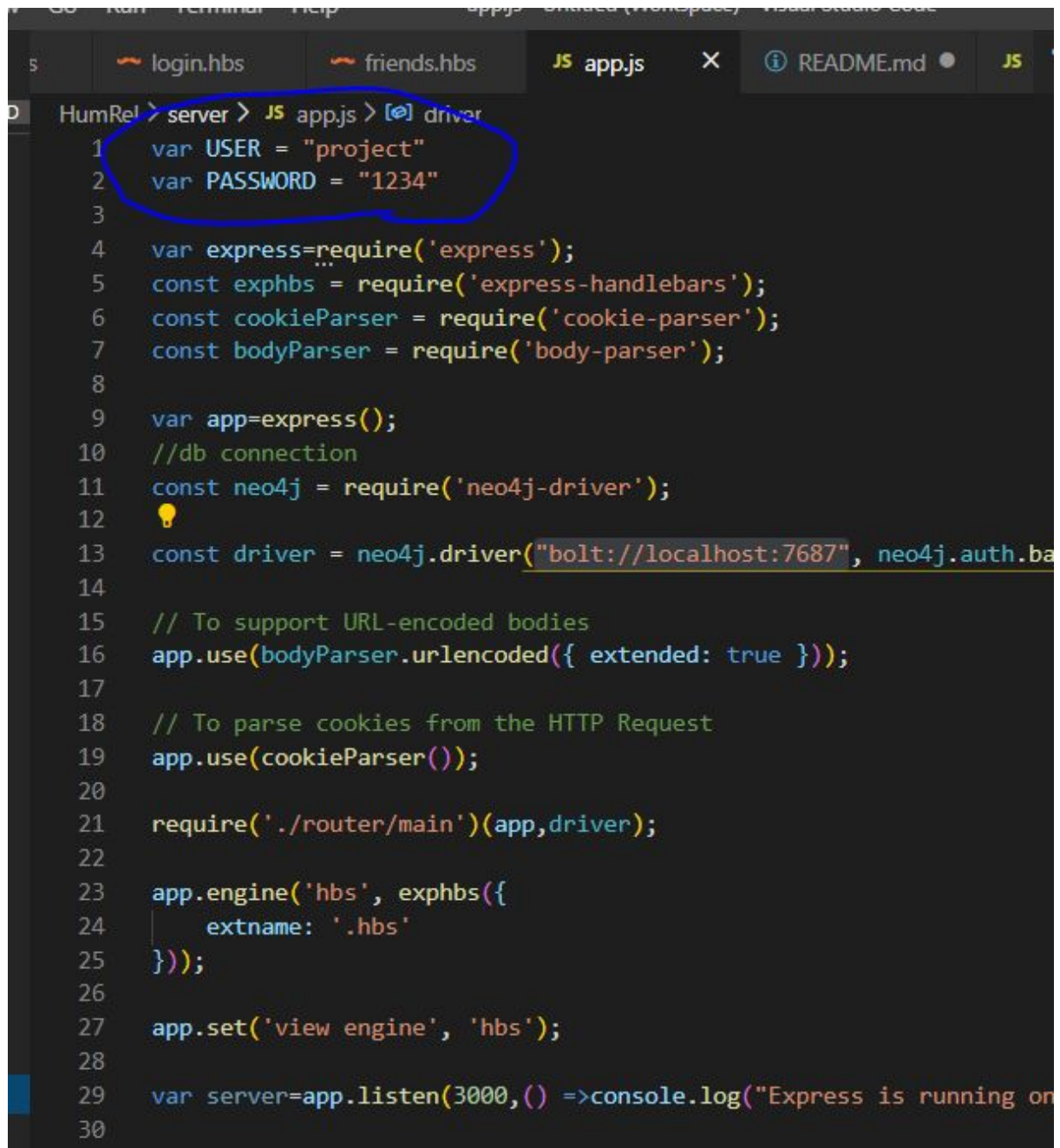


Sprawozdanie z Projektu
Projekt Bazy Danych II
Radosław Kopeć
Paweł Pławecki

<https://github.com/RadekKpc/HumRel>

W pliku server/app.js znajdują się dane do połączenia z bazą Neo4j:

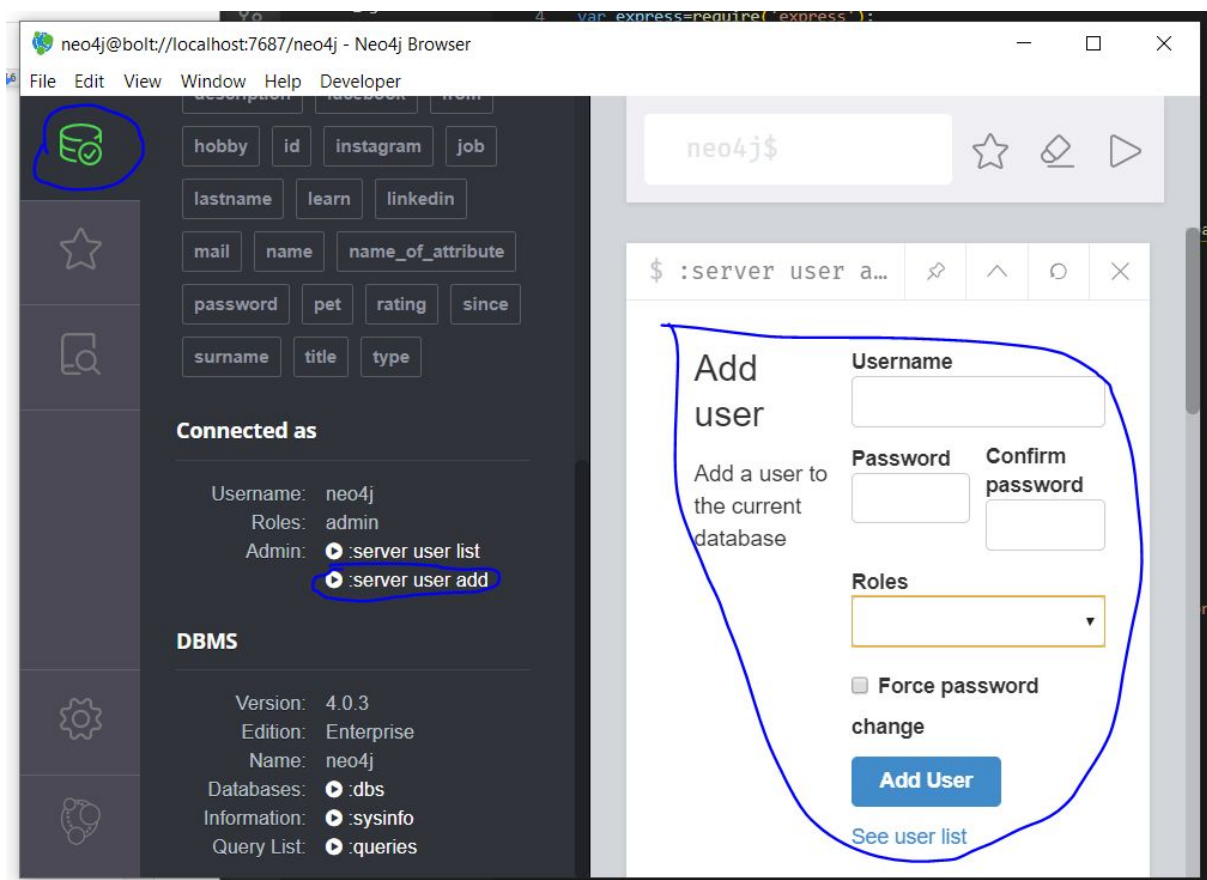


```
HumRel > server > JS app.js > [?] driver
1  var USER = "project"
2  var PASSWORD = "1234"
3
4  var express=require('express');
5  const exphbs = require('express-handlebars');
6  const cookieParser = require('cookie-parser');
7  const bodyParser = require('body-parser');
8
9  var app=express();
10 //db connection
11 const neo4j = require('neo4j-driver');
12
13 const driver = neo4j.driver("bolt://localhost:7687", neo4j.auth.basic(USER, PASSWORD));
14
15 // To support URL-encoded bodies
16 app.use(bodyParser.urlencoded({ extended: true }));
17
18 // To parse cookies from the HTTP Request
19 app.use(cookieParser());
20
21 require('./router/main')(app,driver);
22
23 app.engine('hbs', exphbs({
24   extname: '.hbs'
25 }));
26
27 app.set('view engine', 'hbs');
28
29 var server=app.listen(3000,() =>console.log("Express is running on port 3000"))
30
```

adres połączenia: bolt://localhost:7687

Należy utworzyć nową bazę danych i uruchomić w niej skrypt
server/db_script.txt (metodą kopiuj wklej)

Następnie należy dodać do niej użytkownika o nazwie "project" i hasło
"1234",



Gdy już mamy bazę danych pora na uruchomienie aplikacji

Wchodzimy do folderu server

W konsoli wpisujemy: npm install

Po zainstalowaniu pakietów:

node app.js

Serwer pracuje na localhost:3000

Wchodzi do przeglądarki, logujemy się do aplikacji korzystając z przygotowanego użytkownika.

login: test@test.com

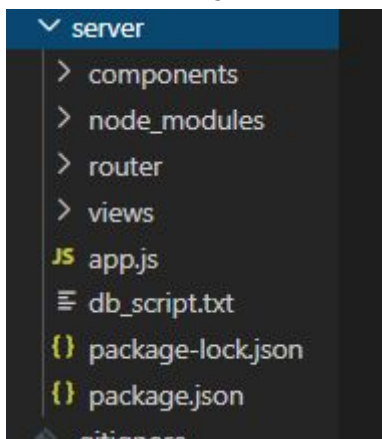
hasło: 1234

Polecam w drugim oknie przeglądarki zarejestrować się jako inny użytkownik żeby przetestować zapraszanie, przyjmowanie zaproszeń, wyszukiwanie, rejestracje.

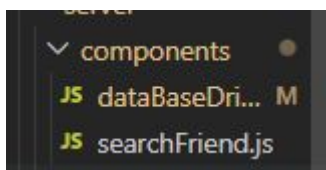
Opis Kodu Radosław Kopeć

Server:

Struktura projektu:



katalog "components" zawiera komponent obsługujące wysyłanie zapytań do bazy, oraz komponent do wyszukiwania znajomych (Nie są one wykorzystywane jednak logika jest zaimplementowana)



JS dataBaseDriver.js X

HumRel > server > components > JS dataBaseDriver.js > dataBaseDriver > exec_

```
1  class dataBaseDriver{
2
3      constructor(driver) {
4          this.driver= driver;
5      }
6
7
8      async exec_question(request){
9          const session = this.driver.session()
10         try {
11             const result = await session.run(request)
12             result.records.forEach(element => {
13                 console.log("element",element.get(0))
14             });
15             return result.records
16         }
17         finally {
18             await session.close()
19         }
20     }
21
22 }
23 module.exports = dataBaseDriver
24
```

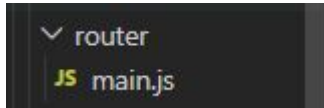
JS searchFriend.js X

HumRel > server > components > JS searchFriend.js > ...

```
1  const dataBaseDriver = require("../dataBaseDriver.js")
2
3  class searchFriends {
4
5      constructor(driver) {
6          this.db = new dataBaseDriver(driver)
7      }
8
9      async search(searchString){
10
11         const words = searchString.split(' ');
12         let result = [];
13         console.log("words:", words)
14         words.forEach(async s => {
15             let prom = await this.db.exec_question("MATCH (m:Person) WHERE
16                 console.log("prom:",prom)
17                 result.push(prom)
18             });
19         console.log("result: ",result)
20         return result;
21     }
22 }
23
24 module.exports = searchFriends
25
```

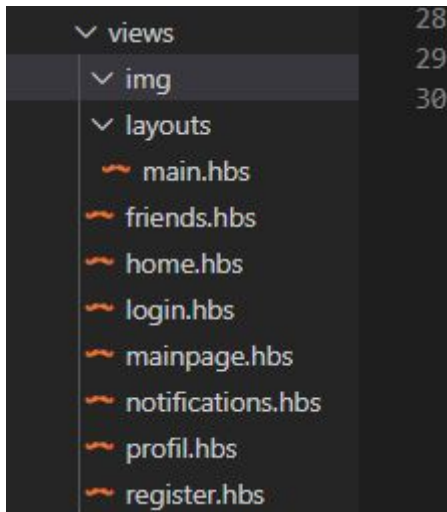
katalog "node_modules" - pakiety zainstalowane za pomocą polecenia npm install

katalog router - zawiera jeden plik "main.js"



W którym zdefiniowane są wszystkie end-pointy, aplikacji, jest w nim zaimplementowana główna logika serwera zostanie on omówiony później.

