



BKSZC Pogány Frigyes Technikum

## **Aquntant**

szoftverfejlesztő és -tesztelő  
vizsgaremek

2022. május

Készítette:  
Máté Andrea  
Börzsönyi Bence

Konzulens:  
Tóth József

# Tartalom

<b>Tartalom</b>	<b>1</b>
<b>A feladat</b>	<b>2</b>
<b>GitHub repository</b>	<b>2</b>
<b>Követelmények</b>	<b>2</b>
<b>Az adatbázis elkészítése</b>	<b>2</b>
User tábla	3
Partner tábla	3
Tax tábla	4
Movement tábla	5
<b>Az API elkészítése</b>	<b>6</b>
<b>Backend és frontend közötti kapcsolat</b>	<b>11</b>
<b>Kliens oldal az API-hoz</b>	<b>17</b>
Bejelentkezés nélkül	17
Bejelentkezve	22
<b>Továbbfejlesztés</b>	<b>29</b>
<b>Források</b>	<b>29</b>

# A feladat

Szeretnénk ügyfeleinknek egy olyan oldalt biztosítani, ahol biztonságosan rögzíthetik pénzmozgásaikat egy egyszerűen kezelhető felületen. Ezért hoztunk létre ehhez egy weboldalt. Itt felvehetik egy adatbázisba a tranzációikat és a partnereikkel való kapcsolattartáshoz elengedhetetlen adatokat. Ügyfeleinknek adunk lehetőséget felhasználói fiók létesítésére is. Adataikat online adatbázisban tároljuk, amit API-n keresztül lehet elérni. Az elkészített webalkalmazás erről az API-ról töltik le és jelenítik meg az adatokat.

## GitHub repository

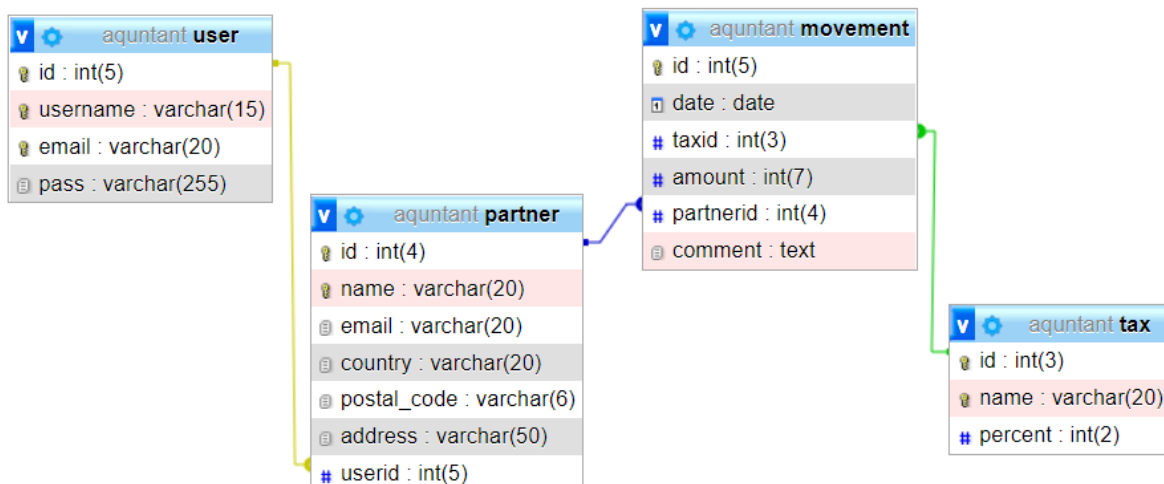
Az alkalmazás kódja ebben a repository-ban érhető el:  
<https://github.com/mt-andrea/aquantant>

## Követelmények

Az alkalmazás működéséhez szükséges, hogy a node js legfrissebb LTS (jelenleg 16.14.2) verzióját használják a szerver oldalon. Ennek hiányában az oldal bizonyos elemei nem fognak megfelelően működni.

## Az adatbázis elkészítése

Az adatbázist MySQL adatbázis-kezelővel és a PHPMyAdmin programmal készítettük el. Az adatokat négy táblára bontottuk:



## User tábla

Ez a tábla tartalmazza a felhasználók adatait.

- id: A felhasználó azonosítója (elsődleges kulcs)
- username: A felhasználónév
- email: A felhasználó email címe
- pass: A felhasználó hashelt jelszava

A tábla szerkezete:

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	int(5)			No	None		AUTO_INCREMENT	Change  Drop  More
2	username	varchar(15)	utf8mb4_hungarian_ci		No	None			Change  Drop  More
3	email	varchar(20)	utf8mb4_hungarian_ci		No	None			Change  Drop  More
4	pass	varchar(255)	utf8mb4_hungarian_ci		No	None			Change  Drop  More

Azért, hogy ne legyen duplikált felhasználónév és email regisztrálva, UNIQUE indexet alkalmaztunk.

Action	Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
Edit  Rename  Drop	PRIMARY	BTREE	Yes	No	id	8	A	No	
Edit  Rename  Drop	Unique	BTREE	Yes	No	username	8	A	No	
					email	8	A	No	

A tesztadatok:

				id	username	email	pass
<input type="checkbox"/>	Edit	Copy	Delete	1	tester1	test@email.com	\$2b\$10\$45TWVxfW4J6bEQR9O9JYlulbK7VkoIq.MD75srZSqs/...
<input type="checkbox"/>	Edit	Copy	Delete	2	fixer1	bugfix@email.com	\$2b\$10\$HD7zOpPr9AixFb2NgxwG/eNZpj2Kcl02YKspY3aa/XT...
<input type="checkbox"/>	Edit	Copy	Delete	3	pyot1	tpyo@email.com	\$2b\$10\$gaDopZOXuhvRINnZUIlBJOQ/dTUnxZP58DdM10GMtHh...
<input type="checkbox"/>	Edit	Copy	Delete	4	router2	route@email.com	\$2b\$10\$eLk7gq4RWtV4NtnQAo4cO.BtIL.AkJ5PaojBYkVg4Pu...
<input type="checkbox"/>	Edit	Copy	Delete	5	borzsteszt	tesztnew@email.com	\$2b\$10\$ILRNxURXGf3BxDGfOdkcdeyqllecUITv8u39bSH.NZI...
<input type="checkbox"/>	Edit	Copy	Delete	6	react	react@gmail.com	\$2b\$10\$9W/NRQs2jxEYe4Kim8J8fOQxliIWg9QPUYrrXB8RspHN...
<input type="checkbox"/>	Edit	Copy	Delete	8	react3	react3@gmail.com	\$2b\$10\$BwXDOPpYLJeRZfbvWb5AC.kiWU2bSWf2DM.9HJfReTI...
<input type="checkbox"/>	Edit	Copy	Delete	11	react2	react2@gmail.com	\$2b\$10\$ZRBxRxCnkHq1lamf8HgYcsennlKERoOEYUuX1aM6Tbd...

## Partner tábla

A partner táblába vesszük fel a felhasználóknak a partnereit.

