

console.error(error);

});

}

velemeny=()=>{

if(this.state.bevitel1=="" )

alert("Nem maradhat üresen!")

else{

var bemenet={

bevitel1:this.state.bevitel1,

bevitelvelemenyid:this.state.valaszto2

}

fetch(IP.ipcim+"velemeny", {

method: "POST",

body: JSON.stringify(bemenet),

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

}

)

.then(x => x.json())

.then(y => {

(JSON.stringify (y))

this.setState({dataSource: y})

}

);

}

}

keres=()=>{

//alert("Hello")

var bemenet={

bevitel1:this.state.szo

}

fetch(IP.ipcim+"keresvelemeny", {

method: "POST",

body: JSON.stringify(bemenet),

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

}

)

.then(x => x.json())

.then(y => {

(JSON.stringify (y))

this.setState({dataSource: y})

}

);

}

render(){

if(this.state.isLoading){

return(

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

<ActivityIndicator/>

</View>

)

}

return(

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

{/\*-----------------------------------------------------------------------------Felvital \*/}

<Text style={{marginLeft:10, marginRight:10, marginTop:10}}>Mit üzen neked? Ide írhatod.</Text>

<TextInput style={{marginLeft:10, marginRight:10, marginTop:10, borderBottomColor: "blue", borderBottomWidth:2}}

placeholder="Üzenet szövege..."

onChangeText={(beirtszoveg)=>this.setState({bevitel1:beirtszoveg})}

value={this.state.bevitel1}

></TextInput>

<Picker

style={{backgroundColor:"#42adf5",color:"white",marginTop:10, marginBottom:10, marginLeft:10, marginRight:10}}

selectedValue={this.state.valaszto2}

onValueChange={(ertek) =>

this.setState({valaszto2:ertek})

}>

{this.state.dataSource2.map(item=>

<Picker.Item label={item.idezet\_szoveg} value={item.idezet\_id} />

)}

</Picker>

<TouchableOpacity

style={styles.kekgomb}

onPress={async ()=>this.velemeny()}

>

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

</TouchableOpacity>

{/\*---------------------------------------------------kereses \*/}

<Text style={{marginLeft:10, marginTop:10, marginRight:10, marginBottom:10, fontSize:20}}>Add meg a keresendő szót:</Text>

<TextInput

style={{height: 40,marginLeft:10,marginRight:10, borderBottomColor: "blue", borderBottomWidth:2}}

placeholder="Szó megadása"

onChangeText={(beirtszoveg)=>this.setState({szo:beirtszoveg})}

value={this.state.szo}

/>

<TouchableOpacity style={{marginTop:10, marginBottom:10, marginLeft:10, marginRight:10}}

onPress={()=>this.keres()}

>

<Text style={{textAlign:"center",color:"white",fontWeight:"bold",fontSize:15,marginLeft:10,marginRight:10, backgroundColor:"blue"}} >Keresés</Text>

</TouchableOpacity>

{/\*---------------------------------------------------talalatok \*/}

<FlatList

data={this.state.dataSource}

renderItem={({item}) =>

<View style={{borderWidth:2,borderColor:"blue", borderRadius:7, marginLeft:10, marginRight:10, marginTop:10, marginBottom:10}}>

<Text style={{marginRight:"auto",marginLeft:"auto",color:"blue", fontSize:20,textAlign:"center",marginLeft:10, marginRight:10, marginTop:10}} >Sorszám: #{item.velemeny\_id}</Text>

<Text style={{marginRight:"auto",marginLeft:"auto",color:"green",fontStyle:"italic", fontSize:20,textAlign:"center",marginLeft:10, marginRight:10, marginTop:10}} >Felhasználó véleménye:</Text>

<Text style={{marginRight:"auto",marginLeft:"auto",color:"green",fontStyle:"italic", fontSize:20,textAlign:"center",marginLeft:10, marginRight:10, marginTop:10}} >{item.velemeny\_szoveg} </Text>

<Text style={{marginRight:"auto",marginLeft:"auto",color:"blue",fontSize:20,textAlign:"center",marginLeft:10, marginRight:10, marginTop:10}} >{item.idezet\_szoveg} </Text>

<Text style={{fontStyle:"italic", fontSize:20,textAlign:"left",marginLeft:10, marginRight:10, marginTop:10}} >{item.idezet\_konyv} {item.idezet\_fejezet\_vers} </Text>

<Text style={{color:"black",fontSize:20,textAlign:"center",marginTop:15,marginBottom:5}} >Időpont: {moment(item.idezet\_datum).format('YYYY MMMM DD, H:mm:ss')} </Text>

<Text style={{marginLeft:10, marginRight:10, marginTop:10,color:"blue",fontSize:20,textAlign:"left",marginTop:15,marginBottom:5}} >Kategória: {item.kategoria\_nev} </Text>

</View>

}

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

/>

</View>

);

}

}

const styles = StyleSheet.create({

kekgomb: {

alignItems: "center",

backgroundColor: "blue",

padding: 10,

marginLeft:10,

marginRight:10,

marginBottom:10,

marginTop:10

}

});

A Vélemény.js fájl lehetővé teszi, hogy leírjuk, hogy mit üzen számunkra az adott ige. A felső Küldés gomb megnyomására lefut a velemeny függvény, amely elküldi a backendnek az adatokat. Ha üresen hagyjuk, akkor a „Nem maradhat üresen!” üzenet ugrik fel. A lentebb lévő Keresés gomb megnyomására lefut a kereses nevű függvény, amely a keresést teszi lehetővé.

### Welcome.js

import React, {Component} from 'react';

import {Alert, Text, StyleSheet, View} from 'react-native';

export default class ButtonBasics extends Component {

render() {

return (

<View style={styles.container}>

<Text style={{textAlign:"center"}}>Üdvözöllek az alkalmazásban aminek a témája Bibliai igékkel kapcsolatos. Ebben az alkalmazásban elmondhatja, hogy mit üzen az adott Ige.</Text>

</View>

);

}

}

const styles = StyleSheet.create({

container: {

flex: 1,

justifyContent: 'center',

},

buttonContainer: {

margin: 20,

},

alternativeLayoutButtonContainer: {

margin: 20,

flexDirection: 'row',

justifyContent: 'space-between',

},

});

Kezdőlap azaz a kiindulópont.

