ELTE címer

Eötvös Loránd Tudományegyetem

Informatikai Kar

Média- és Oktatásinformatika Tanszék

Webes alkalmazás társasházak közüzemi fogyasztásainak adminisztrálására

Dr. Horváth Győző

egyetemi adjunktus

Soós Bálint

programtervező informatikus Bsc

Budapest, 2018

Tartalomjegyzék

[Bevezetés 3](#_Toc499592040)

[Motiváció 3](#_Toc499592041)

[Felhasználási igények 3](#_Toc499592042)

[Felhasználói dokumentáció 4](#_Toc499592043)

[Követelmények 4](#_Toc499592044)

[Támogatott böngészők: 4](#_Toc499592045)

[Alkalmazás használata 4](#_Toc499592046)

[Alkalmazás használata közös képviselők számára 4](#_Toc499592047)

[Alkalmazás használata lakók számára 6](#_Toc499592048)

[Fejlesztői dokumentáció 8](#_Toc499592049)

[Feladat elemzése 8](#_Toc499592050)

[Felhasználói esetek 8](#_Toc499592051)

[Tervezés 8](#_Toc499592052)

[Szerver oldali architektúra 8](#_Toc499592053)

[RESTful API 8](#_Toc499592054)

[Adatbázis 11](#_Toc499592055)

[Modellek 11](#_Toc499592056)

[Kliens oldali architektúra 12](#_Toc499592057)

[Felhasznált technológiák 12](#_Toc499592058)

[Komponens hierarchia 12](#_Toc499592059)

[Kommunikáció 12](#_Toc499592060)

[Authentikáció 12](#_Toc499592061)

[Felhasznált technológiák 12](#_Toc499592062)

[Értesítési rendszer 12](#_Toc499592063)

[Felhasznált technológiák 12](#_Toc499592064)

[DVD melléklet tartalma 12](#_Toc499592065)

[Fejlesztőkörnyezet 13](#_Toc499592066)

[Rendszerkövetelmények 13](#_Toc499592067)

[A fejlesztőkörnyezet kialakítása 13](#_Toc499592068)

[Fejlesztési munkafolyamat 14](#_Toc499592069)

[Verziókövetés 14](#_Toc499592070)

[Continuous delivery 14](#_Toc499592071)

[Tesztelés 15](#_Toc499592072)

[Fejlesztési lehetőségek 15](#_Toc499592073)

# Bevezetés

## Motiváció

Újépítésű társasházakban egyre jellemzőbb a központi kazán, amellyel a lakások fűtését és melegvízellátását biztosítják. Ilyen rendszer mellett a lakók fogyasztásainak adminisztrálása a közös képviselő feladata lesz, azonban ennek kivitelezése a lakóknak és a közös képviselőnek is komoly odafigyelést és terhet jelent.

Saját helyzetemből kiindulva, az én társasházamban - mely 96 lakásból áll - minden hónap végén egy papírlapot függesztenek ki a lépcsőházban, amelyre minden lakónak kézzel kell beírni a hőmennyiségmérő és a vízórák állását. Ezt a papírlapot a bejelentési időszak végeztével a közös képviselő elviszi és kézzel viszi fel az adatokat egy Excel táblába, ahol kiszámolja a havi fűtés és vízmelegítés díját. A számlák kitöltése szintén kézzel történik, amiket ezután a postaládába dobva kézbesítenek a lakóknak. Ez a rendszer lassú, és a legtöbb lépésben ott van az emberi hibalehetőség is. Ennek a rendszernek a kiváltására szeretnék tervezni egy webes alkalmazást, amely a lépések automatizálásával kiküszöbölheti az emberi tényezőt és kényelmes felületet nyújthat a társasház lakóinak és a közös képviselőnek.

## Felhasználási igények

Az alkalmazásnak két fő felhasználási igényt kell kielégíteni. A lakók számára egy könnyen átlátható felületet kell biztosítani, ahol felölthetik a fogyasztásmérők állását, illetve megtekinthetik a kapott számlákat. A közös képviselőnek egy adminisztrációs felületet kell biztosítani, ahol láthatja a fogyasztási adatokat, és ezek alapján számlákat tud kiállítani, majd ezeket elküldeni a lakóknak.