Katalog "views" zawiera widoki aplikacji, templatki generowane za pomocą ślinika hbs.



Plik "app.js" główny plik uruchomieniowy serwera, obsługuje łączenie z bazą, definiuje użycie silnika hbs.

```
JS app.js X
HumRel > server > JS app.js > [e] USER
1  var USER = "project"
2  var PASSWORD = "1234"
3
4  var express=require('express');
5  const expubs = require('express-handlebars');
6  const cookieParser = require('cookie-parser');
7  const bodyParser = require('body-parser');
8
9  var app=express();
10 //db connection
11 const neo4j = require('neo4j-driver');
12
13 const driver = neo4j.driver("bolt://localhost:7687", neo4j.auth.basic(USER,PASSWORD));
14
15 // To support URL-encoded bodies
16 app.use(bodyParser.urlencoded({ extended: true }));
17
18 // To parse cookies from the HTTP Request
19 app.use(cookieParser());
20
21 require('./router/main')(app,driver);
22
23 app.engine('hbs', expubs({
24   extname: '.hbs'
25 }));
26
27 app.set('view engine', 'hbs');
28
29 var server=app.listen(3000,() =>console.log("Express is running on port 3000"));
30
```

Dane użytkownika do połączenia z bazą danych:

```
1  var USER = "project"
2  var PASSWORD = "1234"
```

Połączenie z bazą:

```
10 //db connection
11 const neo4j = require('neo4j-driver');
12
13 const driver = neo4j.driver("bolt://localhost:7687", neo4j.auth.basic
14
```

Ustawienie silnika do widoków html.


```

app.use(cookieParser());

require('./router/main')(app,driver);

app.engine('hbs', exphbs({
  extname: '.hbs'
}));

app.set('view engine', 'hbs');

```

Nasłuchiwanie:

```

var server=app.listen(3000,() =>console.log("Express is running on port 3000"));

```

Widoki:

Template, w miejsce `{{body}}` są generowane odpowiednie widoki

main.hbs

```

main.hbs X
HumRel > server > views > layouts > main.hbs > ...
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <meta http-equiv="X-UA-Compatible" content="ie=edge">
7   <title>HumRel</title>
8
9   <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css">
10 </head>
11 <body>
12
13 <div class="container">
14   <div class="row">
15     <div class="col">
16       <div class="card">
17         <div class="card-body">
18           <div class="card-text">
19             {{body}}
20           </div>
21         </div>
22       </div>
23     </div>
24   </div>
25 </div>
26
27 <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"></script>
28 <script src="https://cdn.jsdelivr.net/npm/popper.js@1.14.7/dist/umd/popper.min.js"></script>
29 <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"></script>
30 </body>
31 </html>

```

home.hbs

Strona startowa, umożliwi przejście do rejestracji i logowania:

```
home.hbs X
HumRel > server > views > home.hbs > div
1 <nav class="navbar navbar-expand-lg navbar-light bg-light">
2   <a class="navbar-brand" href="/mainpage">Humrel Please Register</a>
3 </nav>
4
5 <div style="margin-top: 30px">
6   <a class="btn btn-primary btn-lg active" href="/login">Login</a>
7   <a class="btn btn-primary btn-lg active" href="/register">Register</a>
8 </div>
```

login.hbs

Strona do logowania

```
login.hbs X
HumRel > server > views > login.hbs > ...
1 <div class="row justify-content-md-center" style="margin-top: 100px">
2   <div class="col-md-6">
3
4     {{#if message}}
5     <div class="alert {{messageClass}}" role="alert">
6       {{message}}
7     </div>
8     {{/if}}
9
10    <form method="POST" action="/login">
11      <div class="form-group">
12        <label for="exampleInputEmail1">Email address</label>
13        <input name="email" type="email" class="form-control" id="exampleInputEmail1" placeholder="Enter email">
14      </div>
15      <div class="form-group">
16        <label for="exampleInputPassword1">Password</label>
17        <input name="password" type="password" class="form-control" id="exampleInputPassword1" placeholder="Password">
18      </div>
19      <button type="submit" class="btn btn-primary">Login</button>
20      <a class="btn btn-primary" href="/register">Register</a>
21    </form>
22  </div>
23 </div>
24
25 </div>
26
```

register.hbs

Strona do rejestracji


```

register.hbs X
HumRel > server > views > register.hbs > div.row.justify-content-md-center > div.col-md-4 > form
1 <div class="row justify-content-md-center" style="margin-top: 30px">
2   <div class="col-md-4">
3
4     {{#if message}}
5       <div class="alert {{messageClass}}" role="alert">
6         {{message}}
7       </div>
8     {{/if}}
9
10    <form method="POST" action="/register">
11      <div class="form-group">
12        <label for="firstNameInput">First Name</label>
13        <input name="firstName" type="text" class="form-control" id="firstNameInput">
14      </div>
15
16      <div class="form-group">
17        <label for="lastNameInput">Last Name</label>
18        <input name="lastName" type="text" class="form-control" id="lastNameInput">
19      </div>
20
21      <div class="form-group">
22        <label for="emailInput">Email address</label>
23        <input name="email" type="email" class="form-control" id="emailInput" placeholder="Enter email">
24      </div>
25
26      <div class="form-group">
27        <label for="passwordInput">Password</label>
28        <input name="password" type="password" class="form-control" id="passwordInput" placeholder="Password">
29      </div>
30
31      <div class="form-group">
32        <label for="confirmPasswordInput">Confirm Password</label>
33        <input name="confirmPassword" type="password" class="form-control" id="confirmPasswordInput"
34          placeholder="Re-enter your password here">
35      </div>
36
37      <button type="submit" class="btn btn-primary">Login</button>
38    </form>
39  </div>
40 </div>