## Weblap

### backend.js:

const { authJwt } = require("../middleware");

const controller = require("../controllers/user.controller");

module.exports = function(app) {

app.use(function(req, res, next) {

res.header(

"Access-Control-Allow-Headers",

"x-access-token, Origin, Content-Type, Accept"

);

next();

});

var connection

function db\_connect(){

var mysql = require('mysql')

connection = mysql.createConnection({

host: 'localhost',

user: 'root',

password: '',

database: 'Vallas'

})

connection.connect()

}

app.get('/idezet', (req, res) => {

db\_connect()

connection.query('SELECT \* from idezet inner join kategoria on idezet\_kategoria=kategoria\_id order by idezet\_id desc', function (err, rows, fields) {

if (err) throw err

console.log(rows)

res.send(rows)

})

connection.end()

})

app.delete('/torles\_idezet', (req, res) => {

db\_connect()

let parancs="DELETE FROM idezet WHERE idezet.idezet\_id="+req.body.bevitel1+";";

connection.query(parancs, function (err, rows, fields) {

if (err) throw err

console.log(rows)

res.send(rows)

})

connection.end()

})

app.delete('/torles\_velemeny', (req, res) => {

db\_connect()

let parancs="DELETE FROM velemeny WHERE velemeny.velemeny\_id="+req.body.bevitel1+";";

connection.query(parancs, function (err, rows, fields) {

if (err) throw err

console.log(rows)

res.send(rows)

})

connection.end()

})

app.post('/keresidezet', (req, res) => {

db\_connect()

let parancs='SELECT \* from idezet inner join kategoria on idezet\_kategoria=kategoria\_id where idezet.idezet\_szoveg like "%'+req.body.bevitel1+'%" order by idezet\_id desc'

connection.query(parancs, function (err, rows, fields) {

if (err) throw err

console.log(rows)

res.send(rows)

})

connection.end()

})

app.post('/keresvelemeny', (req, res) => {

db\_connect()

let f='"%'+req.body.bevitel1+'%"'

let parancs='SELECT \* from velemeny inner join idezet on idezet\_id=velemeny\_idezet\_id inner join kategoria on idezet\_kategoria=kategoria\_id where velemeny\_szoveg like '+f+' order by velemeny\_id desc'

connection.query(parancs, function (err, rows, fields) {

if (err) throw err

console.log(rows)

res.send(rows)

})

connection.end()

})

app.get('/statisztika3', (req, res) => {

db\_connect()

connection.query('SELECT count(id) as userscount FROM users', function (err, rows, fields) {

if (err) throw err

console.log(rows)

res.send(rows)

})

connection.end()

})

app.get('/kategoria', (req, res) => {

db\_connect()

connection.query('SELECT \* from kategoria', function (err, rows, fields) {

if (err) throw err

console.log(rows)

res.send(rows)

})

connection.end()

})

app.post('/felvitel', (req, res) => {

db\_connect()

connection.query("INSERT INTO idezet VALUES (NULL, '"+req.body.bevitel1+"',NOW(),'"+req.body.bevitel2+"' ,'"+req.body.bevitel3+"' ,'"+req.body.bevitel4+"')", function (err, rows, fields) {

if (err)

console.log( err)

else{

console.log("Sikeres felvitel!")

res.send("Sikeres felvitel!")}

})

connection.end()

})

app.get('/velemenyek', (req, res) => {

db\_connect()

connection.query("select \* from velemeny inner join idezet on idezet\_id=velemeny\_idezet\_id inner join kategoria on idezet\_kategoria=kategoria\_id order by velemeny\_id desc", function (err, rows, fields) {

if (err)

console.log( err)

else{

console.log("Sikeres felvitel!")

res.send(rows)}

})

connection.end()

})

app.get('/statisztika1', (req, res) => {

db\_connect()

connection.query("SELECT count(idezet\_id) as idezetek\_szama FROM idezet", function (err, rows, fields)

{

if (err)

console.log( err)

else{

console.log("Sikeres felvitel!")

res.send(rows)}

})

connection.end()

})

app.get('/statisztika2', (req, res) => {

db\_connect()

connection.query("SELECT count(velemeny\_id) as velemenyek\_szama FROM velemeny", function (err, rows, fields) {

if (err)

console.log( err)

else{

console.log("Sikeres felvitel!")

res.send(rows)}

})

connection.end()

})

app.post('/velemeny', (req, res) => {

db\_connect()

connection.query("INSERT INTO velemeny VALUES (NULL, '"+req.body.bevitel1+"',NOW(),'"+req.body.bevitelvelemenyid+"')", function (err, rows, fields) {

if (err)

console.log( err)

else{

console.log("Sikeres felvitel!")

res.send("Sikeres felvitel!")}

})

connection.end()

})

};

A backend.js fájlban 13 darab végpont van (ebből 2 delete, 4 post, 6 get)

get (6):

* idezet: Végrehajt egy get utasítást, amely lekérdezi az összes adatot az idezet táblából.
* kategoria: Végrehajt egy get utasítást, amely lekérdezi az összes adatot a kategoria táblából.
* velemenyek: Végrehajt egy get utasítást, amely lekérdezi az összes adatot a velemeny táblából.
* statisztika1: Végrehajt egy get utasítást, amely lekérdezi az idézetek számát az idezet táblából.
* statisztika2: Végrehajt egy get utasítást, amely lekérdezi a vélemények számát a velemeny táblából.
* statisztika3: Végrehajt egy get utasítást, amely lekérdezi a felhasználók számát az users táblából.

post (4):

* keresidezet: Keresést végez az idézet táblában.
* keresvelemeny: Keresést végez a velemeny táblában.
* felvitel: Végrehajt egy post utasítást, amely a frontendről érkező adatokat felviszi az idezet táblába.
* velemeny: Végrehajt egy post utasítást, amely a frontendről érkező adatokat felviszi az velemeny táblába.

delete (2):

* torles\_idezet: Végrehajt egy delete utasítást, amely törli az adott idézetet az idezet táblából.
* torles\_velemeny: Végrehajt egy delete utasítást, amely törli az adott véleményt a velemeny táblából.

### Felvitel.js

import React from 'react';

import {StyleSheet, FlatList, ActivityIndicator, Text, View, TouchableOpacity, TextInput, Button, Menu} from 'react-native';

import {Picker} from '@react-native-picker/picker';

const IP=require("./Ipcim")

export default class FetchExample extends React.Component {

constructor(props){

super(props);

this.state ={

isLoading: true,

bevitel1:"",

bevitel2:"",

bevitel3:"",

bevitel4:"",

valaszto:1,

dataSource:[]

}

}

componentDidMount(){

return fetch(IP.ipcim+"kategoria")

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

.then((responseJson) => {

this.setState({

isLoading: false,

dataSource: responseJson,

}, function(){

});

})

.catch((error) =>{

console.error(error);

});

}

felvitel=()=>{

//alert("Hello")

if(this.state.bevitel1=="" || this.state.bevitel2==""||this.state.bevitel3=="")

alert("Egyetlen sor sem maradhat üresen!")

else{

var bemenet={

bevitel1:this.state.bevitel1,

bevitel2:this.state.bevitel2,

bevitel3:this.state.bevitel3,

bevitel4:this.state.valaszto

}

fetch(IP.ipcim+"felvitel", {

method: "POST",

body: JSON.stringify(bemenet),

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

}

)

.then(x => x.json())

.then(y => {

(JSON.stringify (y))

this.setState({dataSource: y})

}

);

}

}

render(){

if(this.state.isLoading){

return(

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

<ActivityIndicator/>

</View>

)

}

return(

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

{/\*---------------------------------------------------kereses \*/}

<Text style={{marginLeft:10, marginTop:10, marginRight:10, marginBottom:10, fontSize:20}}>Bibliai ige:</Text>