- id: A partner azonosítója (elsődleges kulcs)
- name: A partner neve
- email: A partner email címe
- country: A partner országa
- postal\_code: A partner irányítószáma
- address: A partner kapcsolattartó / székhelyének címe
- userid: A partnerhez tartozó felhasználónak azonosítója (idegen kulcs)

A tábla szerkezete:

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	int(4)			No	None		AUTO_INCREMENT	Change  Drop  More
2	name	varchar(20)	utf8mb4_hungarian_ci		No	None			Change  Drop  More
3	email	varchar(20)	utf8mb4_hungarian_ci		No	None			Change  Drop  More
4	country	varchar(20)	utf8mb4_hungarian_ci		No	None			Change  Drop  More
5	postal_code	varchar(6)	utf8mb4_hungarian_ci		No	None			Change  Drop  More
6	address	varchar(50)	utf8mb4_hungarian_ci		No	None			Change  Drop  More
7	userid	int(5)			No	None			Change  Drop  More

## A tesztadatok:

				id	name	email	country	postal_code	address	userid
<input type="checkbox"/>				1	Tester Toby	test@email.com	testnation1	420	Test city,test str. 21	1
<input type="checkbox"/>				2	Bugfix Brigitte	bugfix@email.com	Fixination	2233	Fixer's city,Fixin str. 21	2
<input type="checkbox"/>				3	Typo Thomas	tpyo@email.com	Typonation	2737	Typyoers's city,topy str. 12	3
<input type="checkbox"/>				4	Router Richard	route@email.com	Routernation	3437	Routers's city,Router str. 12	4
<input type="checkbox"/>				5	Macy's	info@macys.com	US, Florida	33166	Miami Springs, 2549 Arbutus Drive	1
<input type="checkbox"/>				6	Citigroup	info@citigroup.com	US, Virginia	22304	Alexandria, 2983 Perine Street	1
<input type="checkbox"/>				7	JetBlue Airways	info@jetblue-airways	US, Georgia	30097	Duluth, 168 Junior Avenue	1
<input type="checkbox"/>				8	Caterpillar	info@caterpillar.com	US, Washington DC	30097	Washington, 3912 Hickory Lane	1
<input type="checkbox"/>				9	Hormel Foods	info@hormel-foods.co	US, Texas	77036	Houston, 1706 Mulberry Street	1
<input type="checkbox"/>				10	Cintas	info@cintas.com	Hungary	8564	Ugod, Kárpát u. 27.	3
<input type="checkbox"/>				11	Mondelez Internation	info@mondelez.com	Hungary	9371	Vitnyéd, Wesselényi u. 68.	3
<input type="checkbox"/>				12	Newell Brands	info@newell-brands.c	Hungary	9736	Csepreg, Belgrád rkp. 12.	3
<input type="checkbox"/>				13	Dominion Energy	info@dominion.com	Hungary	2194	Tura, Síp utca 48.	3
<input type="checkbox"/>				14	Realogy Holdings	info@realogy.com	Hungary	2760	Nagykátá, Munkácsy Mihály út 28.	3

## Tax tábla

Ez a tábla tartalmazza az adó adatokat.

- id: Az adó azonosítója (elsődleges kulcs)
- name: Az adó neve
- percent: Az adó százaléka

A tábla szerkezete:

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	int(3)			No	None		AUTO_INCREMENT	Change  Drop  More
2	name	varchar(20)	utf8mb4_hungarian_ci		No	None			Change  Drop  More
3	percent	int(2)			No	None			Change  Drop  More

Azért, hogy ne szerepeljen egy adó kétszer a táblában, UNIQUE indexet állítottunk be.

Action	Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
Edit  Rename  Drop	PRIMARY	BTREE	Yes	No	id	8	A	No	
Edit  Rename  Drop	Unique	BTREE	Yes	No	username	8	A	No	
					email	8	A	No	

## A tesztadatok:

				id	name	percent
<input type="checkbox"/>				1	altalános	27
<input type="checkbox"/>				2	tej-/gabonatermek	18
<input type="checkbox"/>				3	gyogyszer/gyogyszer	5
<input type="checkbox"/>				4	konyv/ujsg	5
<input type="checkbox"/>				5	haziasított állat és	5
<input type="checkbox"/>				6	tojás/hal	5
<input type="checkbox"/>				7	tárgyi adómentes	0
<input type="checkbox"/>				8	alanyi adómentes	0

# Movement tábla

Itt tároljuk a felhasználók tranzakcióit.

- id: A tranzakció azonosítója (elsődleges kulcs)
- date: A tranzakció dátuma
- taxid: Az adó azonosítója (idegen kulcs)
- amount: A tranzakció összege
- partnerid: A partner azonosítója (idegen kulcs)
- comment: A tranzakcióhoz fűzött megjegyzés / megnevezés

A tábla szerkezete:

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	int(5)			No	None		AUTO_INCREMENT	Change  Drop  More
2	date	date			No	None			Change  Drop  More
3	taxid	int(3)			No	None			Change  Drop  More
4	amount	int(7)			No	None			Change  Drop  More
5	partnerid	int(4)			No	None			Change  Drop  More
6	comment	text	utf8mb4_hungarian_ci		Yes	NULL			Change  Drop  More

A tesztadatok:

					id	date	taxid	amount	partnerid	comment
<input type="checkbox"/>	Edit	Copy	Delete		1	2022-11-01	6	-248696	9	Procurement of goods
<input type="checkbox"/>	Edit	Copy	Delete		2	2022-03-18	7	302286	11	Sale
<input type="checkbox"/>	Edit	Copy	Delete		4	2022-11-21	6	-733131	4	Procurement of goods
<input type="checkbox"/>	Edit	Copy	Delete		5	2022-04-04	4	144381	7	Sale
<input type="checkbox"/>	Edit	Copy	Delete		6	2022-09-28	3	-822387	11	Procurement of goods
<input type="checkbox"/>	Edit	Copy	Delete		7	2022-02-09	2	124729	11	Sale
<input type="checkbox"/>	Edit	Copy	Delete		8	2022-07-03	5	624999	9	Sale
<input type="checkbox"/>	Edit	Copy	Delete		9	2022-06-22	3	64760	9	Sale
<input type="checkbox"/>	Edit	Copy	Delete		10	2022-09-28	3	887767	11	Sale
<input type="checkbox"/>	Edit	Copy	Delete		11	2022-10-13	8	943888	9	Sale
<input type="checkbox"/>	Edit	Copy	Delete		12	2022-03-03	2	345127	9	Sale
<input type="checkbox"/>	Edit	Copy	Delete		13	2022-05-31	2	-827014	14	Procurement of goods
<input type="checkbox"/>	Edit	Copy	Delete		14	2022-08-03	5	802472	6	Sale
<input type="checkbox"/>	Edit	Copy	Delete		15	2022-05-24	3	704809	2	Sale
<input type="checkbox"/>	Edit	Copy	Delete		16	2022-12-26	4	116005	2	Sale
<input type="checkbox"/>	Edit	Copy	Delete		17	2022-02-26	7	-242189	1	Procurement of goods
<input type="checkbox"/>	Edit	Copy	Delete		18	2022-02-09	1	249391	13	Sale
<input type="checkbox"/>	Edit	Copy	Delete		19	2022-08-17	6	-61321	7	Procurement of goods
<input type="checkbox"/>	Edit	Copy	Delete		20	2022-12-29	1	-346662	12	Procurement of goods
<input type="checkbox"/>	Edit	Copy	Delete		21	2022-04-28	6	495080	1	Sale
<input type="checkbox"/>	Edit	Copy	Delete		22	2022-06-04	2	520912	3	Sale
<input type="checkbox"/>	Edit	Copy	Delete		23	2022-04-14	7	-574770	6	Procurement of goods
<input type="checkbox"/>	Edit	Copy	Delete		24	2022-10-23	3	667578	7	Sale
<input type="checkbox"/>	Edit	Copy	Delete		25	2022-01-01	3	-368329	8	Procurement of goods
<input type="checkbox"/>	Edit	Copy	Delete		26	2022-02-13	8	-523194	5	Procurement of goods