```

We wszystkich poniższych plikach znajduje top-bar postaci:
Zawiera on przejście do friends, settings, homepage, notifications.

```

<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <a class="navbar-brand" href="/mainpage">HumRel</a>
  <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">
    <span class="navbar-toggler-icon"></span>
  </button>

  <div class="collapse navbar-collapse" id="navbarSupportedContent">
    <ul class="navbar-nav mr-auto">
      <li class="nav-item active">
        <a class="nav-link" href="/mainpage/my_profile">Profil</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="/mainpage/friends">Friends</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="/mainpage/notifications">Notifications
          {{#if notifications}}
            <b> {{notifications}} </b>
          {{/if}}
        </a>
      </li>
      <li class="nav-item dropdown">
        <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown" role="button" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
          Settings
        </a>
        <div class="dropdown-menu" aria-labelledby="navbarDropdown">
          <a class="dropdown-item disabled" href="#">Layout</a>
          <a class="dropdown-item disabled" href="#">Theme</a>
          <div class="dropdown-divider"></div>
          <a class="dropdown-item" href="#">Profil</a>
        </div>
      </li>
    </ul>

    <form method="POST" action="/logout">
      <div>
        <button type="submit" class="btn btn-primary">Logout</button>
      </div>
    </form>
  </div>
</nav>

```

Plik profil.hbs zawiera profil użytkownika

```
<div class="col-xs-1 center-block">
<div class="card" style="width: 18rem;">
  
  <div class="card-body">
    <h5 class="card-title">{{name}} {{lastname}}</h5>
    <p class="card-text">{{description}}</p>
  </div>
  <ul class="list-group list-group-flush">
    <li class="list-group-item">Age: {{age}}</li>
    <li class="list-group-item">Job {{job}}</li>
  </ul>
  <div class="card-body">
    <a href={{facebook}} class="card-link">Facebook</a>
    <a href={{instagram}} class="card-link">Instagram</a>
    <a href={{linkedin}} class="card-link">LinkedIn</a>
  </div>
</div>
</div>
```

Plik notification.hbs zawierający powiadomienia użytkownika

```
<form method="POST" action="/mainpage/search">
  <div class="md-form mt-0">
    <input name="search_string" class="form-control" type="text" placeholder="Search friends" aria-label="Search">
  </div>
  <div>
    <br>
    <button type="submit" class="btn btn-primary">Search</button>
  </div>
</form>
<br>
<br>
{{#if invites}}
<div>
  <ul>
    <{{#each invites}}>
      <li class="list-group-item" >{{this.name}} {{this.lastname}} invites you to firends
        <form style="display:inline;" method="POST" action="/mainpage/accept_invite" >
          <button type="submit" class="btn btn-success btn-small" >Accept</button>
          <input type="hidden" name="id" value="{{this.id}}" />
          <input type="hidden" name="invites" value="{{invites}}" />
          <input type="hidden" name="type" value="accept" />
        </form>
        <form style="display:inline;" method="POST" action="/mainpage/accept_invite" >
          <button type="submit" class="btn btn-danger btn-small" >Not Accept</button>
          <input type="hidden" name="id" value="{{this.id}}" />
          <input type="hidden" name="type" value="reject" />
          <input type="hidden" name="invites" value="{{invites}}" />
        </form>
      </li>
    </{{each}}>
  </ul>
</div>
</{{if}}>
```

plik mainpage.hbs zawierający pasek wyszukiwania znajomych:

```
<br>
    {{#if message}}
        <div class="alert {{messageClass}}" role="alert">
            {{message}}
        </div>
    {{/if}}
<br>
<div>
    <form method="POST" action="/mainpage/search">
        <div class="md-form mt-0">
            <input name="search_string" class="form-control" type="text" placeholder="Search friends" aria-label="Search">
        </div>
        <div>
            <br>
            <button type="submit" class="btn btn-primary">Search</button>
        </div>
    </form>
    <br>
    <br>
    {{#if friends}}
    <div>
        <ul>
            {{#each friends}}
            <li class="list-group-item" >{{this.name}} {{this.lastname}}
                <form style="display:inline;" method="POST" action="/mainpage/invite" >
                    <button type="submit" class="btn btn-success btn-small">Invite</button>
                    <input type="hidden" name="id" value="{{this.id}}" />
                </form>
            </li>
            {{/each}}
        </ul>
    </div>
    {{/if}}
</div>
```

plik friends.hbs zawierający znajomych:

```
<div>
    <form method="POST" action="/mainpage/search">
        <div class="md-form mt-0">
            <input name="search_string" class="form-control" type="text" placeholder="Search friends" aria-label="Search">
        </div>
        <div>
            <br>
            <button type="submit" class="btn btn-primary">Search</button>
        </div>
    </form>
    <br>
    <br>
    {{#if friends}}
    <div>
        <ul>
            {{#each friends}}
            <li class="list-group-item" >{{this.name}} {{this.lastname}}
                <form style="display:inline;" method="POST" action="/mainpage/profile" >
                    <button type="submit" class="btn btn-success btn-small">View Profile</button>
                    <input type="hidden" name="id" value="{{this.id}}" />
                </form>
            </li>
            {{/each}}
        </ul>
    </div>
    {{/if}}
</div>
```

main.js

Zawiera on logikę aplikacji, definiuje end-pointy.

```
const dataBaseDriver = require("../components/dataBaseDriver.js")

const searchFriends = require("../components/searchFriend.js")

const crypto = require('crypto');

const getHashedPassword = (password) => {
  const sha256 = crypto.createHash('sha256');
  const hash = sha256.update(password).digest(['base64']);
  return hash;
}

const generateAuthToken = () => {
  return crypto.randomBytes(30).toString('hex');
}

const requireAuth = (req, res, next) => {
  if (req.user) {
    console.log("User authenticated " + req.user.properties.name + " " + req.user.properties.lastname )
    // console.log(req.user)
    next();
  } else {
    res.render('login', {
      message: 'Please login to continue',
      messageClass: 'alert-danger'
    });
  }
};
```

getHashedPassword - zwraca sha256 hasła

generateAuthToken - zwraca token do autentykacji

requireAuth - bardzo ważna funkcja która wymusza bycie zalogowanym przy przejściu na wybrane adresy.