<TextInput

style={{height: 40,marginLeft:10,marginRight:10}}

placeholder="Ige szövege"

onChangeText={(beirtszoveg)=>this.setState({bevitel1:beirtszoveg})}

value={this.state.bevitel1}

/>

<TextInput

style={{height: 40,marginLeft:10,marginRight:10}}

placeholder="Könyv"

onChangeText={(beirtszoveg)=>this.setState({bevitel2:beirtszoveg})}

value={this.state.bevitel2}

/>

<TextInput

style={{height: 40,marginLeft:10,marginRight:10}}

placeholder="Fejezet és vers"

onChangeText={(beirtszoveg)=>this.setState({bevitel3:beirtszoveg})}

value={this.state.bevitel3}

/>

<Picker

style={{backgroundColor:"#42adf5",color:"white",marginTop:10, marginBottom:10}}

selectedValue={this.state.valaszto}

onValueChange={(ertek) =>

this.setState({valaszto:ertek})

}>

{this.state.dataSource.map(item=>

<Picker.Item label={item.kategoria\_nev} value={item.kategoria\_id} />

)}

</Picker>

<TouchableOpacity style={{marginTop:10}}

onPress={()=>this.felvitel()}

>

<Text style={{textAlign:"center",color:"white",fontWeight:"bold",fontSize:15,marginLeft:10,marginRight:10, backgroundColor:"blue"}} >Felvitel</Text>

</TouchableOpacity>

{/\*---------------------------------------------------talalatok \*/}

</View>

);

}

}

const styles = StyleSheet.create({

kekgomb: {

alignItems: "center",

backgroundColor: "blue",

padding: 10,

width:300,

marginLeft:"auto",

marginRight:"auto",

marginTop:10

}

});

A Felvitel.js fájlban az adatok(idézetek) felvitelét oldottam meg. Egyik sor sem maradhat üresen, mindegyiket ki kell tölteni, hogy megtörténjen a felvitel. Ellenkező esetben figyelmeztetés ugrik fel, hogy nem maradhatnak üresen.

### Ipcim.js

module.exports = {

ipcim: 'http://localhost:8080/'

};

Ugyan az, ami a mobil applikációban, csak nem távoli szerverről veszi az adatokat, hanem a localhostról(helyi) ez nem publikus mint a mobilos.

### Keresesidezet.js

import React from 'react';

import {StyleSheet, FlatList, ActivityIndicator, Text, View, TouchableOpacity, TextInput, Button, Menu } from 'react-native';

import moment from 'moment';

import 'moment/locale/hu'

moment.locale('hu')

const IP=require("./Ipcim")

export default class FetchExample extends React.Component {

constructor(props){

super(props);

this.state ={

isLoading: true,

szo:"",

bevitel1:"",

dataSource:[],

dataSource2:[]

}

}

componentDidMount(){

fetch(IP.ipcim+"idezet")

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

.then((responseJson) => {

this.setState({

isLoading: false,

dataSource: responseJson,

}, function(){

});

})

.catch((error) =>{

console.error(error);

});

fetch(IP.ipcim+"kategoria")

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

.then((responseJson) => {

this.setState({

isLoading: false,

dataSource2: responseJson,

}, function(){

});

})

.catch((error) =>{

console.error(error);

});

}

keres=()=>{

//alert("Hello")

var bemenet={

bevitel1:this.state.szo

}

fetch(IP.ipcim+"keresidezet", {

method: "POST",

body: JSON.stringify(bemenet),

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

}

)

.then(x => x.json())

.then(y => {

(JSON.stringify (y))

this.setState({dataSource: y})

}

);

}

render(){

if(this.state.isLoading){

return(

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

<ActivityIndicator/>

</View>

)

}

return(

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

{/\*---------------------------------------------------kereses \*/}

<Text style={{marginLeft:10, marginTop:10, marginRight:10, marginBottom:10, fontSize:20}}>Add meg a keresendő szót:</Text>

<TextInput

style={{height: 40,marginLeft:10,marginRight:10,borderBottomWidth:2,borderBottomColor:'blue'}}

placeholder="Szó megadása"

onChangeText={(beirtszoveg)=>this.setState({szo:beirtszoveg})}

value={this.state.szo}

/>

<TouchableOpacity style={{marginTop:10}}

onPress={()=>this.keres()}

>

<Text style={{textAlign:"center",color:"white",fontWeight:"bold",fontSize:15,marginLeft:10,marginRight:10, backgroundColor:"blue"}} >Keresés</Text>

</TouchableOpacity>

{/\*---------------------------------------------------talalatok \*/}

<FlatList

data={this.state.dataSource}

renderItem={({item}) =>

<View style={{borderWidth:2,borderColor:"blue", borderRadius:7, marginLeft:10, marginRight:10, marginTop:10}}>

<Text style={{marginRight:"auto",marginLeft:"auto",color:"blue",fontSize:20,textAlign:"center",marginLeft:10, marginRight:10, marginTop:10}} >Sorszám: #{item.idezet\_id} </Text>

<Text style={{marginRight:"auto",marginLeft:"auto",color:"blue",fontSize:20,textAlign:"center",marginLeft:10, marginRight:10, marginTop:10}} >{item.idezet\_szoveg} </Text>

<Text style={{fontStyle:"italic", fontSize:20,textAlign:"left",marginLeft:10, marginRight:10, marginTop:10}} >{item.idezet\_konyv} {item.idezet\_fejezet\_vers} </Text>

<Text style={{color:"black",fontSize:20,textAlign:"center",marginTop:15,marginBottom:5}} >Időpont: {moment(item.idezet\_datum).format('YYYY MMMM DD H:mm:ss')} </Text>

<Text style={{color:"blue",fontSize:20,textAlign:"center",marginTop:15,marginBottom:5}}>Kategória: {item.kategoria\_nev}</Text>

</View>

}

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

/>

</View>

);

}

}

const styles = StyleSheet.create({

kekgomb: {

alignItems: "center",

backgroundColor: "blue",

padding: 10,

width:300,

marginLeft:"auto",

marginRight:"auto",

marginTop:10

}

});

A Keresesidezet.js lehetővé teszi az idézetek közötti keresést. Itt, ha üresen hagyjuk a beviteli mezőt, akkor az összes megjelenítésre kerül.

### Nevjegy.js

import React, {Component} from 'react';

import {Alert, Text, StyleSheet, View} from 'react-native';

export default class ButtonBasics extends Component {

render() {

return (

<View style={styles.container}>

<Text>© Copyright: Mező Tamás 13/5, DSZC Baross Gábor Technikum</Text>

</View>

);

}

}

const styles = StyleSheet.create({

container: {

flex: 1,

justifyContent: 'center',

},

buttonContainer: {

margin: 20,

},

alternativeLayoutButtonContainer: {

margin: 20,

flexDirection: 'row',

justifyContent: 'space-between',

},

});

A Nevjegy.js a záródolgozat készítőjének a nevét tartalmazza.