# Az API elkészítése

Az API-t node.js alkalmazásként készítettük el express framework segítségével.  
A fájlt a server/backend/app.js útvonalon lehet megtalálni.

Az alkalmazásba a következő modulokat importáltuk:

- express
- mysql
- dotenv
- cors
- bcrypt
- jsonwebtoken

Jsonwebtoken (jwt) segítségével tudjuk elérni azt, hogy az API védettebb legyen.

A bcrypt modul segít a felhasználóink jelszavának titkosításában (hashelésében).

A dotenv modulra azért volt szükségünk, hogy a jwt által titkosított adatokat vissza tudjuk fejteni egy titkos kulcs segítségével.

A titkos kulcsot a következő node paranccsal generáltuk le a terminálban:

```
require('crypto').randomBytes(64).toString('hex')
```

Azért, hogy elérjük tetszőleges címről az API-t a cors modult alkalmaztuk.

A mysql modul pedig azért kell, hogy az adatbázist tudjuk használni.

A modulokat az API elején meghívtuk és úgy konfiguráltuk, hogy a felhasználó tudjon adatot küldeni az alkalmazás segítségével.

```
require('dotenv').config();  
...  
const express = require("express");  
const app = express();  
app.use(express.urlencoded({extended: true}))  
app.use(express.json());  
const mysql = require('mysql');  
const cors = require('cors');  
const bcrypt = require('bcrypt');  
require('dotenv').config();  
const jwt = require('jsonwebtoken');  
const saltRounds = 10; |  
app.use(cors());
```

Készítettünk egy poolt, ami az adatbázis és az API közti kapcsolatot hozza létre. A multipleStatements beállítás engedélyezi az API-nak, hogy egyszerre több lekérdezést tudjon kezelni.

```
var pool = mysql.createPool({  
  host: 'localhost',  
  port: "3306",  
  user: 'root',  
  password: '',  
  database: 'aquantant',  
  multipleStatements: true  
});
```

Az API-t a 4000-es porton futtatjuk.

```
app.listen(4000, () => {  
  console.log("Server started on port 4000...")  
});
```

A regisztráció elején ellenőrizzük, hogy az email vagy a felhasználónév foglalt-e. Ha mind a kettő szabad, akkor titkosítjuk a megadott jelszót és felvesszük az adatbázisunkba.

```
app.post("/register", (req, res) => {  
  const {username, email, password} = req.body  
  
  const q = "SELECT email FROM user WHERE email = ?;" +  
    "SELECT username FROM user WHERE username = ?;"  
  pool.query(q, [email, username], (error, result) => {  
    if(result[0].length > 0) {  
      return res.send({message: "The email is in use"});  
    }  
    if(result[1].length > 0) {  
      return res.send({message: "The username is in use"})  
    }  
  
    let hashPass = bcrypt.hashSync(password, 10);  
  
    placeholders = [username, email, hashPass]  
    q3 = "INSERT INTO user (username, email, pass) VALUES (?);"  
    pool.query(q3, [placeholders], (error, result) => {  
      if(!error){  
        return res.send({message: "Success"});  
      } else {  
        return res.send({message: "Failure"})  
      }  
    })  
  })  
})  
})
```



A bejelentkezésnél ha a megadott felhasználónév szerepel az adatbázisunkban, akkor a jelszavakat összehasonlítjuk bcrypt.compareSync metódusával. Ha minden megfelel, akkor a felhasználó kap egy tokenet, amivel képes használni az alkalmazást.

```
app.post("/login", (req, res) => {
  const { username, password } = req.body;
  const q = "SELECT * FROM user WHERE username = ?";
  pool.query(q, [username],
    function (error, result) {
      if (error)
        return res.send({message: "Database error"});
      else if (result.length == 0) {
        return res.send({message: "Incorrect username or password"})
      } else {
        user = JSON.parse(JSON.stringify(result[0]));
        if (!bcrypt.compareSync(password, user.pass))
          return res.send({message: "Incorrect username or password"})
        const token = jwt.sign(user, process.env.TOKEN_SECRET)
        res.json({ token: token, message: "Success" })
      }
    }
  )
})
```

A bejelentkezés utáni lekérdezések egy middlewaret használnak, ami ellenőrzi, hogy a token, amivel a felhasználó rendelkezik érvényes-e.

```
function authenticateToken(req, res, next) {
  const authHeader = req.headers['authorization']
  const token = authHeader && authHeader.split(' ')[1]
  if (!token)
    return res.status(401).send({ message: "Authorization required!" })
  jwt.verify(token, process.env.TOKEN_SECRET, (err, user) => {
    if (err)
      return res.status(403).send({ message: "Access denied!" })
    req.user = user
    next()
    //console.log(user)
  })
}
```

A jelszó megváltoztatásához a compareSync metódussal ellenőriztük, hogy a régi jelszavak megegyeznek-e. Ha igen, akkor az új jelszót hashelve eltároljuk.

Az email megváltoztatásához hasonlóan járultunk el

```
app.patch("/change/pass", authenticateToken ,(req, res) => {
  const {oldpassword,newpassword} = req.body;
  const q = "UPDATE user SET pass=? WHERE username=?";
  let newHashPass = bcrypt.hashSync(newpassword, saltRounds);

  if(bcrypt.compareSync(oldpassword, req.user.pass)) {
    pool.query(q,[newHashPass, req.user.username], (error, result) =>{
      if(!error) {
        res.send(result)
      } else {
        res.send(error)
      }
    })
  } else {
    console.log("Error") //PH message
  }
})
```

A tranzakciók megjelenítéséhez SELECT lekérdezést készítettünk, ahol a dátumot formáztuk és az idegen kulcsokat INNER JOIN-nal összekötöttük.