Endpointy do głównej strony, (requireAuth wymusza autentykacje)

```
app.get('/', function (req, res) {
  res.render('home');
});

app.get('/login', (req, res) => {
  res.render('login');
});

const authTokens = {};

app.get('/mainpage', requireAuth, (req, res) => {
  res.render('mainpage');
});

app.post('/logout', requireAuth, (req, res) => {
  res.cookie('AuthToken', null);
  res.redirect('/login');
});
```

Kod wysyłający zapytanie do bazy o użytkownika, powoduje zalogowanie, uzyskanie tokana i zapisanie go w cookies:

```
app.post('/login', async function(req, res) {
  const { email, password } = req.body;
  const hashedPassword = getHashedPassword(password);
  console.log(email)
  console.log(hashedPassword)
  const session = driver.session()
  try {
    const result = await session.run(
      'MATCH (a:Person {password: "' + hashedPassword + '", mail: "' + email + '"}) RETURN a'
    )
    var = 0
    // here parse to json and send to html
    result.records.forEach(element => {
      var += 1
    });
    if(var != 0){
      result.records.forEach(element => {
        user = element.get(0)
      });
    }
    else{
      user = false
    }
  } finally {
    await session.close()
  }

  if (user) {
    const authToken = generateAuthToken();

    // Store authentication token
    authTokens[authToken] = user;

    // Setting the auth token in cookies
    res.cookie('AuthToken', authToken);

    // Redirect user to the protected page
    res.redirect('/mainpage');
  } else {
    res.render('login', {
      message: 'Invalid username or password',
      messageClass: 'alert-danger'
    });
  }
});
```


Kod rejestrujący użytkownika:

```
app.post('/register', async function (req, res) {
  const { email, firstName, lastName, password, confirmPassword } = req.body;

  // Check if the password and confirm password fields match
  if (password !== confirmPassword) {
    const hashedPassword = getHashedPassword(password);
    // Store user into the database if you are using one
    const session = driver.session()
    id = Math.random()* 1000000000;
    try {
      const result = await session.run(
        `CREATE (:a:Person {id: "${id.toString()}", mail: "${email}", password: "${hashedPassword}", name: "${firstName} ${lastName}", date_of_account: "${Date.now()}", description: `
      )
      // here phrase to json and send to html
    } finally {
      await session.close()
    }
    // on application exit:

    res.render('login', {
      message: 'Registration Complete. Please login to continue.',
      messageClass: 'alert-success'
    });
  } else {
    res.render('register', {
      message: 'Password does not match.',
      messageClass: 'alert-danger'
    });
  }
});
```

Kod pobierający znajomych użytkownika, sprawdza czy już taki nie istnieje w bazie:

```
app.get('/register', (req, res) => {
  res.render('register');
});

app.get('/mainpage/friends', requireAuth, async (req, res) => {
  friends = []

  const authToken = req.cookies['AuthToken'];
  id = authTokens[authToken].properties.id;
  const session = driver.session()
  query = `MATCH (p:Person {id: "${id}"})-[:FRIENDS]->(friend) return {name: friend.name, lastname: friend.lastname, id: friend.id}`

  try {
    const result = await session.run(query)
    result.records.forEach(element => {
      friends.push(element._fields[0])
    });
  } finally {
    await session.close()
  }

  res.render('friends', {
    friends: friends
  });
});
```

Kod pobierający profil użytkownika:

```
app.post('/mainpage/profile', requireAuth, async (req, res) => {
  const { id } = req.body;
  profil = {}

  let query = `MATCH (m:Person {id: "${id}"}) RETURN {id: m.id, name: m.name, lastname: m.lastname, age: m.age, description: m.description, facebook: m.facebook, instagram: m.instagram, linkedin: m.linkedin, job: m.job}`

  const session = driver.session()
  try {
    const result = await session.run(query)
    result.records.forEach(element => {
      profil = element._fields[0]
    });
  } finally {
    await session.close()
  }
  console.log(profil)
  res.render('profil', profil);
});
```

Kod wyszukujący użytkowników:

```
app.post('/mainpage/search', requireAuth, async (req, res) => {
  const { search_string } = req.body;
  firends = []
  const words = search_string.split(' ');
  let query = "MATCH (m:Person) WHERE"
  words.forEach(e => {
    query += " m.name CONTAINS '"+ e +"' OR m.lastname CONTAINS '"+e +"' OR ";
  })
  query = query.slice(0, -3)
  query += " RETURN {name: m.name, lastname: m.lastname, id: m.id}"

  const session = driver.session()
  try {
    const result = await session.run(query)
    result.records.forEach(element => {
      console.log(element)
      firends.push(element._fields[0])
    });
  }
  finally {
    await session.close()
  }
  console.log(firends)

  res.render('mainpage', {
    friends: firends
  });
});
```

Kod zapraszający do znajomych:

```
app.post('/mainpage/invite', requireAuth, async (req, res) => {

  message = "Wysłano zaproszenie"
  messageClass = "alter alert-success"
  const { id } = req.body
  const authToken = req.cookies['AuthToken'];

  your_id = authTokens[authToken].properties.id;

  query = 'MATCH (a:Person),(b:Person) WHERE a.id = "' + your_id + '" AND b.id = "' + id + '" CREATE (a)-[r:INVITE]->(b)'

  const session = driver.session()
  try {
    const result = await session.run(query)
    result.records.forEach(element => {
      console.log(element)
    });
  }
  finally {
    await session.close()
  }
  res.render('mainpage', {
    message: message,
    messageClass: messageClass
  });
});
```

Kod pobierający profil użytkownika:

```
app.get('/mainpage/my_profile', requireAuth, async (req, res) => {  
    const authToken = req.cookies['AuthToken'];  
  
    user = authTokens[authToken].properties;  
    user.password = "##"  
    console.log(authTokens[authToken])  
    console.log(user)  
    res.render('profil', user);  
});
```