# Felhasználói dokumentáció

## Követelmények

A webes alkalmazás használatához szükség van egy JavaScript futtatására alkalmas webböngészőre, internet hozzáférésre és email címre. Az alkalmazás csak angol nyelven érhető el, így szükség van minimális angol nyelvtudásra is.

### Támogatott böngészők:

* Google Chrome (v62.x és későbbi verziók)
* Google Chrome for Android (v62.x és későbbi verziók)

## Alkalmazás használata

Az alkalmazás elérhető a következő címen: https://app-rezsi.herokuapp.com/

### Alkalmazás használata közös képviselők számára

#### Regisztráció

Az alkalmazás használatához rendelkeznünk kell egy regisztrált fiókkal, amit a Sign Up (Regisztrálás) aloldalon hozhatunk létre. Meg kell adnunk az email címünket, felhasználó nevünket és a jelszónkat. Sikeres regisztráció esetén a megadott email címre egy visszaigazoló email fog érkezni. Az emailben található gombra kattintva erősítsük meg, hogy valódi címet adtunk meg.

Sign Up screenshot

#### Bejelentkezés

Az alkalmazásba való belépéshez navigáljunk a Login (Bejelentkezés) aloldalra és írjuk be a regisztrációkor megadott email címet és jelszót.

Login screenshot

#### Kijelentkezés

Az alkalmazás jobb felső sarkában található menü ikonra kattintva a felugró menüben láthatjuk az aktuálisan bejelentkezett felhasználó nevét és email címét. A Sign Out gombra kattintva kijelentkezhetünk az alkalmazásból.

Logout screenshot

#### Új csoport létrehozása

Jövőbeli felhasználóinkat (lakóinkat) saját preferenciáink szerint csoportokba oszthatjuk. A csoportosítás módja a közös képviselőre van bízva, ez lehet például épület, lépcsőház, vagy akár szint alapján. Egy felhasználó csak egy darab csoport tagja lehet. Új csoport létrehozásához a csoport nevét kell megadni.

Groups screenshot

Create group screenshot

#### Csoport szerkesztése és törlése

A már létező csoportok nevét átírhatjuk, vagy akár törölhetjük az egész csoportot a hozzá tartozó felhasználókkal együtt. Ebben az esetben a törölt felhasználók többé nem lesznek képesek bejelentkezni az alkalmazásba.

Group options screenshot

Edit group screenshot

Delete group screenshot

#### Lakók meghívása a csoportba

Új lakók meghívásához kattintsunk az adott csoport lakóinak listájánál az Invite gombra. A megjelenő párbeszédablakban található linken keresztül tudnak regisztrálni a lakók a csoportunkba. Ez a link teljesen publikus, így fokozottan figyeljünk oda, hogy kinek küldjük el.

Invite dialog screenshot

#### Lakó törlése

A csoportba felvett lakókat egyesével törölhetjük. A törölt lakók többé már nem lesznek képesek bejelentkenzni az alkalmazásba. Újrafelvételükhöz újra regisztrálni kell.

#### Csoportos számla kiállítása

A csoport összes tagja számára egyszerre tudunk kiállítani számlát. A csoport számláinak listájánál kattintsunk a Create bill gombra. A felugró párbeszédablakban meg kell adnunk a számlázási időszak kezdetének dátumát, a számlázási időszak végének dátumát, a pénznemet, a fűtés egységárát pénznem/kWh-ban, a melegvíz egységárát pénznem/köbméterben, és a hidegvíz egységárát pénznem/köbméterben. A számla készítéséről minden érintett felhasználó értesítést kap.

#### Csoportos számla adatainak letöltése

A kiállított csoportos számla adatait letölthetjük CSV formátumban. Ehhez a csoport számláinak listájában a számla alatti Download as CSV gombra kell kattintani. A fájl egy sora tartalmazza a csoporthoz tartozó lakó nevét, email címét, az adott időszakban felhasznált fűtés mennyiségét kWh-ban, a fűtés egységárát pénznem/kWh-ban, fűtés összdíját az adott pénznemben, melegvíz fogyasztást köbméterben, melegvíz egységárát pénznem/köbméterben, melegvíz összdíját, hidegvíz fogyasztást köbméterben, hidegvíz egységárát pénznem/köbméterben, hidegvíz összdíját, a megadott pénznemet és a végösszeget.

