

VAS VÁRMEGYEI SZAKKÉPZÉSI CENTRUM
HORVÁTH BOLDIZSÁR
KÖZGAZDASÁGI ÉS INFORMATIKAI
TECHNIKUM

A 5 0613 12 03 számú Szoftverfejlesztő és –tesztelő vizsgaremek

Daily planner

Készítették:

KERVÁRICS NORBERT

KENDIK ÁRMIN

SZOMBATHELY

2024

Tartalom

Bevezetés	2
A szoftver célja.....	2
Használati leírás	2
Backend	3
Frontend	5
Adatbázis	7
users:	7
plans:	7
Er-modell:	7
Relációs adatmodell:	8
Tesztek	9
Jövőbeli tervek	11

Bevezetés

Az elején több ötlet is felmerült, hogy milyen weboldalt kéne csinálni. Volt szó webshopról, játékról stb. Végül egy napi tervezőn egyeztünk meg, ugyanis mindketten elég szétszórtak vagyunk, és nincs szigorú napirendünk, ami miatt elég nagy szervezetlenség alakul ki.

A „Daily Planner” oldalon bejelentkezés után teendőidet eltároljuk és töröljük, ha azt szeretnéd.

A szoftver célja

Szoftverünk célja, hogy terveidet, teendőidet könnyen és gyorsan nyomon tudd követni, ezzel megkönnyebbítve mindennapjaidat.

Használati leírás

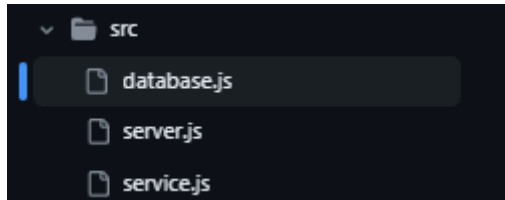
Bejelentkezés után jobb felül a menüpontok segítségével tudsz lépegetni a fülek között. Ha a „Planning” pontra kattintunk, akkor az átdob a tervező képernyőre, ahol megtervezheted, melyik napra milyen teendőt kell majd elintézned, amit mi eltárolunk az adatbázisba, majd a „My Plans” menüpontra kilistázzuk neked.

Backend

A backend fájlok a „src” (három „js” fájl) mappában találhatók.

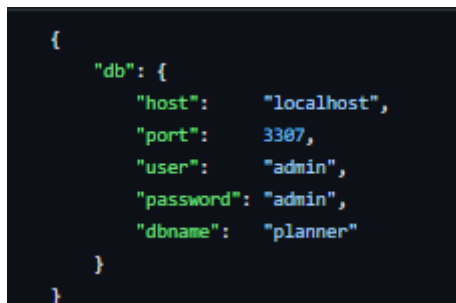
A backendhez 4.18.2-es „express” -t és 3.9.1 -es „mysql2” -öt használtunk (package.json).

Az adatbázissal való kommunikációt a „database.js” hozza létre:

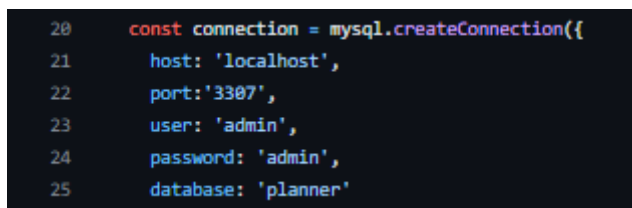


Az adatbázishoz kettő API -val csatlakozik, mert volt, amit nem tudtunk az egyikkel megoldani, ezért kellett egy másik fajta.

Első API-hoz MySQL kapcsolat kiépítése „dblogin.json” -ben található:



Második API-hoz MySQL kapcsolat kiépítése „server.js” -ben található:



Mind a kettő REST API kérés a „server.js” -ben található:

```
30  /*REST API POST type 1*/
31  app.post('/api', async (request, response) => {
32    if (request.body.action && request.body.action === 'login') {
33      const name = await Service.login(request.body.uid, request.body.password)
34
35      response.send(JSON.stringify({ name: name }))
36    }
37    /*Tervbetöltés API-ja*/
38    else if (request.body.action && request.body.action === 'loadplans') {
39      const plans = await Service.loadPlans()
40      response.send(JSON.stringify(plans))
41    }
42    /*Terv törlésének API-ja*/
43    else if (request.body.action && request.body.action === 'deleteplan') {
44      await Service.deletePlan(request.body.plan)
45      response.send(JSON.stringify(true))
46    }
47  })
48
49
50
51  /*REST API POST type 2*/
52  app.post('/api/insert', (req, res) => {
53    const data = req.body;
54    console.log(data.uid)
55    const sql = `INSERT INTO plans(PLANNAME, DATE, DESCRIPTION, UID) VALUES ("${data.name}", "${data.date}", "${data.description}", "${data.uid}")`;
56    connection.query(sql, [data.name, data.date, data.description, data.uid], (err, result) => {
57      if (err) {
58        console.error(err);
59        res.status(500).json({ error: 'Error inserting data' });
60      } else {
61        res.json({ success: true, message: 'Data inserted successfully' });
62      }
63    });
64  });
```

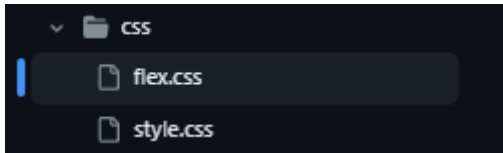
A szerver a 9000-es porton fut (szintén a „server.js” -ben található):

```
66  /* PORT */
67  const PORT = 9000
68  console.log(`WEB -es kiszolgáló indítása a ${PORT} -porton`)
69
70  app.listen(PORT)
```

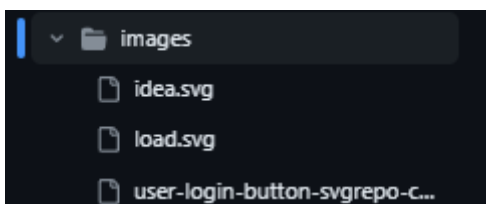
Frontend

A frontend fájlok a „www” („css”, „images”, „js” mappa és négy „html” fájl) mappában találhatók.

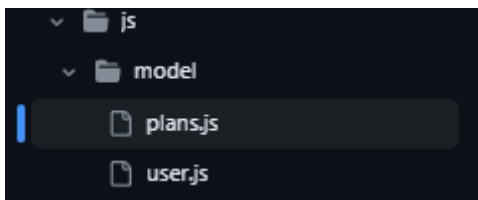
Az oldal kinézetéért felelős fájlok a „css” (kettő „css” fájl) mappában találhatók:



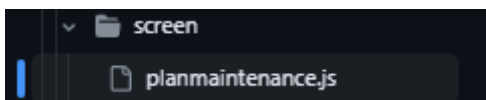
Az oldalhoz használt képek az „images” (három képfájl) mappában találhatók:



A „js” („model”, „screen” mappa és három „js” fájl) mappán belül a „model” (kettő „js” fájl) mappában találhatók a „plan.js” és „user.js” fájlok, ezen fájlok egy-egy „class” -t tartalmaznak, amik visszaadják a tervek, felhasználók adatait, illetve „JSON” objektummá és „string” -é konvertálja azokat.



A „screen” (egy „js” fájl) mappában a „planmaintenance.js” található, ami a tervek táblájának a kirajzolásáért, illetve a tervek törléséért felel.



Az „app.js” -ben vannak azok a függvények, amik a fókuszot kezelik, illetve a bejelentkezést vizsgálják.

Az „index.js” -ben olyan függvények vannak, amik helyet csinálnak a menünek telefonon, a kijelentkezést végzik el, beállítják az időt és terveket mentik.

A „service.js” -ben a tervek betöltéséért, törléséért és a bejelentkezésért felelős függvények vannak.

Az „index.html” a bejelentkező oldal, ami sikeres bejelentkezés esetén a „main.html” oldalra dob át, ami afféle főoldalként üzemel, ahol szlogenek találhatók.

A menü használatával tudsz oda-vissza lépkedni a „main.html”, „myplans.html” és a „planner.html” között.

A „planner.html” -en tudod megtervezni, hogy melyik napra milyen feladatot akarsz elmenteni, és ha meg akarod tekinteni terveid, a „myplans.html” -en kilistázva megtalálod őket.

Adatbázis

Az adatbázis fájlok a „sql” (kettő „sql” fájl) mappában találhatók.