Kod pobierający powiadomienia użytkownika:

```
app.get('/mainpage/notifications', requireAuth, async (req, res) => {  
    const authToken = req.cookies['AuthToken'];  
  
    id = authTokens[authToken].properties.id;  
  
    invites = []  
  
    const session = driver.session()  
    query = 'MATCH (p:Person {id: "' + id + '"})<-[:INVITE]-(:friend) return {name: friend.name, lastname: friend.lastname, id: friend.id}'  
    try {  
        const result = await session.run(query)  
        result.records.forEach(element => {  
            console.log(element)  
            invites.push(element._fields[0])  
        });  
    }  
    finally {  
        await session.close()  
    }  
    console.log(invites)  
  
    res.render('notifications', {  
        invites: invites  
    });  
});
```

Kod odpowiadający za dodanie do znajomych:

```

app.post('/mainpage/accept_invite', requireAuth, async (req, res) => {

  const authToken = req.cookies['AuthToken'];

  const { id, type, invites } = req.body;

  your_id = authTokens[authToken].properties.id;

  query = 'MATCH (p:Person {id: "' + id + '"})-[r:INVITE]->(p2:Person {id: "' + your_id + '"}) DELETE r'
  const session = driver.session()
  try {
    const result = await session.run(query)
    result.records.forEach(element => {
      console.log(element)
    });
  }
  finally {
    await session.close()
  }

  if(type == "accept"){
    query1 = 'MATCH (a:Person {id: "' + id + '"}), (b:Person {id: "' + your_id + '"}) CREATE (a)-[r:FRIENDS]-(b)'
    query2 = 'MATCH (a:Person {id: "' + id + '"}), (b:Person {id: "' + your_id + '"}) CREATE (a)-[r:FRIENDS]-(b)'
    const session = driver.session()
    try {
      await session.run(query1)
      await session.run(query2)
    }
    finally {
      await session.close()
    }
    console.log("Akceptuje " + id )
  }
  if(type == "reject"){
    console.log("Odrzucam " + id )
  }

  res.render('notifications', {
    invites: invites
  });

});

```

Generator:

Klasa PeopleCreator:

W tej klasie na początku mamy różne tablice Stringów typu: znaki, imiona, zawody, itp. Następnie w funkcji generateRandomPeople() tworzymy sobie pliki people.txt, gdzie będą przechowywane polecenia do utworzenia osób w neo4j oraz plik id_name.txt, gdzie będzie tworzona tablica stringów z ID ludzi, potrzebna do utworzenia relacji między nimi. Otwieramy nasze strumienie plików. Tworzymy 200 osób do wstawienia do bazy. Pod kolejne zmienne typu name, job, itp. wrzucamy losowe imiona, prace, itd. Tworzymy również losowe adresy e-mail oraz zakodowane hasła. Nasze ID osób zapisujemy również w id_name.txt. Następnie w linii 296 tworzymy nasze zapytania neo4j-owe. W atrybutach Person znajdują się: name, lastname, age, mail, password, date_of_account, job, facebook, instagram, linkedin i description. Dodajemy kolejne linijki do plików people.txt i id_name.txt tak, aby można było w łatwy sposób wygenerować dane do naszej bazy.

Kod (samej funkcji):

```
public void generateRandomPeople() throws IOException {  
    File customersOutput = new File("people.txt");  
    File idNameOutput = new File("id_name.txt");  
    FileOutputStream fileOutputStream1 = new FileOutputStream(customersOutput);  
    FileOutputStream fileOutputStream2 = new FileOutputStream(idNameOutput);  
    BufferedWriter bufferedWriter1 = new BufferedWriter(new  
        OutputStreamWriter(fileOutputStream1));  
    BufferedWriter bufferedWriter2 = new BufferedWriter(new  
        OutputStreamWriter(fileOutputStream2));  
    bufferedWriter2.write("String id_names[] = {");  
    bufferedWriter2.newLine();  
    for (int i = 0; i < 200; i++) {  
        String name = names[(int)(Math.random() * 1000) % names.length];  
        String surname = surnames[(int)(Math.random() * 1000) % surnames.length];  
        String job = jobs[(int)(Math.random() * 1000) % jobs.length];  
        String description = "Random description";  
        int age = (int)(Math.random() * 50) + 18;  
        String hash = "#";  
        String randomNumber = "";  
        String randomMailNumber = "";  
        for(int j=0;j<20;j++){  
            randomNumber = randomNumber.concat(numbers[(int)(Math.random() * 1000) %  
                numbers.length]);  
        }  
        for(int c=0;c<2;c++){  
            randomMailNumber = randomMailNumber.concat(numbers[(int)(Math.random() * 1000) %  
                numbers.length]);  
        }  
    }  
}
```

```

}

String randomPassword = "";

for(int k=0;k<30;k++){

randomPassword = randomPassword.concat(randomLetters[(int)(Math.random() * 10000)
% randomLetters.length]);

}

DateTimeFormatter dtf = DateTimeFormatter.ofPattern("yyyy/MM/dd HH:mm:ss");

LocalDateTime date = LocalDateTime.now();

String id_name = name + "_" + surname + "_" + randomNumber;

String mail = name.toLowerCase() + surname.toLowerCase() + randomMailNumber +
"@example.com";

bufferedWriter1.write("CREATE (" + name + "_" + surname + "_" + randomNumber +
":Person {id:'" + randomNumber + "', name:'" + name + "', lastname:'" + surname +
"', age:'" + age + "', mail:'" + mail + "', password:'" + randomPassword + "',
date_of_account:'" + date + "', job:'" + job + "', facebook:'" + hash + "',
instagram:'" + hash + "', linkedin:'" + hash + "', description:'" + description +
"'})");

bufferedWriter1.newLine();

bufferedWriter2.write("\"" + id_name + "\",");

bufferedWriter2.newLine();

}

bufferedWriter2.write("}");

bufferedWriter1.close();

bufferedWriter2.close();

}

```

Klasa RelationsCreator:

W tej klasie tworzymy relacje między poszczególnymi osobami. Najpierw mamy tablicę z ID poszczególnych osób (niestety, trzeba ją przekopiować z pliku id_name.txt). Następnie mamy funkcję generatePeopleRelations(), w której tworzymy polecenia neo4j-owe do utworzenia połączeń między losowo wybranymi ludźmi. Najpierw tworzymy i otwieramy plik relations_creator.txt. Następnie zaczynamy dla każdej osoby tworzyć kolejne polecenia dodania

losowej liczby przyjaciół. Żeby nie powtórzyły nam się jakieś polecenia, mamy tablicę `tab[][]`, która przechowuje informacje o tym, czy dana krawędź została już utworzona. Następnie jeżeli nie została utworzona, to do pliku dodajemy polecenie dodania przyjaciela zarówno w jedną jak i w drugą stronę. Powtarzamy te operacje dla każdej wygenerowanej wcześniej osoby.