#### Lakó fogyasztási adatainak és számláinak megtekintése

A csoport lakóinak listájánál ha rákattintunk egy adott lakóra, akkor megtekinthetjük az adott lakó összes fogyasztási bejelentését és az összes kiállított számláját. A bejelentések tartalmazzák a bejelentés napjának dátumát, és a fogyasztásmérők állását: a hőmennyiséget kWh-ban és a meleg-, illetve hidegvíz mennyiségét köbméterben. A számlák tartalmazzák a kiállítás dátumát, a számlázási időszak kezdetének és végének dátumát, illetve egy táblázatban lebontva a fogyasztások díjait a végösszeggel együtt.

#### Lakó számlájának letöltése

A lakók kiállított számláit szabadon letölthetjük PDF formátumban. Ehhez kattintsunk a lakó számláinak listájában a számla alatt található Download as PDF gombra. A letöltött fájl tartalmazza a számla azonosítószámát, kiállításának dátumát, a számlázási időszak kezdetének és végének dátumát, illetve egy táblázatban lebontva a fogyasztások díjait a végösszeggel együtt.

### Alkalmazás használata lakók számára

#### Regisztráció

Az alkalmazás használatához rendelkeznünk kell egy regisztrált fiókkal, amit a közös képviselőtől kapott linken keresztül tehetünk meg. Meg kell adnunk az email címünket, felhasználó nevünket és a jelszónkat. Ajánlott a valódi nevünket használni, hiszen a közös képviselő így tud a legegyszerűbben beazonosítani minket. Sikeres regisztráció esetén a megadott email címre egy visszaigazoló email fog érkezni. Az emailben található gombra kattintva erősítsük meg, hogy valódi címet adtunk meg.

Sign Up screenshot

#### Bejelentkezés

Az alkalmazásba való belépéshez navigáljunk a Login (Bejelentkezés) aloldalra és írjuk be a regisztrációkor megadott email címet és jelszót.

Login screenshot

#### Kijelentkezés

Az alkalmazás jobb felső sarkában található menü ikonra kattintva a felugró menüben láthatjuk az aktuálisan bejelentkezett felhasználó nevét és email címét. A Sign Out gombra kattintva kijelentkezhetünk az alkalmazásból.

Logout screenshot

#### Új bejelentés készítése

Új bejelentés készítéséhez kattintsunk a bejelentések listájánál a Create report gombra. A felugró párbeszédablakban adjuk meg a hőmennyiségmérő állását kWh-ban és a meleg-, illetve hidegvíz mérőórájának állását köbméterben.

#### Értesítés számla kiállításáról

Minden új számla kiállításáról a felhasználó email-ben és alkalmazáson belül is értesítést kap. Az alkalmazás jobb felső sarkában található harang ikonra kattintva megtekinthetjük új értesítéseinket, amelyeknek számát a harang mellett található buborékban is jelezzük. A számla adatainak megtekintéséhez kattintsunk az értesítésre.

#### Számla letöltése

A kiállított számlákat szabadon letölthetjük PDF formátumban a számla alatt található Download as PDF gombra kattintva. A letöltött fájl tartalmazza a számla azonosítószámát, kiállításának dátumát, a számlázási időszak kezdetének és végének dátumát, illetve egy táblázatban lebontva a fogyasztások díjait a végösszeggel együtt.