Ebből a listázásból csináltunk egy “szűrt” változatot is, amely a /listing/filtered útvonalon érhető el. A kiválasztott feltételeknek megfelelően kiegészül az alap lekérdezésünk.

```
app.get("/listing", authenticateToken, (req, res)=> {
  const q = "SELECT DATE_FORMAT(movement.date, '%Y-%m-%d') as date, movement.amount, "+
    "partner.name AS name, partner.address, user.username ,movement.comment,tax.name as tax FROM movement "+
    "INNER JOIN partner ON partner.id=movement.partnerid "+
    "INNER JOIN user ON user.id=partner.userid "+
    "INNER JOIN tax ON tax.id=movement.taxid "+
    "WHERE user.username=?";
  pool.query(q,[req.user.username], (error, results) => {
    if (!error) {
      res.send(results);
    } else {
      res.send(error);
    }
  })
})
```

A partnerek hozzáadásához és partnerek listázásához a tokenben tárolt adatokat felhasználtuk. Emiatt itt nem kellett használnunk INNER JOIN-t.

```
app.post("/add/partner", authenticateToken,(req,res) => {
  const {name,email,country,postal_code,address} = req.body
  const userId =req.user.id
  const q = "INSERT INTO partner (name,email,country,postal_code,address,userid) VALUES (?);"
  const placeholders = [name,email,country,postal_code,address,userId]
  pool.query(q,[placeholders], (error, result) => {
    if(!error) {
      res.send(result)
    } else {
      res.send(error)
    }
  })
})
```

```
app.post("/choices/partner", authenticateToken,(req,res)=> {
  const userid =req.user.id
  const q ="SELECT id,name,email,concat(country, ' ',postal_code, ' ',address) as address FROM partner WHERE userid=?;"
  pool.query(q, [userid], (error, result) => {
    if(!error) {
      res.send(result)
    } else {
      res.send(error)
    }
  })
})
```

Készítettünk egy összegzés lekérdezést is, aminek segítségével a felhasználó láthatja jövedelmét, kiadását és ezeknek az adatoknak segítségével egyenlegét..

```
app.post("/summary", authenticateToken,(req, res) => { //unused
  const userid = req.user.id
  const q = "SELECT sum(case when movement.amount < 0 then movement.amount else 0 end) AS negativ, "+
    "sum(case when movement.amount > 0 then movement.amount else 0 end) AS pozitiv FROM movement "+
    "INNER JOIN partner ON partner.id=movement.partnerid "+
    "INNER JOIN user ON user.id=partner.userid "+
    "WHERE user.id=?;";
  pool.query(q,[userid], (error, results) => {
    if (!error) {
      res.send(results);
    } else {
      res.send(error);
    }
  })
})
```

## Backend és frontend közötti kapcsolat

A backend és frontend közötti kommunikációt fetch API-val oldottuk meg.

useState segítségével tároljuk a felhasználó által megadott adatokat és ez alapján tovább küldjük az API-nak feldolgozásra.

A regisztrációnál ellenőrizzük, hogy van-e üres mező. Ha nincs, akkor a megadott két jelszót még megnézzük. Ezek után felvesszük a felhasználó fiókját az adatbázisunkba és átirányítjuk.

```
function register(e) {
  e.preventDefault()
  if (data.username=="" || data.password=="" || data.email=="") {
    setMessage("Username, email or password is empty!")
    return
  }

  if (data.retype !== data.password) {
    e.preventDefault()
    setMessage("The passwords do not match!")
    return
  }

  fetch('http://localhost:4000/register', {
    method: 'POST',
    headers: {
      'Content-type': 'application/json;charset=utf-8'
    },
    body: JSON.stringify({
      "username": data.username,
      "password": data.password,
      "email": data.email,
      "retype": data.retype
    })
  })

  .then((response) => response.json())
  .then(json => {
    if(json.message == "Success") {
      navigate("/successful-reg")
    } else {
      setMessage(json.message)
    }
  })
  .catch(err => console.log(err))
}
```

A bejelentkezésnél ellenőrizzük, hogy üresek-e a beviteli mezők. Ha nincs üres mező, akkor küldünk egy POST kérést. Ha jó adatokat adtunk meg, akkor kapunk egy token, amit a sessionStorageben tárolunk és átirányít az oldal.

```
function signin(e) {
  e.preventDefault()
  if (data.username == "" || data.password == "") {
    setMessage("Username or password is empty!")
    return
  }/*else{
    props.logged=true
  }*/

  fetch('http://localhost:4000/login', {
    method: 'POST',
    headers: {
      'Content-type': 'application/json;charset=utf-8'
    },
    body: JSON.stringify({
      "username": data.username,
      "password": data.password
    })
  })
    .then((response) => response.json())
    .then(json => {
      if (json.message == "Success") {
        sessionStorage.token = json.token
        props.beallit({token:json.token}) //currently soft refresh for navbar
        navigate("/transactions")
      } else {
        setMessage(json.message)
      }
    })
    .catch(err => console.log(err))
}
```

A listázáshoz használt fetchben már el kell küldenünk a saját bearer tokenünket is. Ahhoz, hogy rögtön lefusson 1-szer a readIn metódus, useEffect hookot használtunk.

```
function readIn() {
  fetch(url, {
    method: 'GET',
    headers: {
      'Authorization': token
    }
  })
  .then((response) => response.json())
  .then((json) => setInformation(json))
  .catch(err => console.log(err))
}

useEffect(() => {
  readIn();
}, []);
```

A szűréshez a beírt adatokat elküldjük az API-nak, majd vissza kapjuk a feltételeknek megfelelő tranzakciókat.

```
function filtering() {
  //e.preventDefault()
  fetch(url2, {
    method: 'POST',
    headers: {
      'Authorization': token,
      'Content-type': 'application/json;charset=utf-8'
    },
    body: JSON.stringify({
      "in_out": data.in_out,
      "month": data.month,
      "partner": data.partner
    })
  })
  .then((response) => response.json())
  .then((json) => setInformation(json))
  .catch(err => console.log(err))
}
```

Ennek működéséhez ismét useEffect hookot használtunk, csak annyi különbséggel, hogy most a megadott adatok változása esetén fusson le a filtering metódus.

```
useEffect(() => { |  
  
  filtering()  
}, [data.in_out, data.month, data.partner])
```

A felhasználó jelszavának módosításához bekérjük a régi jelszót és az új jelszót kétszer. Ha minden adat stimmel, akkor a jelszó módosítva lesz és a felhasználó ki lesz jelentkezve. Ugyanígy oldottuk meg az email megváltoztatását is.

```
function save1(e) {  
  e.preventDefault()  
  
  fetch('http://localhost:4000/change/pass', {  
    method: 'PATCH',  
    headers: {  
      'Authorization': token,  
      'Content-type': 'application/json;charset=utf-8'  
    },  
    body: JSON.stringify({  
      "oldpassword": data.oldpassword,  
      "newpassword": data.newpassword,  
      "retype": data.retype  
    })  
  })  
  
  .then((response) => response.json())  
  .then(json => {  
    if(json.message === "Success") {  
      logout()  
    }  
  })  
  .catch(err => console.log(err))  
}
```

A felhasználó kiléptetését a sessionStorageben tárolt token eldobásával és a felhasználó átirányításával oldottuk meg.

```
function logout() {  
  sessionStorage.removeItem('token')  
  props.beallit({token: ""})  
  navigate("/")  
}
```