Az adatbázis neve „planner” (felépítése schema.sql-ben található), ami kettő táblából áll („users”, „plans”):

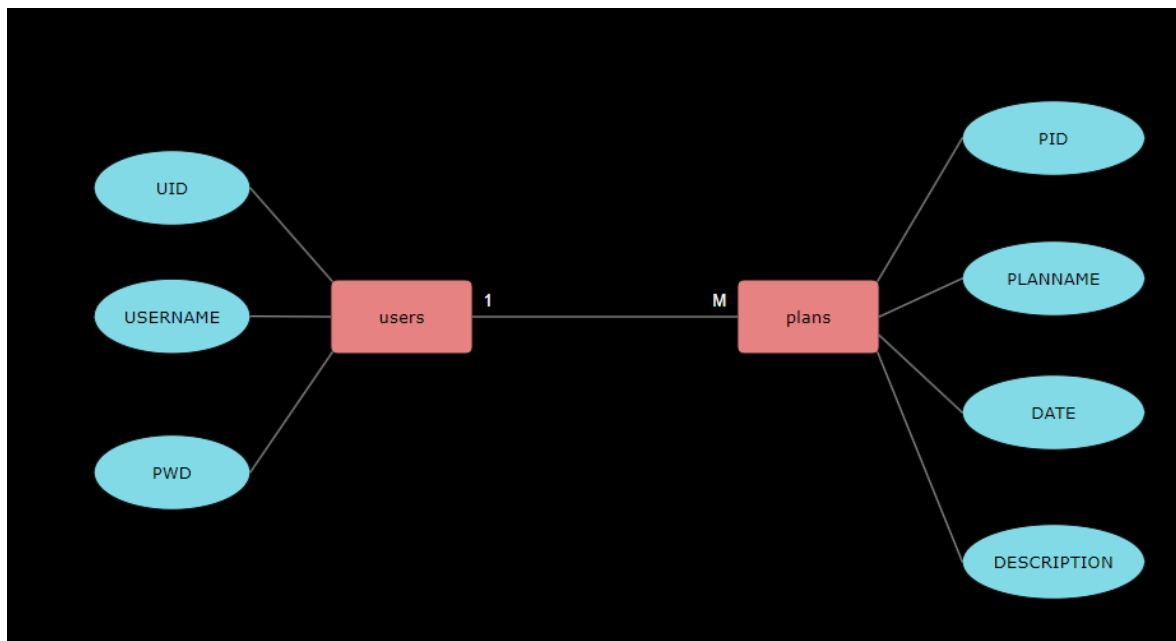
users:

```
DROP TABLE IF EXISTS `users`;  
CREATE TABLE `users` (  
  `UID` varchar(40) NOT NULL,  
  `USERNAME` varchar(100) NOT NULL,  
  `PWD` varchar(100) NOT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_hungarian_ci;
```

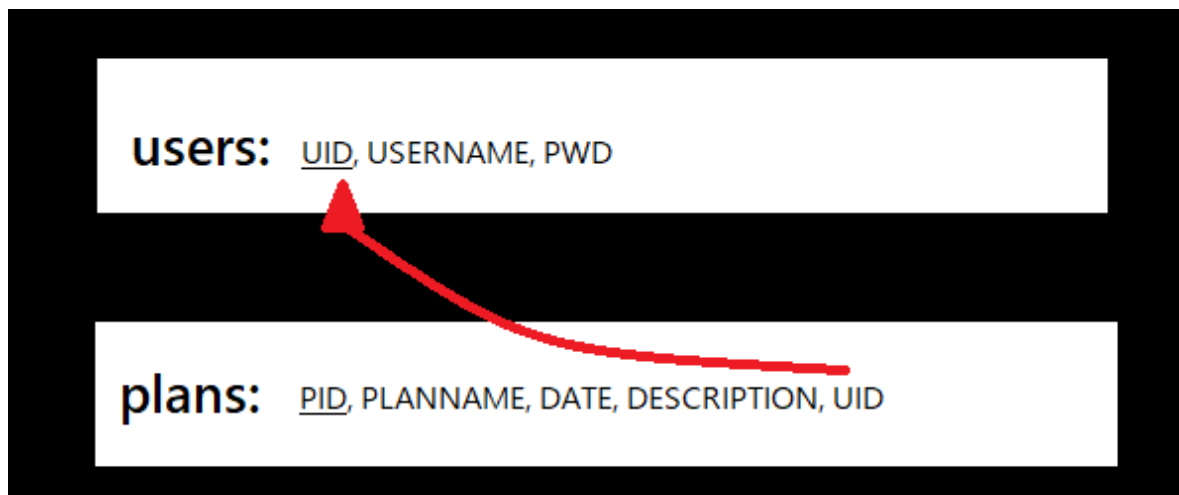
plans:

```
DROP TABLE IF EXISTS `plans`;  
CREATE TABLE `plans` (  
  `PID` int(10) NOT NULL,  
  `PLANNAME` varchar(100) NOT NULL,  
  `DATE` date NOT NULL,  
  `DESCRIPTION` varchar(200) DEFAULT NULL,  
  `UID` varchar(40) NOT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_hungarian_ci;
```