Kod (samej funkcji):

```
public void generatePeopleRelations() throws IOException {  
  
    File attrToCreatorsOutput = new File("relations_creator.txt");  
  
    FileOutputStream fileOutputStream = new FileOutputStream(attrToCreatorsOutput);  
  
    BufferedWriter bufferedWriter = new BufferedWriter(new  
        OutputStreamWriter(fileOutputStream));  
  
    bufferedWriter.write("CREATE");  
  
    bufferedWriter.newLine();  
  
    boolean tab[][] = new boolean[201][201];  
  
    for(int k=0;k<201;k++){  
        for(int l=0;l<201;l++){  
            tab[k][l] = false;  
        }  
    }  
  
    for(int i=0;i<id_names.length;i++){  
  
        tab[i][i]=true;  
  
        int randomNumber = (int)(Math.random() * 4 + 2);  
  
        for(int j=0; j<randomNumber;j++){  
  
            int rand = (int)(Math.random() * 201);  
  
            while(tab[i][rand]){  
  
                rand = (int)(Math.random() * 201);  
  
            }  
  
            bufferedWriter.write("(" + id_names[i] + ")-[:FRIENDS]->(" + id_names[rand] +  
                "),");  
        }  
    }  
}
```

```

bufferedWriter.newLine();

bufferedWriter.write("(" + id_names[rand] + ")-[:FRIENDS]->(" + id_names[i] +
"),");

bufferedWriter.newLine();

tab[i][rand] = tab[rand][i] = true;

}

}

}

```

Klasa Main:

W tej klasie odpalamy sobie funkcje tworzące dane pliki. Polecam najpierw odpalić funkcję generateRandomPeople(), później skopiować plik id_names.txt do klasy RelationsCreator, a na koniec odpalić funkcję generatePeopleRelations().

Kod (żeby go odpalić wystarczy go „odkomentować”):

```

public class Main {

    public static void main(String[] args) throws IOException {

        //PeopleCreator creator = new PeopleCreator();

        //creator.generateRandomPeople();

        //RelationsCreator rel_creator = new RelationsCreator();

        //rel_creator.generatePeopleRelations();

    }

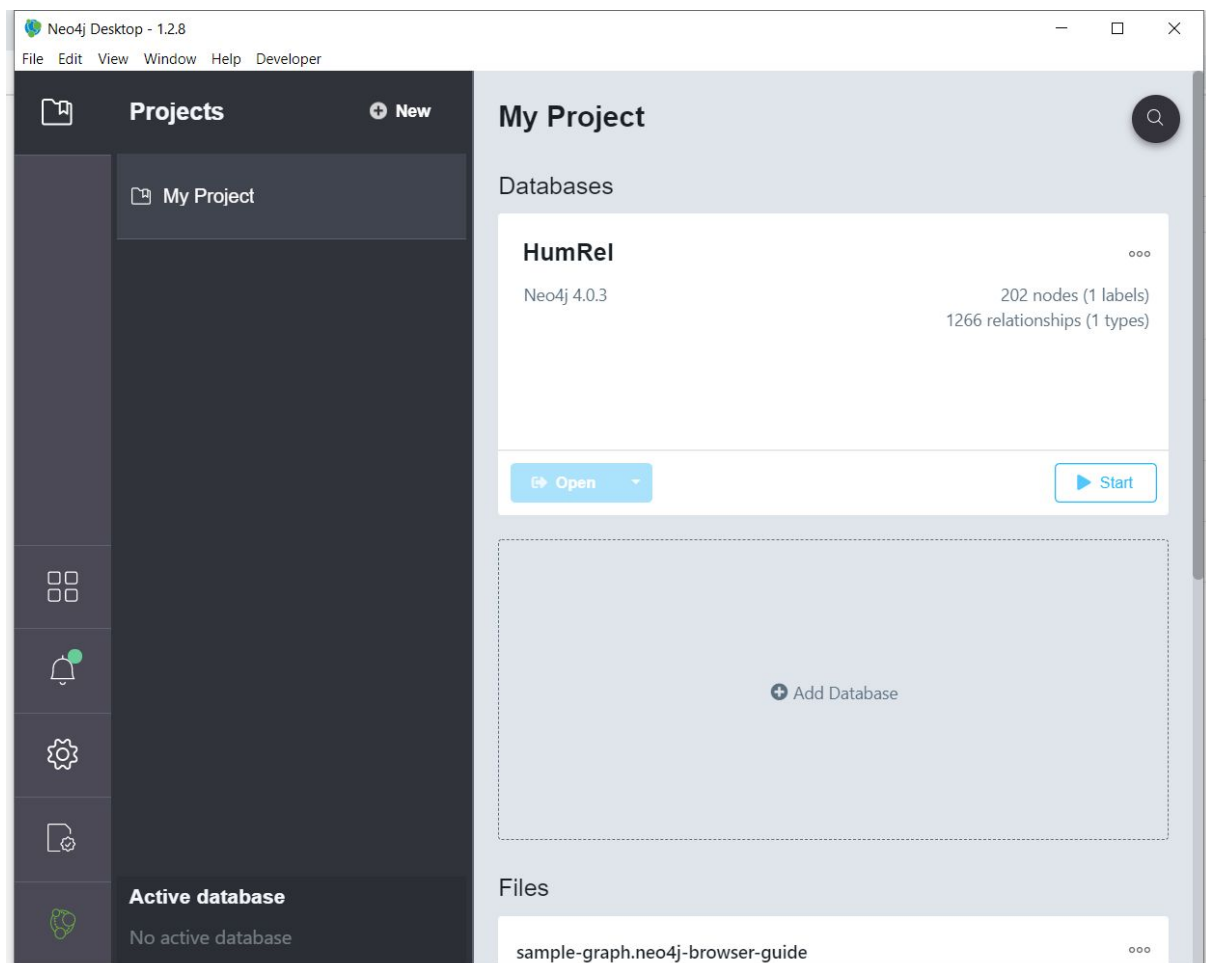
}