Számos helyen a partnerek megjelenítése miatt használtuk ezt a POST kérést.

```
function partners_list() {  
  //e.preventDefault()  
  fetch(url3, {  
    method: 'POST',  
    headers: {  
      'Authorization': token,  
      'Content-type': 'application/json;charset=utf-8'  
    }  
  })  
  .then((response) => response.json())  
  .then(json => setPartners(json))  
  .catch(err => console.log(err))  
}
```

A tranzakció felvételes oldalon ellenőrizzük, hogy van-e üres beviteli mező. Ha nincs, akkor felvesszük az adatokat az adatbázisba, tudatjuk a felhasználóval, hogy sikeres volt a felvétel és a beviteli mezőket üresre állítjuk.

```
function addTransaction(e) {  
  e.preventDefault()  
  if(data.date=="" || data.taxid=="" || data.amount=="" || data.partnerid=="" || data.comment=="") {  
    setMessage("Please fill every input field!")  
    return  
  }  
  fetch(url3, {  
    method: 'POST',  
    headers: {  
      'Authorization': token,  
      'Content-type': 'application/json;charset=utf-8'  
    },  
    body: JSON.stringify({  
      "date": data.date,  
      "taxid": data.taxid,  
      "amount": data.amount,  
      "partnerid": data.partnerid,  
      "comment": data.comment  
    })  
  })  
  .then((response) => response.json())  
  .then(setMessage("Transaction added"))  
  .then(cleardata())  
  .catch(err => console.log(err))  
}
```



A partnerek hozzáadását hasonlóan oldottuk meg a tranzakció felvételhez. A fő különbség az, hogy propok segítségével frissítettük a jelenlegi partnerek listáját a Partners komponensnél.

```
function hozzáad(e){
  e.preventDefault()
  if (data.name=="" || data.email=="" || data.postal_code=="" || data.country=="" || data.address=="") {
    setMessage("One of the input fields is empty!")
    return
  }

  fetch('http://localhost:4000/add/partner', {
    method: 'POST',
    headers: {
      'Authorization': token,
      'Content-type': 'application/json;charset=utf-8'
    },
    body: JSON.stringify({
      "name": data.name,
      "email": data.email,
      "country": data.country,
      "postal_code": data.postal_code,
      "address": data.address
    })
  })
  .then((response) => response.json())
  .then(props.updateList)
  .catch(err => console.log(err))
}
```

# Kliens oldal az API-hoz

## Bejelentkezés nélkül

A kliens oldalon egy reszponzív weboldalt készítettünk a Bootstrap 5 és React segítségével. A fő oldal széles monitoron:

**Aquntant**  
The money flows, we help to keep track on it.

**Elegant design**  
Lorem ipsum dolor sit amet consectetur adipiscing elit. Et reiciendis nulla eveniet praesentium, minus rerum suscipit totam necessitatibus at cum adipisci quas est ad veritatis nobis doloribus nemo esse impedit. Ratione est reprehenderit at temporibus fugit mollitia ad nihil id fugiat quasi. Consequatur magnam possimus, exercitationem iure iste hic. Aut, adipisci beatae sit iste tempora dignissimos doloribus vero voluptate quo? Harum nihil officia incidunt cupiditate iusto qui reprehenderit molestiae ipsa! Atque vitae maxime magni perspiciatis, iste perferendis dolorem, ipsa quasi reprehenderit at culpa veniam mollitia voluptates tempora illum exercitationem beatae.

**Easy to use**  
Lorem ipsum, dolor sit amet consectetur adipiscing elit. Non illum nesciunt tenetur quia pariatur. Optio numquam incidunt perspiciatis porro voluptas fuga architecto commodi, eius quo. Nobis excepturi quod sequi iste? Nisi ipsum quo, incidunt, aspernatur eius nam earum deserunt dolor obcaecati rem id nortum ab corrupti, voluptate voluptas nihil blanditiis! Maiores saepe velit fugit praesentium dolor et cum culpa rerum. Reprehenderit laborum quibusdam labore, necessitatibus rerum quae magnam laudantium accusantium architecto esse distinctio ea dolorem eum nulla eius recusandae saepe vel experte enim error asperiores excepturi est! Cupiditate, soluta iusto. Facere et ab aperiam cumque possimus ipsa, placeat voluptatum. Corrupti at qui nulla sit, quidem hic suscipit iusto? Explicabo quod dolorem enim est esse suscipit molestiae quibusdam soluta velit dignissimos.

**Everything in place**  
Lorem ipsum dolor sit amet consectetur, adipiscing elit. Commodi ipsam odit sed perspiciatis. Velit rem fugit quas quos illum, mollitia repudiandae voluptatem odio molestias autem, fugiat eius saepe veritatis nemi! Dolore aliquam, id, consequatur est veritatis quos labore cupiditate fugiat libero possimus quo explicabo. Explicabo excepturi dolorem sunt delectus, a aspernatur? Laborum illo recusandae commodi veniam, expedita utam? Laudantium, placeat! Asperiores, tenetur! Nulla asperiores ipsa soluta aperiam dolore nisi molestiae, dicta nemo nostrum incidunt eae ex expedita inventore, exercitationem ad architecto odit doloremque esse sit adipisci laboriosam! Enim, voluptates voluptatem! Ex saepe cum quasi nulla voluptas ipsum voluptatum, numquam, magni mollitia in accusantium illum similique adipisci cumque voluptatem eos corporis sint molestias quibusdam voluptates nobis explicabo, pariatur laborum! id, ducimus?

For more information, please visit [About](#) page.

Aquntant © 2021 Copyright: Mate Andrea & Borzsonyi Bence

Mobil kijelzőn:

**Aquntant**  
The money flows, we help to keep track on it.

**Elegant design**  
Lorem ipsum dolor sit amet consectetur adipiscing elit. Et reiciendis nulla eveniet praesentium, minus rerum suscipit totam necessitatibus at cum adipisci quas est ad veritatis nobis doloribus nemo esse impedit. Ratione est reprehenderit at temporibus fugit mollitia ad nihil id fugiat quasi. Consequatur magnam possimus, exercitationem iure iste hic. Aut, adipisci beatae sit iste tempora dignissimos doloribus vero voluptate quo? Harum nihil officia incidunt cupiditate iusto qui reprehenderit molestiae ipsa! Atque vitae maxime magni perspiciatis, iste perferendis dolorem, ipsa quasi reprehenderit at culpa veniam mollitia voluptates tempora illum exercitationem beatae.

**Easy to use**  
Lorem ipsum, dolor sit amet consectetur adipiscing elit. Non illum nesciunt tenetur quia pariatur. Optio numquam incidunt perspiciatis porro voluptas fuga architecto commodi, eius quo. Nobis excepturi quod sequi iste? Nisi ipsum quo, incidunt, aspernatur eius nam earum deserunt dolor obcaecati rem id nortum ab corrupti, voluptate voluptas nihil blanditiis! Maiores saepe velit fugit praesentium dolor et cum culpa rerum. Reprehenderit laborum quibusdam labore, necessitatibus rerum quae magnam laudantium accusantium architecto esse distinctio ea dolorem eum nulla eius recusandae saepe vel experte enim error asperiores excepturi est! Cupiditate, soluta iusto. Facere et ab aperiam cumque possimus ipsa, placeat voluptatum. Corrupti at qui nulla sit, quidem hic suscipit iusto? Explicabo quod dolorem enim est esse suscipit molestiae quibusdam soluta velit dignissimos.