# Fejlesztői dokumentáció

## Feladat elemzése

Az alkalmazás tervezését érdemes az alapprobléma megfogalmazásával kezdeni. A papír alapú folyamat túl lassú, és túl sok hibatényező áll fenn. Mi történik, ha valaki hiányosan tölti ki a papírt, vagy ha a közös képviselő hibásan rögzíti az adatokat. Ezeket mind ki tudnánk küszöbölni egy programmal. A mostani folyamat túl lassú, több nap, akár egy hét is eltelhet mire az adatokat feldolgozzák, és újabb több nap, amíg mindenkihez eljut a kiállított számla. Ez egy száz lakásból álló társasház esetén logisztikailag is megterhelő folyamat, és a közös képviselőnek akár több ilyen társasház ügyeit is intéznie kell párhuzamosan. Egy program esetén ez az egész folyamat automatizálható, és a lakókhoz a kiállítás után pillanatok alatt megérkezhet a számla.

Ezeket az igényeket remekül lefedheti egy webes alkalmazás. A lakók kényelme érdekében az oldalt mobil nézettel is el kell látni, mivel a mérőórák mellett állva így egyből beírhatják az adatokat a mobilkészülékükön keresztül.

### Felhasználói esetek

## Tervezés

Az alkalmazás architektúrája 3 komponensből épül fel. A resource szerver egy REST (representational state transfer) alapokra épített API (Application Programming Interface), ami az adatbázishoz kapcsolódva a rendszer backend felületét nyújtja. A kliens alkalmazás egy Single Page Application, ami AJAX hívásokkal fog kapcsolódni a resource szerverhez. A kliens programot egy static szerver fogja kiszolgálni.

## Szerver oldali architektúra

### RESTful API

A REST, azaz Representational State Transfer egy internetes architektúra típus, amelyben a hálózatot szerverek és kliensek alkotják. A kliensek kéréseket indítanak a szerver felé, amelyre a szerver valamilyen webes erőforrással (HTML, XML, JSON) válaszol. A REST architektúra hat megkötést fogalmaz meg, ami biztosítja az alkalmazások teljesítményét, skálázhatóságát és megbízhatóságát. Ezek a megkötések a következők:

1. Kliens-szerver architektúra: A kliensek és szerverek feladatköreit el kell különíteni. Például a szerverek nem foglalkozhatnak felhasználói felülettel, vagy a kliens állapotával, és a kliensek nem foglalkozhatnak adattárolással. Ez az elv biztosítja a kliens hordozhatóságát és szerver skálázhatóságát. A fejlesztést teljesen szétválaszthatjuk, amíg az interfész nem változik.
2. Állapotmentesség: A kommunikáció további megkötése, hogy a szerver nem tárolhatja a kliens állapotát kérések között. Minden kérésnek tartalmaznia kell elégséges információt, hogy a szerver képes legyen azt végrehajtani és a session állapotot a kliensnek kell megőrizni. Ez kifejezetten érdekes elmélet authentikációs szempontból.
3. Gyorsítárazhatóság: A kliensek gyorsítótárazhatnak bizonyos válaszokat. A válaszoknak tartalmazniuk kell, hogy tárazhatóak-e, vagy sem. Egy jól felépített tárazási stratégia megnövelheti a szerver skálázhatóságát.
4. Rétegelt felépítés: A kliensek nem képesek megmondani, hogy direkt kapcsolódott a szerverhez, vagy közvetítő szervereken keresztül, így terheléselosztó szerverek (load balancers) közbeiktatásával növelhetjük a skálázhatóságot.
5. Code on Demand (igényelhető kód): A szerverek ideiglenes kibővíthetik a kliens funkcionalítását futtatható programrészek elküldésével. Ezt a módszert alkalmazták a Java applet-ek, vagy a kliensoldali JavaScript szkriptek.
6. Egységes interfész: Egyszerűsíti és szétválasztja az architektúrát, ami megkönnyíti a kliensek és szerverek független fejlesztését.

A REST elveit követő webes szolgáltatásokat röviden RESTful Web Services-nek nevezzük. A kérések típusai a HTTP szabány metódusainak felelnek meg, azaz GET, POST, PUT, PATCH, DELETE, stb. A legelterjedtebb interfész a RESTful szolgáltatásoknál a JSON (JavaScript Object Notation).