### Statisztika.js

import React from 'react';

import {StyleSheet, FlatList, ActivityIndicator, Text, View, TouchableOpacity, TextInput, Button, Menu } from 'react-native';

const IP=require("./Ipcim")

export default class FetchExample extends React.Component {

constructor(props){

super(props);

this.state ={

isLoading: true,

dataSource:[],

dataSource2:[],

dataSource3:[]

}

}

componentDidMount(){

fetch(IP.ipcim+"statisztika1" )

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

.then((responseJson) => {

this.setState({

isLoading: false,

dataSource: responseJson,

}, function(){

});

})

.catch((error) =>{

console.error(error);

});

fetch(IP.ipcim+"statisztika2" )

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

.then((responseJson) => {

this.setState({

isLoading: false,

dataSource2: responseJson,

}, function(){

});

})

.catch((error) =>{

console.error(error);

});

fetch(IP.ipcim+"statisztika3" )

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

.then((responseJson) => {

this.setState({

isLoading: false,

dataSource3: responseJson,

}, function(){

});

})

.catch((error) =>{

console.error(error);

});

}

frissit=()=>{

fetch(IP.ipcim+"statisztika1" )

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

.then((responseJson) => {

this.setState({

isLoading: false,

dataSource: responseJson,

}, function(){

});

})

.catch((error) =>{

console.error(error);

});

fetch(IP.ipcim+"statisztika2" )

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

.then((responseJson) => {

this.setState({

isLoading: false,

dataSource2: responseJson,

}, function(){

});

})

.catch((error) =>{

console.error(error);

});

}

render(){

if(this.state.isLoading){

return(

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

<ActivityIndicator/>

</View>

)

}

return(

<View style={{ paddingBottom:20 , paddingTop:20,borderWidth:2,borderColor:"blue", borderRadius:7, marginLeft:10, marginRight:10, marginTop:1}}>

{/\*---------------------------------------------------kereses \*/}

<TouchableOpacity

style={styles.kekgomb}

onPress={async ()=>this.frissit()}

>

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

</TouchableOpacity>

<FlatList

data={this.state.dataSource}

renderItem={({item}) =>

<View >

<Text style={{marginLeft:10, marginTop:10, marginRight:10, marginBottom:10, fontSize:20}}>Összes idézet: {item.idezetek\_szama}</Text>

</View>

}

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

/>

<FlatList

data={this.state.dataSource2}

renderItem={({item}) =>

<View >

<Text style={{marginLeft:10, marginTop:10, marginRight:10, marginBottom:10, fontSize:20}}>Összes vélemény: {item.velemenyek\_szama}</Text>

</View>

}

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

/>

<FlatList

data={this.state.dataSource3}

renderItem={({item}) =>

<View >

<Text style={{marginLeft:10, marginTop:10, marginRight:10, marginBottom:10, fontSize:20}}>Felhasználók száma: {item.userscount}</Text>

</View>

}

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

/>

{/\*---------------------------------------------------talalatok \*/}

</View>

);

}

}

const styles = StyleSheet.create({

kekgomb: {

alignItems: "center",

backgroundColor: "blue",

padding: 10,

width:300,

marginLeft:"auto",

marginRight:"auto",

marginTop:10

}

});

A Statisztika.js lehetővé teszi, hogy kiírja az idézetek, vélemények, felhasználók számát. A Frissítés gombra frissülnek az adatok, amennyiben történt változás.

### Törles.js

import React from 'react';

import {StyleSheet, FlatList, ActivityIndicator, Text, View, TouchableOpacity, TextInput, Button, Menu, Alert } from 'react-native';

import moment from 'moment';

import 'moment/locale/hu'

moment.locale('hu')

const IP=require("./Ipcim")

export default class FetchExample extends React.Component {

constructor(props){

super(props);

this.state ={

isLoading: true,

szo:"",

bevitel1:"",

szam:"",

dataSource:[],

dataSource2:[]

}

}

componentDidMount(){

fetch(IP.ipcim+"idezet")

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

.then((responseJson) => {

this.setState({

isLoading: false,

dataSource: responseJson,

}, function(){

});

})

.catch((error) =>{

console.error(error);

});

fetch(IP.ipcim+"kategoria")

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

.then((responseJson) => {

this.setState({

isLoading: false,

dataSource2: responseJson,

}, function(){

});

})

.catch((error) =>{

console.error(error);

});

}

keres=()=>{

//alert("Hello")

var bemenet={

bevitel1:this.state.szo

}

fetch(IP.ipcim+"keresidezet", {

method: "POST",

body: JSON.stringify(bemenet),

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

}

)

.then(x => x.json())

.then(y => {

(JSON.stringify (y))

this.setState({dataSource: y})

}

);

}

torles=(szam)=>{

alert(szam)

var bemenet={

bevitel1:szam

}

fetch(IP.ipcim+"torles\_idezet", {

method: "DELETE",

body: JSON.stringify(bemenet),

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

}

)

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

.then(y => {

alert(y)

});

}

render(){

if(this.state.isLoading){

return(

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

<ActivityIndicator/>

</View>

)

}

return(

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

{/\*---------------------------------------------------kereses \*/}

<Text style={{marginLeft:10, marginTop:10, marginRight:10, marginBottom:10, fontSize:20}}>Add meg a keresendő szót:</Text>

<TextInput

style={{height: 40,marginLeft:10,marginRight:10, borderBottomColor:'blue',borderBottomWidth:2}}

placeholder="Szó megadása"

onChangeText={(beirtszoveg)=>this.setState({szo:beirtszoveg})}

value={this.state.szo}

/>

<TouchableOpacity style={{marginTop:10}}

onPress={()=>this.keres()}

>

<Text style={{textAlign:"center",color:"white",fontWeight:"bold",fontSize:15,marginLeft:10,marginRight:10, backgroundColor:"blue"}} >Keresés</Text>

</TouchableOpacity>

{/\*---------------------------------------------------talalatok \*/}

<FlatList

data={this.state.dataSource}

renderItem={({item}) =>

<View style={{borderWidth:2,borderColor:"blue", borderRadius:7, marginLeft:10, marginRight:10, marginTop:10}}>

<TouchableOpacity

style={styles.kekgomb}

onPress={async ()=>this.torles(item.idezet\_id)}

>

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

</TouchableOpacity>

<Text style={{marginRight:"auto",marginLeft:"auto",color:"blue",fontSize:20,textAlign:"center",marginLeft:10, marginRight:10, marginTop:10}} >Sorszám: #{item.idezet\_id} </Text>

<Text style={{marginRight:"auto",marginLeft:"auto",color:"blue",fontSize:20,textAlign:"center",marginLeft:10, marginRight:10, marginTop:10}} >{item.idezet\_szoveg} </Text>

<Text style={{fontStyle:"italic", fontSize:20,textAlign:"left",marginLeft:10, marginRight:10, marginTop:10}} >{item.idezet\_konyv} {item.idezet\_fejezet\_vers} </Text>

<Text style={{color:"black",fontSize:20,textAlign:"center",marginTop:15,marginBottom:5}} >Időpont: {moment(item.idezet\_datum).format('YYYY MMMM DD H:mm:ss')} </Text>

<Text style={{color:"blue",fontSize:20,textAlign:"center",marginTop:15,marginBottom:5}}>Kategória: {item.kategoria\_nev}</Text>

</View>

}

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

/>

</View>

);

}

}

const styles = StyleSheet.create({

kekgomb: {

alignItems: "center",

backgroundColor: "blue",

padding: 10,

width:300,

marginLeft:"auto",

marginRight:"auto",

marginTop:10

}

});

A Törlés.js ugyan az, mint a Keresesidezet.js, csak lehet törölni is belőle. ezt csak az adminnak lehetséges. Ugyanúgy lehet keresni is. A törlés a Törlés gomb megnyomásával lehetséges. Amikor rányomnak a törlésre, akkor végrehajtódik egy delete utasítás, amely törli az általa választott tételt.