Er-modell:



Relációs adatmodell:



Tesztek

A tesztek a test mappán belül, a „test.http” fájlban lehet megtalálni.

Ebben a http kérésben 2 dolgot ellenőrzünk:

1.Siker-e a bejelentkezés:

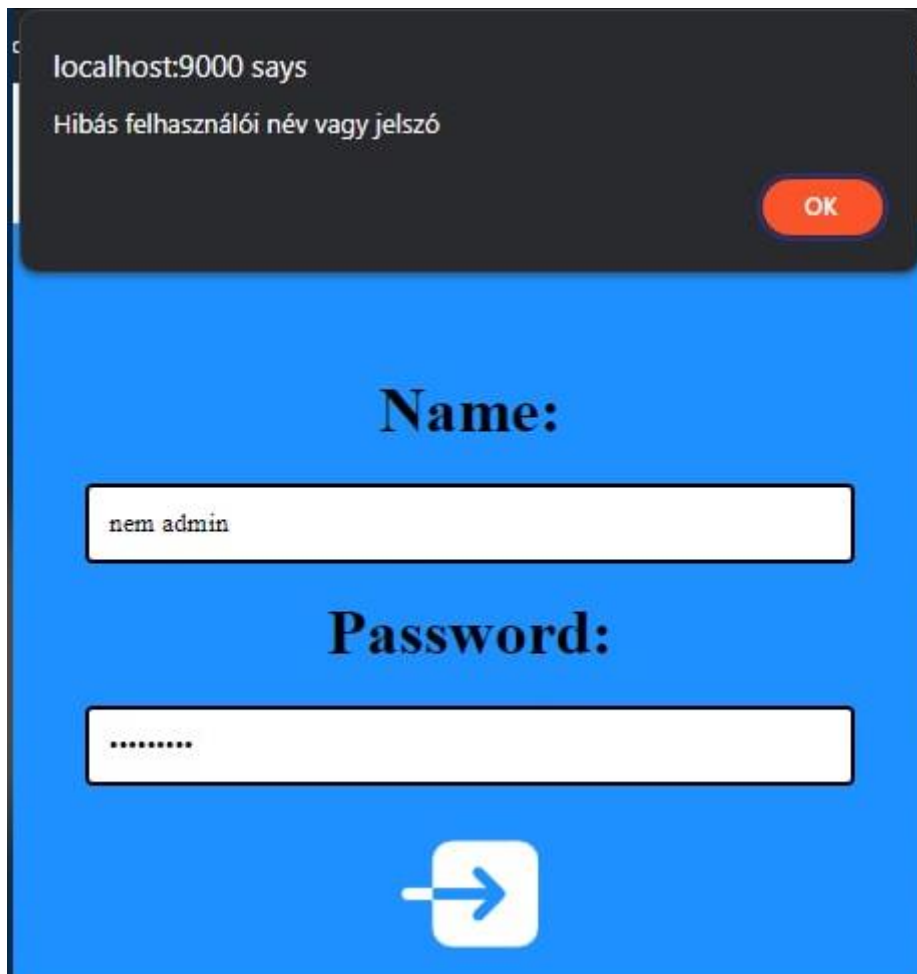
```
### Login ellenőrzése. Sikeres.  
POST http://localhost:9000/api HTTP/1.1  
content-type: application/json  
  
{ "action": "login", "uid": "admin", "password": "admin" }
```

2.Betölti-e a terveket:

```
### Tervek betöltése. Sikeres.  
POST http://localhost:9000/api HTTP/1.1  
content-type: application/json  
  
{ "action": "loadplans" }
```