#### Felhasznált technológiák

##### Node.js

A Node.js egy nyílt forráskódú, platformfüggetlen JavaScript futtatókörnyezet. A JavaScript-et történetileg a böngészők elsődleges nyelvének fejlesztették ki, de a webes szolgáltatások elterjedésével ma már az egyik legnépszerűbb programozási nyelvvé nőtte ki magát. A Node.js a Chrome böngésző JavaScript futtatómotorjára, a V8-ra épül, amely interpretálás helyett natív gépi kódra fordítja le a JavaScript forráskódot. Eseményvezérelt architektúrája ás aszinkron I/O műveletek támogatása mellett a Node.js funkcionalításának kiterjesztésére létrehozták az NPM (Node Package Manager) nevű csomagkezelőt, amely több mint 350 ezer csomaggal a világ legnagyobbjának számít. Hatékonysága és könnyen skálázhatósága miatt a Node.js különösen népszerű eszköz lett valós idejű alkalmazások készítésére. Fejlesztésének felügyelésére és piaci adoptálásának felgyorsítására létrehozták a Node.js Foundation-t, melynek tagjai között ott találjuk a Google, Microsoft, IBM, SAP, Intel, Redhat és a PayPal delegáltjait is.

##### Express.js

Az Express a Node.js beépített HTTP moduljára épülő, fejlesztőbarát API, amely remek absztrakciós réteget ad webszolgáltatások fejlesztéséhez.

#### Végpontok

A resource szerver elérhető a következő címen: https://api-rezsi.herokuapp.com/api/v1

Szerver működésének lekérdezése - URL: /health - METHOD: GET - Sikeres válasz: 200 OK - Sikertelen válasz: A szerver elérhetetlen, valamilyen hibakóddal tér vissza.

Authentikált felhasználó adatainak lekérdezése - URL: /auth - METHOD: GET - Sikeres válasz:

{  
 displayName: ...  
 email:   
}

* Sikertelen válasz: 401 UNAUTHORIZED

Felhasználó bejelentkezése - URL: /auth - METHOD: POST - Sikeres válasz:

{  
 user: { displayName, ... },  
 token: 'refwr42434mnj4h1j4h1j4',  
}

* Sikertelen válasz:

Felhasználó regisztrációja - URL: /users - METHOD: POST - Sikeres válasz: - Sikertelen válasz:

Felhasználó visszaigazolása - URL: /users/:id/confirm - METHOD: GET - Sikeres válasz: - Sikertelen válasz:

Authentikált felhasználó csoportjainak lekérdezése - URL: /groups - METHOD: GET - Sikeres válasz: - Sikertelen válasz: 401 UNAUTHORIZED

Új csoport létrehozása - URL: /groups - METHOD: POST - Sikeres válasz: - Sikertelen válasz: 401 UNAUTHORIZED

Egy csoport lekérdezése - URL: /groups/:id - METHOD: GET - Sikeres válasz: - Sikertelen válasz: 401 UNAUTHORIZED

Egy csoport adatainak módosítása - URL: /groups/:id - METHOD: PATCH - Sikeres válasz: - Sikertelen válasz: 401 UNAUTHORIZED

Egy csoport törlése - URL: /groups/:id - METHOD: DELETE - Sikeres válasz: - Sikertelen válasz: 401 UNAUTHORIZED

Csoport összesítéseinek lekérdezése: - URL: /groups/:id/summaries - METHOD: GET - Sikeres válasz: - Sikertelen válasz: 401 UNAUTHORIZED

Összesítés létrehozása: - URL: /groups/:id/summaries - METHOD: POST - Sikeres válasz: - Sikertelen válasz: 401 UNAUTHORIZED

Összesítés letöltése CSV formátumban: - URL: /groups/:id/summaries/:summaryId/csv - METHOD: GET - Sikeres válasz: - Sikertelen válasz: 401 UNAUTHORIZED

Csoport összes felhasználójának lekérdezése: - URL: /groups/:id/users - METHOD: GET - Sikeres válasz: - Sikertelen válasz: 401 UNAUTHORIZED

Csoport egy felhasználójának lekérdezése: - URL: /groups/:id/users/:userId - METHOD: GET - Sikeres válasz: - Sikertelen válasz: 401 UNAUTHORIZED