### Törlesvelemeny.js

import React from 'react';

import {StyleSheet, FlatList, ActivityIndicator, Text, View, TouchableOpacity, TextInput, Button, Menu } from 'react-native';

import {Picker} from '@react-native-picker/picker';

import moment from 'moment';

import 'moment/locale/hu'

moment.locale('hu')

const IP=require("./Ipcim")

export default class FetchExample extends React.Component {

constructor(props){

super(props);

this.state ={

isLoading: true,

szo:"",

bevitel1:"",

szam:"",

bevitelvelemenyid:"",

valaszto2:2,

dataSource:[],

dataSource2:[]

}

}

componentDidMount(){

fetch(IP.ipcim+"velemenyek")

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

.then((responseJson) => {

this.setState({

isLoading: false,

dataSource: responseJson,

}, function(){

});

})

.catch((error) =>{

console.error(error);

});

fetch(IP.ipcim+"idezet")

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

.then((responseJson) => {

this.setState({

isLoading: false,

dataSource2: responseJson,

}, function(){

});

})

.catch((error) =>{

console.error(error);

});

}

keres=()=>{

//alert("Hello")

var bemenet={

bevitel1:this.state.szo

}

fetch(IP.ipcim+"keresvelemeny", {

method: "POST",

body: JSON.stringify(bemenet),

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

}

)

.then(x => x.json())

.then(y => {

(JSON.stringify (y))

this.setState({dataSource: y})

}

);}

torles=(szam)=>{

alert(szam)

var bemenet={

bevitel1:szam

}

fetch(IP.ipcim+"torles\_velemeny", {

method: "DELETE",

body: JSON.stringify(bemenet),

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

}

)

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

.then(y => {

alert(y)

});

}

velemeny=()=>{

if(this.state.bevitel1=="" )

alert("Nem maradhat üresen!")

else{

var bemenet={

bevitel1:this.state.bevitel1,

bevitelvelemenyid:this.state.valaszto2

}

fetch(IP.ipcim+"velemeny", {

method: "POST",

body: JSON.stringify(bemenet),

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

}

)

.then(x => x.json())

.then(y => {

(JSON.stringify (y))

this.setState({dataSource: y})

}

);

}}

render(){

if(this.state.isLoading){

return(

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

<ActivityIndicator/>

</View>

)

}

return(

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

{/\*-----------------------------------------------------------------------------Felvital \*/}

<Text style={{marginLeft:10, marginRight:10, marginTop:10}}>Mit üzen neked? Ide írhatod.</Text>

<TextInput style={{height: 40,marginLeft:10, marginRight:10, marginTop:10,borderBottomWidth:2,backgroundColor:"white",borderBottomColor: 'blue',}}

placeholder="Üzenet szövege..."

onChangeText={(beirtszoveg)=>this.setState({bevitel1:beirtszoveg})}

value={this.state.bevitel1}

></TextInput>

<Picker

style={{backgroundColor:"#42adf5",color:"white",marginTop:10, marginBottom:10}}

selectedValue={this.state.valaszto2}

onValueChange={(ertek) =>

this.setState({valaszto2:ertek})

}>

{this.state.dataSource2.map(item=>

<Picker.Item label={item.idezet\_szoveg} value={item.idezet\_id} />

)}

</Picker>

<TouchableOpacity

style={styles.kekgomb}

onPress={async ()=>this.velemeny()}

>

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

</TouchableOpacity>

{/\*---------------------------------------------------kereses \*/}

<Text style={{marginLeft:10, marginTop:10, marginRight:10, marginBottom:10, fontSize:20}}>Add meg a keresendő szót:</Text>

<TextInput

style={{height: 40,marginLeft:10,marginRight:10, borderBottomColor:'blue', borderBottomWidth:2}}

placeholder="Szó megadása"

onChangeText={(beirtszoveg)=>this.setState({szo:beirtszoveg})}

value={this.state.szo}

/>

<TouchableOpacity style={{marginTop:10}}

onPress={()=>this.keres()}

>

<Text style={{textAlign:"center",color:"white",fontWeight:"bold",fontSize:15,marginLeft:10,marginRight:10, backgroundColor:"blue"}} >Keresés</Text>

</TouchableOpacity>

{/\*---------------------------------------------------talalatok \*/}

<FlatList

data={this.state.dataSource}

renderItem={({item}) =>

<View style={{borderWidth:2,borderColor:"blue", borderRadius:7, marginLeft:10, marginRight:10, marginTop:10}}>

<TouchableOpacity

style={styles.kekgomb}

onPress={async ()=>this.torles(item.velemeny\_id)}

>

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

</TouchableOpacity>

<Text style={{marginRight:"auto",marginLeft:"auto",color:"blue", fontSize:20,textAlign:"center",marginLeft:10, marginRight:10, marginTop:10}} >Sorszám: #{item.velemeny\_id}</Text>

<Text style={{marginRight:"auto",marginLeft:"auto",color:"green",fontStyle:"italic", fontSize:20,textAlign:"center",marginLeft:10, marginRight:10, marginTop:10}} >Felhasználó véleménye:</Text>

<Text style={{marginRight:"auto",marginLeft:"auto",color:"green",fontStyle:"italic", fontSize:20,textAlign:"center",marginLeft:10, marginRight:10, marginTop:10}} >{item.velemeny\_szoveg} </Text>

<Text style={{marginRight:"auto",marginLeft:"auto",color:"blue",fontSize:20,textAlign:"center",marginLeft:10, marginRight:10, marginTop:10}} >{item.idezet\_szoveg} </Text>

<Text style={{fontStyle:"italic", fontSize:20,textAlign:"left",marginLeft:10, marginRight:10, marginTop:10}} >{item.idezet\_konyv} {item.idezet\_fejezet\_vers} </Text>

<Text style={{color:"black",fontSize:20,textAlign:"center",marginTop:15,marginBottom:5}} >Időpont: {moment(item.idezet\_datum).format('YYYY MMMM DD, H:mm:ss')} </Text>

<Text style={{marginLeft:10, marginRight:10, marginTop:10,color:"blue",fontSize:20,textAlign:"left",marginTop:15,marginBottom:5}} >Kategória: {item.kategoria\_nev} </Text>

</View>

}

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

/>

</View>

);

}

}

const styles = StyleSheet.create({

kekgomb: {

alignItems: "center",

backgroundColor: "blue",

padding: 10,

width:300,

marginLeft:"auto",

marginRight:"auto",

marginTop:10

}

});

A Törlésvelemeny.js ugyanaz, mint a sima Velemeny.js csak itt is lehetőség van törölni. Itt is lehet véleményt írni, szintén érvényes a nem maradhat üresen.