For more information, please visit [About](#) page.

Aquntant © 2021 Copyright: Mate Andrea & Borzsonyi Bence

A szöveg és a képek egy oszlopba kerülnek.

Bejelentkezés nélkül az oldal nem jelenít meg az adatbázisból semmit. Viszont így a felhasználónak alkalmja nyílik igénybe venni egy teljes körű leírást az oldal használatáról.

Step no. 1: Please click on the "Sign Up" menu option on the top.

Step no. 2: Fill the form.

- give a name to your account
- use an e-mail address that can be used to communicate with you
- use a secure password
- please retype your password just to confirm there's no typo in it

Step no. 3: Click the "Register" button.

If you can see this page, you can Sign In. So click on the "Sign In" button on the top right.

Registration successful!! Now You can Sign In.

Step no. 1: After registration, if you can see this page, you can Sign In. So click on the "Sign In" button on the top right.

Note: If you already have an account, you don't have to reregister. Without signing in you're always able to find this button at the same place when you visit the site.

Step no. 2: Fill the form.

- use the name you gave to your account
- write the password you added during registration

Step no. 3: Click the "Sign In" button.

If you can see this page, now you're able to do all sort of things that we'll explain on the [After Sign In](#) page in detail.

Date	Amount	Partner	Tax	Comment
2022-12-23	-895078	termyi/igaz		Procurement of goods
2022-12-01	-248696	igaz/hal		Procurement of goods
2022-10-23	640758	gyogyszor/gyogyszor		Sale
2022-10-16	298384	igaz/hal		Sale
2022-10-13	943888	alany adomentes		Sale
2022-10-07	63854	targy adomentes		Sale
2022-09-18	-230090	alany adomentes		Procurement of goods
2022-08-17	-49321	igaz/hal		Procurement of goods
2022-08-03	802472	halaszkoti atlat es		Sale
2022-07-10	-385507	fej r/gabonatermek		Procurement of goods
2022-07-05	624999	halaszkoti atlat es		Sale



Az oldalon lehetőség van felhasználói fiók létrehozására.

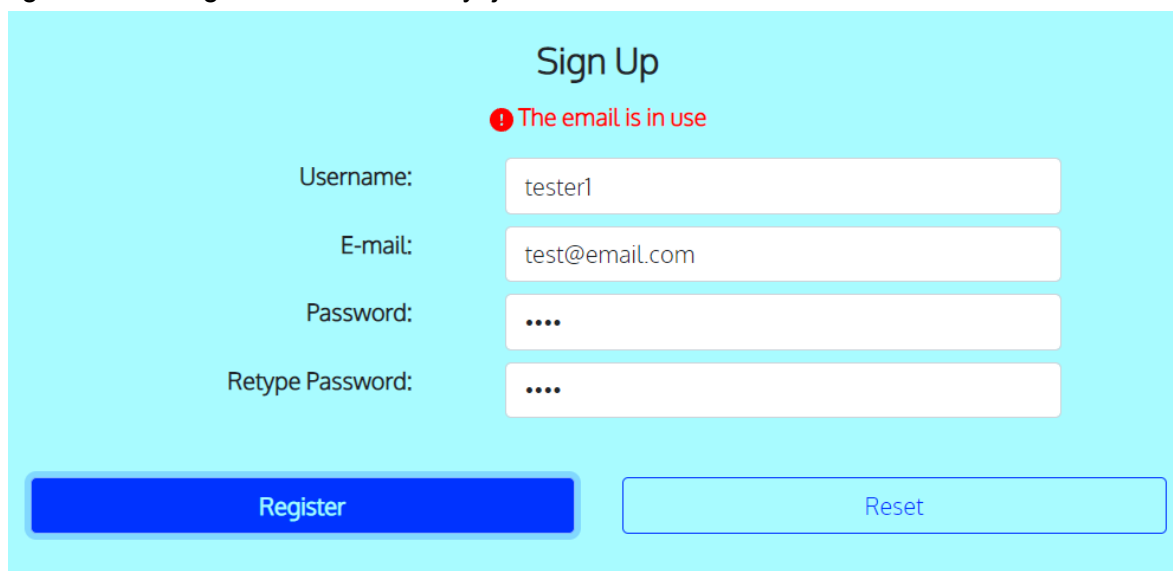


The screenshot shows a 'Sign Up' form on a blue-themed website. The navigation bar at the top has links for 'About', 'Contact', and 'Sign Up'. The form itself is titled 'Sign Up' and contains four input fields: 'Username:', 'E-mail:', 'Password:', and 'Retype Password:'. Below the fields are two buttons: a blue 'Register' button and a light blue 'Reset' button.

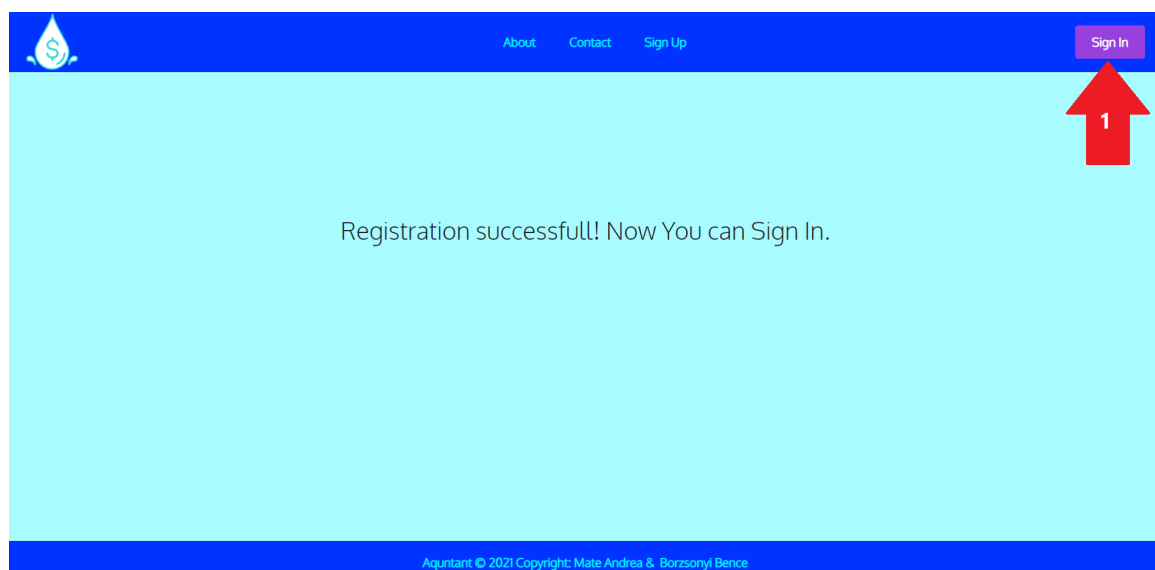
Itt ellenőrizzük:

- üres-e bármelyik mező
- a jelszó ismétlésnél valóban ugyanaz a karaktersorozat szerepel-e
- (a szerver oldalon) a felhasználónév / e-mail használatban van-e

Ezeknek megfelelően üzenetet jelenítünk meg vagy eltároljuk az adatokat és egy sikeres regisztrációt megerősítő oldalra irányítjuk a felhasználót.