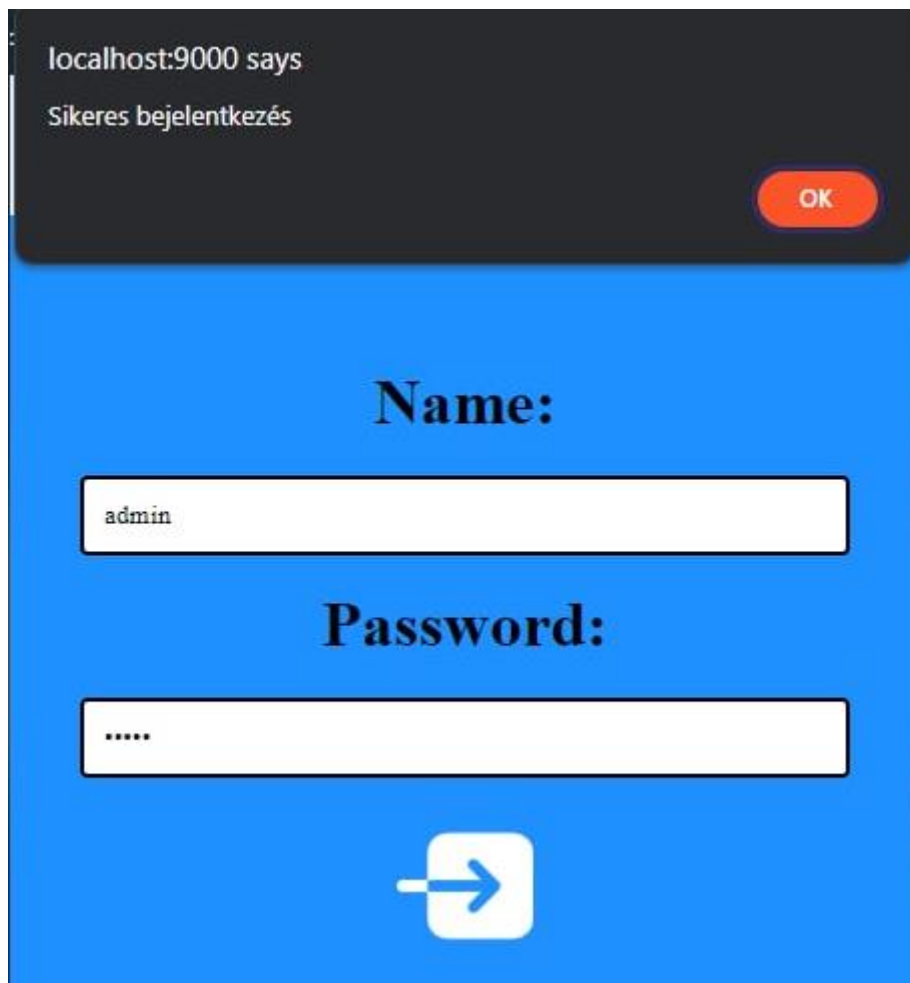
Ezen kívül az oldalon még felugró ablak jelenik meg, ha elrontod a bejelentkezést és akkor is, ha sikeresen bejelentkezel.

Sikertelen bejelentkezés:



The screenshot shows a web application interface with a blue background. At the top, a dark grey error message box is displayed, containing the text "localhost:9000 says" and "Hibás felhasználói név vagy jelszó" (Incorrect username or password), with an "OK" button. Below the error box, the "Name:" label is followed by a text input field containing the text "nem admin". The "Password:" label is followed by a password input field filled with dots. At the bottom, there is a white button with a blue right-pointing arrow.

Sikeres bejelentkezés:



The image shows a web application interface. At the top, a dark grey notification box contains the text "localhost:9000 says" and "Sikeres bejelentkezés" (Successful login) with an "OK" button. Below this, the main area has a blue background. It features a "Name:" label followed by a text input field containing "admin". Below that is a "Password:" label followed by a password input field with masked characters "*****". At the bottom of the blue area is a white button with a blue right-pointing arrow.

Az oldalon egészen addig nem tudod használni a menüpontokat, amíg sikeresen bejelentkezel.

Jövőbeli tervek

- Telefonos applikációt csinálni hozzá.
- Telefonos applikációban üzenetet küldeni, ha aznapra vagy másnapra van valami terved, amit nem csináltál még meg.
- Lehessen rendezni, illetve módosítani a terveid.