### Velemeny.js

import React from 'react';

import {StyleSheet, FlatList, ActivityIndicator, Text, View, TouchableOpacity, TextInput, Button, Menu } from 'react-native';

import {Picker} from '@react-native-picker/picker';

import moment from 'moment';

import 'moment/locale/hu'

moment.locale('hu')

const IP=require("./Ipcim")

export default class FetchExample extends React.Component {

constructor(props){

super(props);

this.state ={

isLoading: true,

szo:"",

bevitel1:"",

bevitelvelemenyid:"",

valaszto2:2,

dataSource:[],

dataSource2:[]

}

}

componentDidMount(){

fetch(IP.ipcim+"velemenyek")

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

.then((responseJson) => {

this.setState({

isLoading: false,

dataSource: responseJson,

}, function(){

});

})

.catch((error) =>{

console.error(error);

});

fetch(IP.ipcim+"idezet")

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

.then((responseJson) => {

this.setState({

isLoading: false,

dataSource2: responseJson,

}, function(){

});

})

.catch((error) =>{

console.error(error);

});

}

velemeny=()=>{

if(this.state.bevitel1=="" )

alert("Nem maradhet üresen!")

else{

var bemenet={

bevitel1:this.state.bevitel1,

bevitelvelemenyid:this.state.valaszto2

}

fetch(IP.ipcim+"velemeny", {

method: "POST",

body: JSON.stringify(bemenet),

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

}

)

.then(x => x.json())

.then(y => {

(JSON.stringify (y))

this.setState({dataSource: y})

}

);

}}

keres=()=>{

//alert("Hello")

var bemenet={

bevitel1:this.state.szo

}

fetch(IP.ipcim+"keresvelemeny", {

method: "POST",

body: JSON.stringify(bemenet),

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

}

)

.then(x => x.json())

.then(y => {

(JSON.stringify (y))

this.setState({dataSource: y})

}

);}

render(){

if(this.state.isLoading){

return(

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

<ActivityIndicator/>

</View>

)

}

return(

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

{/\*-----------------------------------------------------------------------------Felvital \*/}

<Text style={{marginLeft:10, marginRight:10, marginTop:10}}>Mit üzen neked? Ide írhatod.</Text>

<TextInput style={{height: 40,marginLeft:10, marginRight:10, marginTop:10,backgroundColor:"white", borderBottomColor:'blue',borderBottomWidth:2}}

placeholder="Üzenet szövege..."

onChangeText={(beirtszoveg)=>this.setState({bevitel1:beirtszoveg})}

value={this.state.bevitel1}

></TextInput>

<Picker

style={{backgroundColor:"#42adf5",color:"white",marginTop:10, marginBottom:10}}

selectedValue={this.state.valaszto2}

onValueChange={(ertek) =>

this.setState({valaszto2:ertek})

}>

{this.state.dataSource2.map(item=>

<Picker.Item label={item.idezet\_szoveg} value={item.idezet\_id} />

)}

</Picker>

<TouchableOpacity

style={styles.kekgomb}

onPress={async ()=>this.velemeny()}

>

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

</TouchableOpacity>

{/\*---------------------------------------------------kereses \*/}

<Text style={{marginLeft:10, marginTop:10, marginRight:10, marginBottom:10, fontSize:20}}>Add meg a keresendő szót:</Text>

<TextInput

style={{height: 40,marginLeft:10,marginRight:10, borderBottomColor:'blue',borderBottomWidth:2}}

placeholder="Szó megadása"

onChangeText={(beirtszoveg)=>this.setState({szo:beirtszoveg})}

value={this.state.szo}

/>

<TouchableOpacity style={{marginTop:10}}

onPress={()=>this.keres()}

>

<Text style={{textAlign:"center",color:"white",fontWeight:"bold",fontSize:15,marginLeft:10,marginRight:10, backgroundColor:"blue"}} >Keresés</Text>

</TouchableOpacity>

{/\*---------------------------------------------------talalatok \*/}

<FlatList

data={this.state.dataSource}

renderItem={({item}) =>

<View style={{borderWidth:2,borderColor:"blue", borderRadius:7, marginLeft:10, marginRight:10, marginTop:10}}>

<Text style={{marginRight:"auto",marginLeft:"auto",color:"blue", fontSize:20,textAlign:"center",marginLeft:10, marginRight:10, marginTop:10}} >Sorszám: #{item.velemeny\_id}</Text>

<Text style={{marginRight:"auto",marginLeft:"auto",color:"green",fontStyle:"italic", fontSize:20,textAlign:"center",marginLeft:10, marginRight:10, marginTop:10}} >Felhasználó véleménye:</Text>

<Text style={{marginRight:"auto",marginLeft:"auto",color:"green",fontStyle:"italic", fontSize:20,textAlign:"center",marginLeft:10, marginRight:10, marginTop:10}} >{item.velemeny\_szoveg} </Text>

<Text style={{marginRight:"auto",marginLeft:"auto",color:"blue",fontSize:20,textAlign:"center",marginLeft:10, marginRight:10, marginTop:10}} >{item.idezet\_szoveg} </Text>

<Text style={{fontStyle:"italic", fontSize:20,textAlign:"left",marginLeft:10, marginRight:10, marginTop:10}} >{item.idezet\_konyv} {item.idezet\_fejezet\_vers} </Text>

<Text style={{color:"black",fontSize:20,textAlign:"center",marginTop:15,marginBottom:5}} >Időpont: {moment(item.idezet\_datum).format('YYYY MMMM DD, H:mm:ss')} </Text>

<Text style={{marginLeft:10, marginRight:10, marginTop:10,color:"blue",fontSize:20,textAlign:"left",marginTop:15,marginBottom:5}} >Kategória: {item.kategoria\_nev} </Text>

</View>

}

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

/>

</View>

);

}

}

const styles = StyleSheet.create({

kekgomb: {

alignItems: "center",

backgroundColor: "blue",

padding: 10,

width:300,

marginLeft:"auto",

marginRight:"auto",

marginTop:10

}

});

A Velemeny.js lehetővé teszi, hogy véleményt írjunk egy általunk választott idézethez. A lenyíló menüben ki lehet választani, hogy melyikről szereznénk véleményt írni és a szövegmezőbe leírjuk a gondolatainkat és a Küldés gombbal elküldjük azt. Itt a küldés akkor történik meg, ha nem üres a szövegbeviteli mező.