This screenshot shows the 'Sign Up' form with an error message: 'The email is in use' in red text with an exclamation mark icon. The input fields are filled with 'tester1' for Username, 'test@email.com' for E-mail, and four dots for both Password and Retype Password. The 'Register' and 'Reset' buttons are still present at the bottom.



The screenshot shows a confirmation message: 'Registration successfull! Now You can Sign In.' in the center of the page. In the top right corner, there is a 'Sign In' button highlighted with a red arrow and the number '1'. The navigation bar at the top includes a logo, 'About', 'Contact', 'Sign Up', and the 'Sign In' button.

```

function register(e) {
  e.preventDefault()
  if (data.username=="" || data.password=="" || data.email=="") {
    setMessage("Username, email or password is empty!")
    return
  }

  if (data.retype !== data.password) {
    e.preventDefault()
    setMessage("The passwords do not match!")
    return
  }
  fetch('http://localhost:4000/register', {
    method: 'POST',
    headers: {
      'Content-type': 'application/json;charset=utf-8'
    },
    body: JSON.stringify({
      "username": data.username,
      "password": data.password,
      "email": data.email,
      "retype": data.retype
    })
  })
  .then((response) => response.json())
  .then(json => {
    if(json.message == "Success") {
      navigate("/successfull-reg")
    } else {
      setMessage(json.message)
    }
  })
  .catch(err => console.log(err))
}

```

Bejelentkezni is lehetőség van egy form kitöltésével.

Itt ellenőrizzük:

- (szerver oldalon) van-e a megadott felhasználónévvel felhasználó az adatbázisban, ha van jó-e a jelszó
- üres-e bármelyik mező

Ezeknek megfelelően vagy üzenetet jelenítünk meg vagy átirányítjuk a felhasználót.

Sign In

! Username or password is empty!

Username:

Password:

Sign In Register

## Bejelentkezve

Bejelentkezést követően kerülünk egy olyan oldalra, ahol az API-tól lekért tranzakciók jelennek meg tablázatos formában. Illetve a szűrő funkció is az API-ból kérdezi le a felhasználó partnereinek a nevét.

Itt lehetősége van a felhasználónak szűréseket végezni a már rögzített tranzakcióin.

Megjelenítheti csak a bevételeit vagy csak a kiadásait, adott hónapra és adott partnerre szűrhet, illetve akár mindhármát is megteheti egyszerre.

A lekérdezéshez a fetch promise-t használjuk. Ha sikerül a kérés, akkor JSON formátumra alakítjuk, majd az adatokat useState segítségével eltároljuk és map segítségével helyezzük el az adatokat a megfelelő tagek közé. Ezenkívül megjelenítjük a felhasználó összes bevételet, kiadását és az egyenlegét.

2022-10-07	65854	Tester Toby	targyi adomentes	Sale
2022-09-18	-230090	Caterpillar	alanyi adomentes	Procurement of goods
2022-08-17	-61321	JetBlue Airways	tojas/hal	Procurement of goods
2022-08-03	802472	Citigroup	haziasított állat es	Sale
2022-07-10	-385507	JetBlue Airways	tej-/gabonatermek	Procurement of goods
2022-07-03	624999	Hormel Foods	haziasított állat es	Sale
2022-06-22	64760	Hormel Foods	gyogyszer/gyogyszer	Sale
2022-05-09	-965607	Macy's	konyv/ujsgag	Procurement of goods
2022-04-28	495080	Tester Toby	tojas/hal	Sale

```
function readIn() {  
  fetch(url, {  
    method: 'GET',  
    headers: {  
      'Authorization': token  
    }  
  })  
  .then((response) => response.json())  
  .then((json) => setInformation(json))  
  .catch(err => console.log(err))  
}
```

```

<table className='table table-striped table-hover m-2'>
  <thead>
    <tr>
      <th>Date</th>
      <th>Amount</th>
      <th>Partner</th>
      <th>Tax</th>
      <th>Comment</th>
    </tr>
  </thead>
  <tbody>
    {information && information.length>0 && information.map(
      (item)=>
        <tr>
          <td>{item.date}</td>
          <td>{item.amount}</td>
          <td>{item.name}</td>
          <td>{item.tax}</td>
          <td>{item.comment}</td>
        </tr>)}</tbody>
  </table>

```

```

const [partners, setPartners] = useState({
  partner: ""
})

useEffect(() => {
  partners_list();
}, []);

function partners_list() {
  //e.preventDefault()
  fetch(url3, {
    method: 'POST',
    headers: {
      'Authorization': token,
      'Content-type': 'application/json;charset=utf-8'
    }
  })
  .then((response) => response.json())
  .then(json => setPartners(json))
  .catch(err => console.log(err))
}

```



```


<div className='col-lg-4'>
  <label htmlFor='partner'>Partner</label>
  <select className='form-control p-1' id='partner'
    name='partner' value={props.data.partner} onChange={props.
    change}>
    <option value='{0}' selected>-- Select --</option>
    {partners && partners.length>0 && partners.map(
      (item)=>
        <option value={item.name}>{item.name}</option>
      )}
  </select>
</div>

```

A "New Transaction" fülön lehetőség van új pénzmozgás rögzítésére. Itt csak azt ellenőrizzük, hogy üres-e bármelyik mező. Ha üres, üzenetet jelenítünk meg, ha nem eltároljuk az adatokat, illetve erről is értesítjük a felhasználót.

Az adó kiválasztásához az API-ból kérjük le, hogy miből lehet választani. Ezenkívül legördülő menüvel biztosítjuk, hogy csak olyan partnerhez köthető mozgást rögzítsen a felhasználó, akinek az adatai már szerepelnek az adatbázisban. Ezt is az API segítségével töltjük fel adatokkal.

A "Partners" fülön adunk alkalmat a felhasználónak, hogy új partnert adjon a meglévőekhez és egy táblázatban megtekintheti a már az adatbázisban szereplő partnereit. Itt is csak azt ellenőrizzük, hogy minden mező kitöltésre került-e, mivel engedjük, hogy ugyanazt a partnert több felhasználó is megadja és feltételezzük, hogy a felhasználó látja, hogy ha már egy partnert hozzáadott. A táblázat feltöltésére itt is az API-t használjuk.