Csoport egy felhasználójának törlése: - URL: /groups/:id/users/:userId - METHOD: DELETE - Sikeres válasz: - Sikertelen válasz: 401 UNAUTHORIZED

Felhasználó óraállásainak lekérdezése: - URL: /groups/:id/users/:userId/reports - METHOD: GET - Sikeres válasz: - Sikertelen válasz: 401 UNAUTHORIZED

Óraállás bejelentése: - URL: /groups/:id/users/:userId/reports - METHOD: POST - Sikeres válasz: - Sikertelen válasz: 401 UNAUTHORIZED

Felhasználó számláinak lekérdezése: - URL: /groups/:id/users/:userId/bills - METHOD: GET - Sikeres válasz: - Sikertelen válasz: 401 UNAUTHORIZED

Felhasználó számláinak letöltése PDF formátumban: - URL: /groups/:id/users/:userId/bills/pdf - METHOD: GET - Sikeres válasz: - Sikertelen válasz: 401 UNAUTHORIZED

### Adatbázis

#### Felhasznált technológiák

##### MongoDB

A MongoDB egy nyílt forráskódú, NoSQL, dokumentumorientált adatbázis. Az adatokat BSON (JSON-höz hasonló) formátumban tárolja. A dokuementumok szerkezeti felépítését Schema-k segítségével definiálhatjuk. Lekérdezésekben és aggregációs függvényekben natívan használhatunk JavaScript kifejezéseket, emiatt igazán elterjedt Node.js alapú rendszerekben.

##### Mongoose

A Mongoose egy Node.js csomag, ami megkönnyíti a MongoDB Schema-k definiálását, validálását, és egy magasabb szintű intefészt biztosít a MongoDB dokumentumom kezelésére.

### Modellek

User: - email - password - displayName - role - group - confirmed - disabled

Group: - name - leader - disabled

Report: - hotWater - coldWater - heat - user

Summary: - from - to - hotWaterPrice - coldWaterPrice - heatPrice - currency - group

Bill: - hotWaterConsumption - coldWaterConsumption - heatConsumption - summary - user

Notification: - type - seen - user - bill

## Kliens oldali architektúra

### Felhasznált technológiák

#### React

#### Redux

### Komponens hierarchia

### Kommunikáció

## Authentikáció

### Felhasznált technológiák

#### Passport.js

#### JWT

## Értesítési rendszer

### Felhasznált technológiák

#### WebSocket

## DVD melléklet tartalma

A melléklet két mappát tartalmaz. Az api.rezsi.io nevű mappában a resource szerver forráskódját, az app.rezsi.io mappában a static szerver és a kliens alkalmazás forráskódját találjuk.

api.rezsi.io  
├── src  
| ├── config  
| ├── controllers  
| ├── models  
| ├── routes  
| ├── utils  
| └── index.js  
├── .env.example  
├── .editorconfig  
├── .eslintignore  
├── .eslintrc  
├── package.json  
└── yarn.lock

app.rezsi.io  
├── app  
| ├── components  
| ├── containers  
| ├── images  
| ├── tests  
| ├── translations  
| ├── utils  
| ├── app.js  
| ├── configureStore.js  
| ├── global-styles.js  
| ├── i18n.js  
| ├── index.html  
| ├── manifest.json  
| └── reducers.js  
├── internals  
├── server  
├── .editorconfig  
├── package.json  
├── Procfile  
└── yarn.lock

A mappák további konfigurációs fájlokat is tartalmazhatnak, de ezek a dokumentáció szempontjából nem lényegesek.

## Fejlesztőkörnyezet

### Rendszerkövetelmények

* macOS (Sierra v10.12.6 és újabb verziók)
* MongoDB (v3.4.2)
* Node.js (v8.4.0 és újabb verziók)
* NPM (v5.3.0 és újabb verziók)
* Yarn.pkg (v1.0.1 és újabb verziók)

A fejlesztés természetesen történhet más operációs rendszeren is, de a követelményekben feltüntetett eszközök elérhetőségét és kompatibilitását az adott rendszeren előzetesen meg kell vizsgálni.