### Welcome.js

import React, {Component} from 'react';

import {Alert, Text, StyleSheet, View} from 'react-native';

export default class ButtonBasics extends Component {

render() {

return (

<View style={styles.container}>

<Text style={{textAlign:"center"}}>Üdvözöllek az alkalmazásban aminek a témája Bibliai igékkel kapcsolatos. Ebben az alkalmazásban elmondhatja, hogy mit üzen az adott Ige.</Text>

</View>

);

}

}

const styles = StyleSheet.create({

container: {

flex: 1,

justifyContent: 'center',

},

buttonContainer: {

margin: 20,

},

alternativeLayoutButtonContainer: {

margin: 20,

flexDirection: 'row',

justifyContent: 'space-between',

},

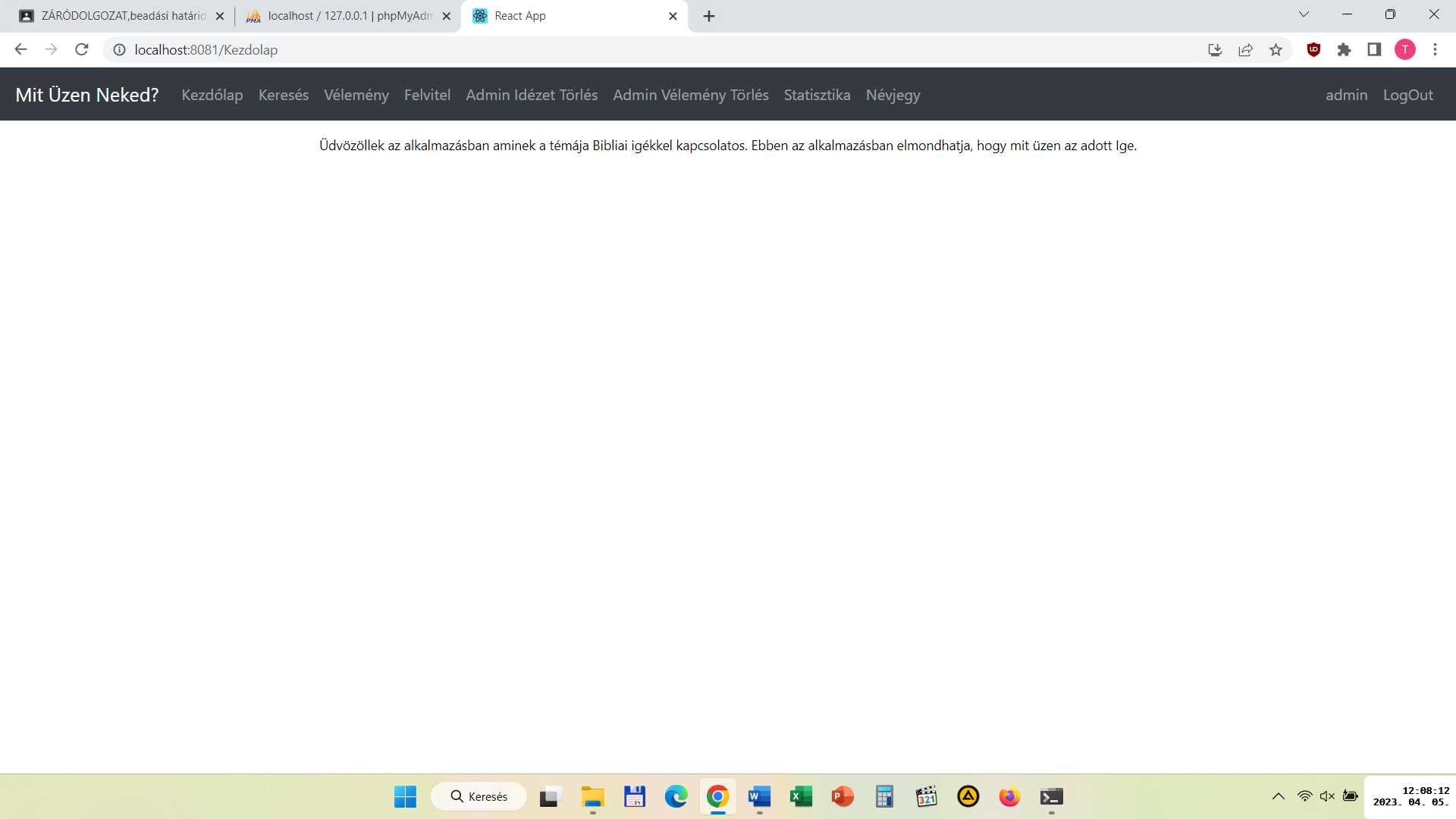
});

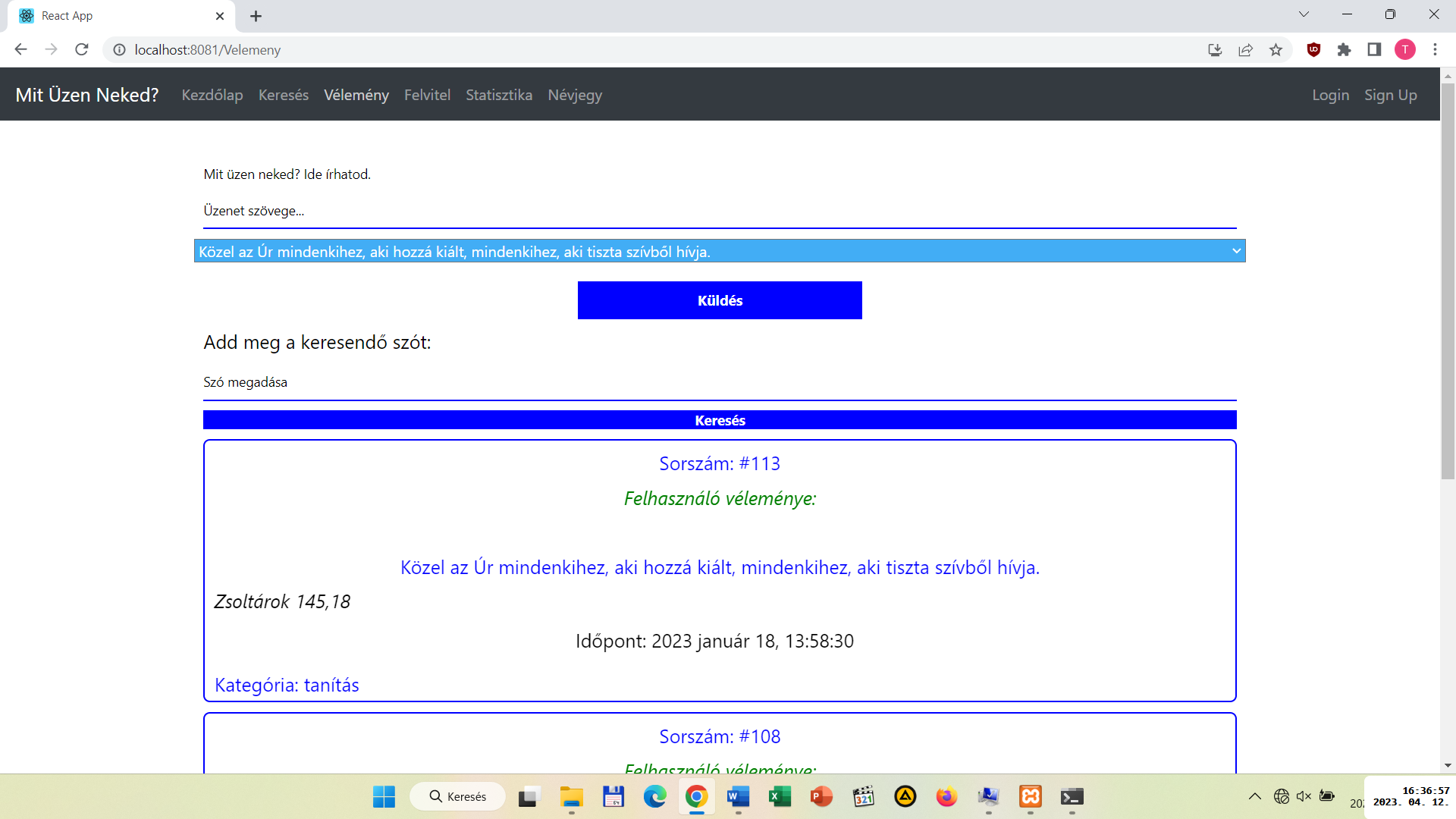
A Welcome.js üdvölés, és ez a kiindulópont.