New Transaction
Partners
Settings

Partner data:

Name

E-mail

Country


Postal code

City, street and house number

+

Name	E-mail	Address
Tester Toby	test@email.com	testnationl 420 Test city,test str. 21
Macy's	info@macys.com	US, Florida 33166 Miami Springs, 2549 Arbutus Drive
Citigroup	info@citigroup.com	US, Virginia 22304 Alexandria, 2983 Perine Street
JetBlue Airways	info@jetblue-airways	US, Georgia 30097 Duluth, 168 Junior Avenue
Caterpillar	info@caterpillar.com	US, Washington DC 30097 Washington, 3912 Hickory Lane
Hormel Foods	info@hormel-foods.co	US, Texas 77036 Houston, 1706 Mulberry Street

A "Settings" gomb rejti a felhasználói beállításokat, ahol lehetőséget adunk a jelszó-, és az e-mail módosítására. Be kérjük a régi jelszót, ellenőrizzük, hogy helyes-e. Ezután kétszer kérjük az új jelszót, ezeknek az egyezését is vizsgáljuk. E-mail változtatásánál kérjük a régi címet, ellenőrizzük, hogy helyes-e. Kérjük az új címet, és az aktuális jelszót, ennek is ellenőrizzük a helyességét. Az oldal alján található a kijelentkezésre szolgáló gomb. Ezzel dobjuk el a tokent és irányítjuk át a felhasználót a kezdőoldalra.


New Transaction
Partners
Settings

### Password modification

Old Password:

New password:

Retype New Password:

Save

Reset

### E-mail modification

Old E-mail:

New E-mail:

Password:

Save

Reset

Log out

A navigációhoz a react-router-dom csomagot használtuk, illetve a styled-components csomagot annak formázására.

Az App.jsx komponensben adjuk meg a path propertyben, hogy milyen útvonal esetén, és az element propertyben, hogy mit jelenítsen meg az oldal. A Navbar komponensben a to propertyben adjuk meg, hogy milyen útvonalra irányítsa a böngészőt a linkre való kattintáskor.

```
import styled from "styled-components";
import { NavLink as Link } from "react-router-dom";

const Nav = styled.nav`
  background: #0033FF;
  height: 80px;
  display: flex;
  justify-content: space-between;
  padding: 0.5rem calc((100vw-1000px)/2);
  z-index: 10;
`;

const NavLink = styled(Link)`
  color: #00F2FF;
  display: flex;
  align-items: center;
  text-decoration: none;
  padding: 0 1rem;
  height: 100%;
  cursor: pointer;
  padding: 10px 22px;

  &:hover{
    color: #fff;
  }

  &.active{
    color: #0033FF;
    background: #00F2FF;
    border-radius: 50% 50% 0 0;
    padding: 10px 22px;
    border: none;
    outline: none;
  }
`;
```

```

const App = () => {
  const [userData, setUserData] = useState({
    token: ""
  });

  return (
    <Router>
      <Header />
      <Routes>
        <Route path='/transactions' element={<Transactions/>} />
        <Route path='/new' element={<New/>} />
        <Route path='/partners' element={<Partners/>} />
        <Route path='/settings' element={<UserSettings beallit={setUserData}/>} />
        <Route path="/" exact element={<Home/>} />
        <Route path='/contact' element={<Contact/>} />
        <Route path='/sign-up' element={<SignUp/>} />
        <Route path='/sign-in' element={<SignIn beallit={setUserData}/>} />
        <Route path='/successful-reg' element={<SuccessfulReg/>} />
        <Route path='/about' exact element={<About/>}/>
        <Route path='/about/singup' element={<SinUp/>}/>
        <Route path='/about/singin' element={<SinIn/>}/>
        <Route path='/about/aftersignin' element={<AfterSignIn/>}/>
      </Routes>
      <Footer/>
    </Router>
  )
}

export default App

```

```

const NavbarLogged = () => {
  return (
    <Nav >
      <NavLink to='/transactions' activeStyle >
        <img src={icon} alt='Aquntant' height={80} />
      </NavLink>
      <NavMenu >
        <NavLink to='/new' activeStyle>New Transaction { /*page to add a new
          trasaction */}
        </NavLink>
        <NavLink to='/partners' activeStyle>Partners { /*page to show the
          list of partners the user has*/}
        </NavLink>
      </NavMenu>
      <NavBtn >
        <NavBtnLink to='/settings' >
          Settings { /*page for modifying email, pass etc.*/}
        </NavBtnLink>
      </NavBtn>
    </Nav>
  )
}

export default NavbarLogged

```

Használtunk konstansokat a stílushoz és a színekhez, és innen importáltuk a stílusokat és a színeket.

```
import colors from './colors'

const style = {
  content: {
    background: colors.vilagoskek,
    minHeight: '100vh',
    marginBottom: 55
  },
  btnPrim: {
    background: colors.sotettek,
    color: colors.vilagoskek,
    fontWeight: 'bolder'
  },
  btnSec: {
    color: colors.sotettek,
    border: 'solid 1px ' + colors.sotettek
  },
  message: {
    color: colors.piros,
    fontWeight: "bolder",
    textAlign: 'center'
  },
  strip_form: {
    border: 'solid 1px ' + colors.sotettek,
    borderRadius: 5,
    background: colors.feher
  },
  btnPlus: {
    background: colors.sotettek,
    color: colors.vilagoskek,
    border: 'solid 1px ' + colors.sotettek,
    borderRadius: '50%',
    width: 60,
    height: 60,
    fontWeight: 'bolder',
    fontSize: 30
  }
}

export {style}
```

```
const colors={
  sotettek: "#0033FF",
  vilagoskek: '#A8FBFF',
  lila: '#9740DE',
  piros: '#FF0000',
  fehér: '#fff'
}

export default colors
```

Minden form működését manuálisan teszteltük. A megjelenítés teszteléséhez azonban közvetlenül az adatbázisba importáltunk adatokat.

# Továbbfejlesztés

Szerintünk ez a weboldal egy remek alap egy könyvelő alkalmazás létrehozásához. Ehhez bővíteni kell az adó táblát és utána nézni a főkönyv működésének.

Továbbá lehetne megrendelések leadására alkalmas oldallal bővíteni, ahol rögzíteni lehetne, hogy mikorra kell, mit, milyen értékben és mennyiségbe szállítani kinek. Ezenkívül ugyanilyen paraméterekkel beszállítóknak megrendeléseket rögzíteni.

Hogy kisebb módosításokra is kitérjek, érdemes lehet a legördülő menüknél engedni a gépelést és a begépett adat alapján szűrni a listában megjelenő adatokat. Ez sok adónem és partner tárolásakor felhasználó barátabbá tenné használatot.

Jelenleg, ha egy felhasználó elfelejti a jelszavát, kénytelen nekünk e-mailt írni, ekkor nekünk generálni kell egy nehezen kitalálható jelszót erre módosítani kézzel az adatbázisban a jelszót és elküldeni a felhasználónak. Ezt a folyamatot lehetne automatizálni.

Diagramok használatával is látványosabbá lehet tenni az adatok megjelenítését.

## Források

Navbar: [https://www.youtube.com/watch?v=VzWBLj\\_CfpE](https://www.youtube.com/watch?v=VzWBLj_CfpE)

<https://stackoverflow.com/>

<https://getbootstrap.com/>