### A fejlesztőkörnyezet kialakítása

Másoljuk a DVD melléklet tartalmát a számítógépre.

#### MongoDB szerver elindítása

Készítsünk egy új mappát, ahol az adatbázis fájlokat fogjuk tárolni, majd indítsuk el a MongoDB szervert.

$ mkdir rezsi.io  
$ mongod --dbpath rezsi.io

#### Resource szerver konfigurálása

A szervert környezetváltozókkal konfigurálhatjuk. A legegyszerűbb módja ennek ha létrehozunk egy .env fájlt az api.rezsi.io mappa gyökerében, amelyben kulcs-érték párokban felsoroljuk a változókat.

* NODE\_ENV: beállíthatjuk, hogy a szerver éles vagy fejlesztői módban induljon el
* PORT: szerver portja
* JWT\_SECRET: JWT token generálásához szükséges titkosítási kulcs
* MONGO\_HOST: MongoDB szerver kapcsolódási URI-ja
* MONGO\_PORT: MongoDB szerver kapcsolódási portja
* GMAIL\_USER: GMail SMTP felhasználó neve
* GMAIL\_PASS: GMail SMTP felhasználó jelszava
* GMAIL\_ADDRESS: GMail SMTP-n keresztül küldött emailek feladója
* DEBUG: debug.js csomag konfigurációs stringje

A forrásfájlok között találhatunk egy .env.example névvel ellátott példafájlt, amely tartalmazza az alapvető konfigurációt. Ezt elég csak átnevezni és átírni az értékeket.

$ cp .env.example .env

Fokozottan figyeljünk arra, hogy ez a fájl jelszókat és titkosítási kulcsokat tartalmaz, ezért semmilyen körülmények között sem juthat illetéktelen kezekbe. Ha ez mégis megtörténik, azonnal cseréljünk jelszót és változtassuk meg a kulcsokat. Ezt a fájlt ne commitoljuk a verziókövető rendszerbe.

#### Resource szerver elindítása

Navigáljunk a DVD mellékletről származó forrásmappába, telepítsük a JavaScript dependenciákat és indítsuk el a szervert.

$ cd api.rezsi.io  
$ yarn install  
$ yarn start

A szervert watch üzemmódban is indíthatjuk, ekkor minden fájlmódosításkor a szerver újraindul.

$ yarn start-dev

#### Static szerver elindítása

Navigáljunk a DVD mellékletről származó forrásmappába, telepítsük a JavaScript dependenciákat és indítsuk el a szervert.

$ cd app.rezsi.io  
$ yarn install  
$ yarn start

## Fejlesztési munkafolyamat

### Verziókövetés

A fejlesztéshez a Git verziókövető rendszert használtam. A resource szervert és a kliens alkalmazást (a static szerverrel együtt) külön repository-ban helyeztem el, mivel ezek a projektek akár párhuzamosan is fejleszthetők és csak egy közös interfészre van szükség a kommunikációjukhoz. A repositoryk elérhetőek a GitHub felületén is.

### Continuous delivery

Az alkalmazást a Heroku felhő platformon futtatjuk. Ez a szolgáltatás a legtöbb üzemeltetéssel kapcsolatos terhet leveszi a fejlesztő válláról. A rendszer összeköthető a GitHub repositoryval, így a master branch változáskor automatikusan kirakja az új verziót az éles környezetbe.

Mivel nem commitolhatjuk be a verziókövető rendszerbe a szenzitív adatokat tartalmazó .env fájlt, ezért a környezetváltozókat a Heroku erre a célra kialakított felületén kell megadni.

### Tesztelés

## Fejlesztési lehetőségek

* Többnyelvűség támogatása: Az alkalmazás felépítése során kezdettől fogva szem előtt tartottam a többnyelvűség támogatását így csak a feliratok lefordítása és egy nyelvválasztó felület hiányzik.
* Értesítési rendszer kiszervezése külön web szolgáltatásba: A rendszert saját adatbázzissal kell ellátni, ami tárolja az értesítéseket. A resource szerver HTTP kéréseken keresztül kommunikál az értesítési szolgáltatással.