## Tesztelés

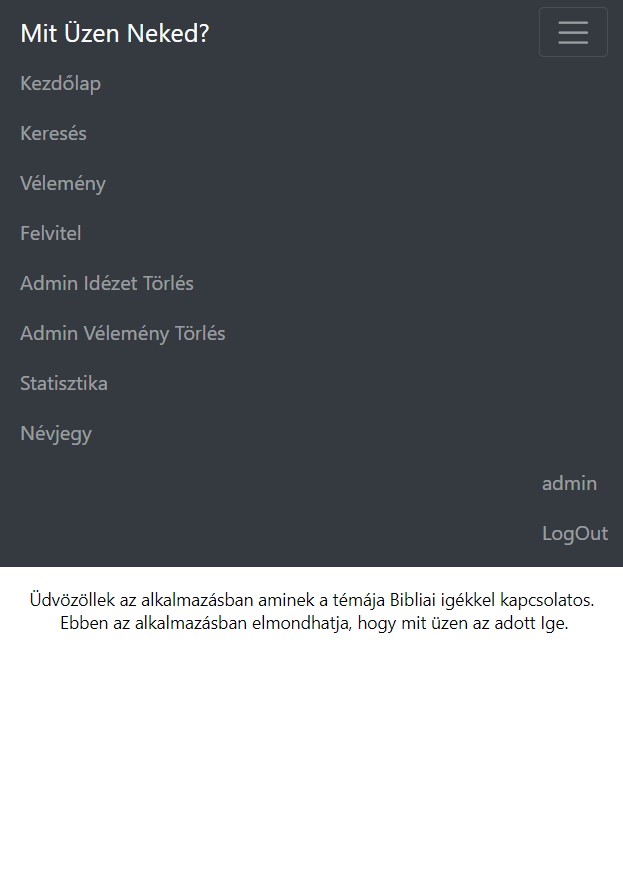
### Web:

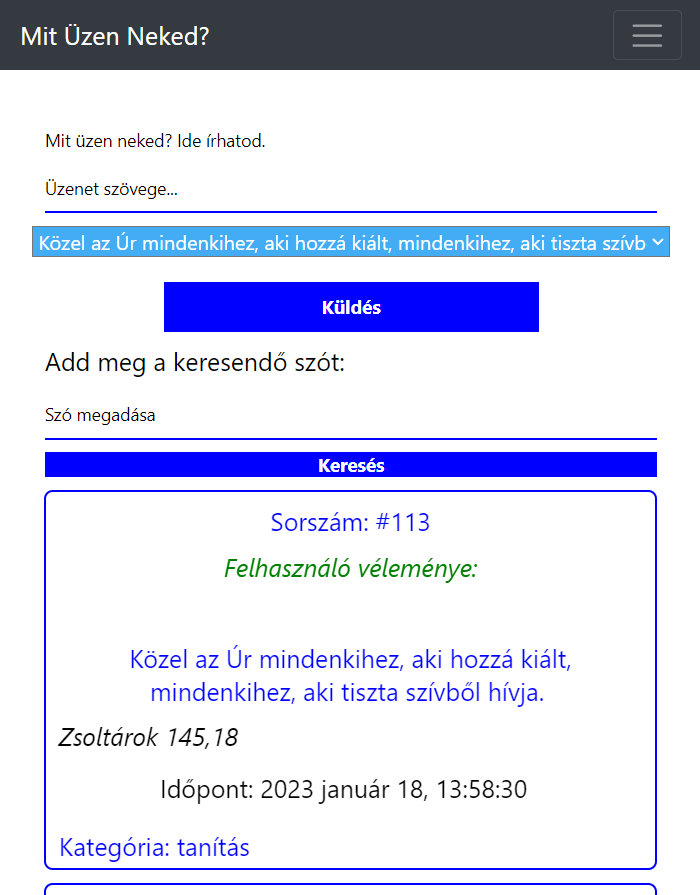
Az oldal megjelenése nagy kijelző (laptop, pc pl 1920x1080)





Az oldal megjelenése kicsi kijelzőn (mobil pl 720x1600)





Az oldal reszponzív, ami azt jelenti, hogy az oldal különböző kijelző méretek esetén is jól használható. A kisebb kijelző esetén átrendeződnek az oldal elemei a kijelző méretétől függően.

A fenti példa a kezdőlap és a vélemény.

## Továbbfejlesztési lehetőségek

A következő fejlesztések, funkciók hiányoznak még:

### Mobil applikáció:

* Fájlba mentés.
* Megosztás.
* Oldalak hozzáadása, hogy ne legyen végtelen hosszú a menü.
* Esetleges hibák javítása.

### Weblap:

* Több elem törlése admin.
* Oldalak hozzáadása.

# Összegzés

A szakdolgozatom azoknak lehet hasznos, akik érdeklődnek a Biblia iránt, és szeretnék másokkal megosztani a gondolataikat. A munkámat jónak értékelem és sok hasznos dolgot tanultam, amit máshol is használhatok. Amikor megmutattam a barátaimnak a szakdolgozatomat, tetszett nekik, és nagyon hasznosnak mondták. Ez nagyon jól esett nekem. És köszönet a tanároknak a lehetőséget, hogy egy új programozási nyelvet, a React Native-t és sok más érdekes dolgot is megtanultam. Köszönöm a segítséget is és a tanácsokat, ötletet a szakdolgozatomhoz.

# Irodalomjegyzék

React native:

<https://reactnative.dev/docs/environment-setup>

Néhány fontosabb node.js modulok és használatuk:

<https://www.npmjs.com/package/express>

<https://www.npmjs.com/package/cors>

<https://www.npmjs.com/package/mysql>

<https://www.npmjs.com/package/bcryptjs>

<https://www.npmjs.com/package/react>

<https://www.npmjs.com/package/react-native-web>

<https://www.npmjs.com/package/moment>

<https://www.npmjs.com/package/@react-native-picker/picker>

<https://www.npmjs.com/package/@react-navigation/drawer>

# Eredetiségnyilatkozat

Alulírott Mező Tamás Zsolt nyilatkozom, hogy záró dolgozatom a saját szellemi termékem.

Debrecen, 2023. ápr. 12. Mező Tamás Zsolt

.

aláírás