```

Po połączeniu plików people.txt oraz relations_creator.txt i wrzuceniu ich do pustej bazy neo4j utworzy nam się baza grafowa

Neo4J

Baza danych jest utworzona tutaj:

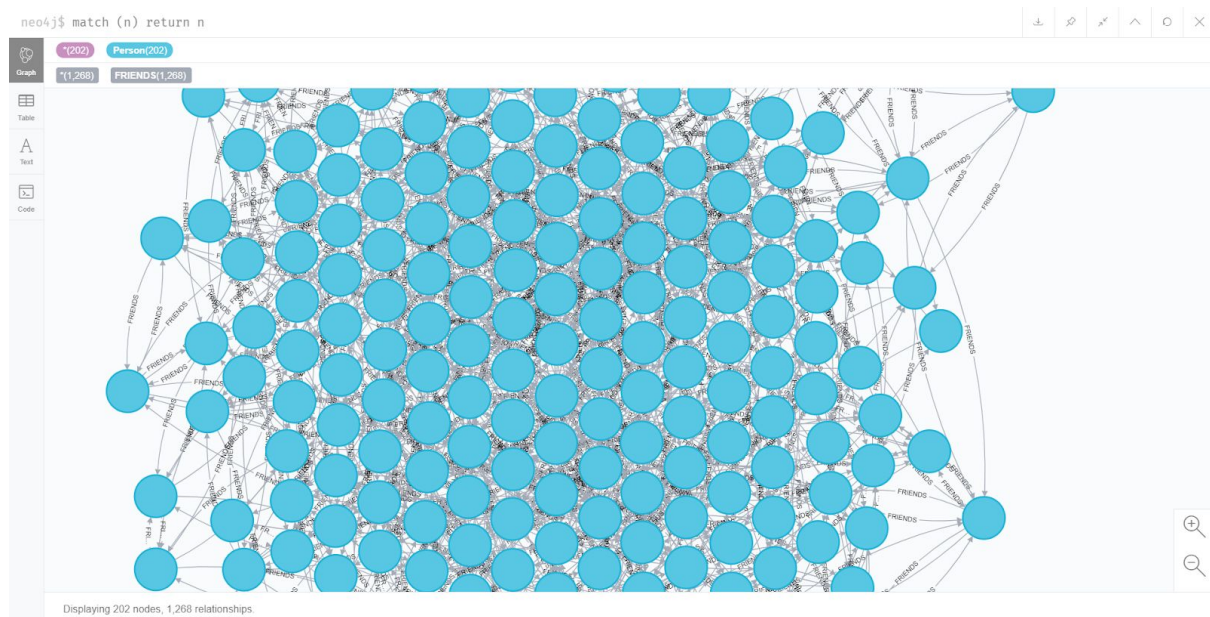


Odpalamy ją naciskając Start i wchodzimy do niej.

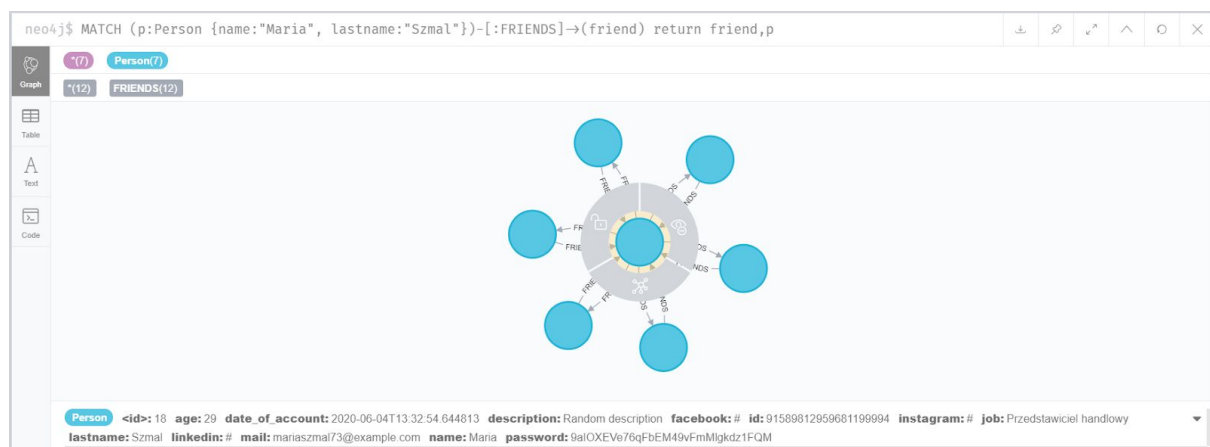
Za pomocą polecenia `match (n) return n` zobaczymy sobie, jak wygląda ta baza od środka.

```
neo4j$ match (n) return n
```

A tak wygląda nasza baza od środka:



Możemy sobie w niej np. za pomocą polecenia `match (p:Person {name:"Maria", lastname:"Szmal"})-[:FRIENDS]→ (friend) return friend,p` wyszukać wszystkich znajomych Marii Szmal:



Aplikacja:

Logujemy się do aplikacji w tym panelu:

Email address

test@test.com

Password

....

Login

Register

Zarejestrować się możemy w tym panelu:

← → ↻ localhost:3000/register

First Name

Paweł

Last Name

Plawecki

Email address

b@b.pl

Password

....

Confirm Password

....

Login

Po zalogowaniu się do systemu nasz panel główny wygląda tak:

HumRel Profil Friends Notifications Settings ▾

Logout

Search friends

Search

Możemy w oknie Search friends wyszukać nowych przyjaciół do dodania:

HumRel

Profil

Friends

Notifications

Settings ▾

Logout

Search

Bartosz Jurecki

Invite

Bartosz Kowalski

Invite

Bartosz Nowak

Invite

Bartosz Lasek

Invite

Bartosz Szukala

Invite

Bartosz Kondratiuk

Invite

Bartosz Boniek

Invite

Bartosz Szczepny

Invite

Możemy również ich zaprosić do znajomych poprzez kliknięcie przycisku Invite. PO kliknięciu, na danym koncie, które zaproszono pojawia się powiadomienie o możliwości zaakceptowania zaproszenia do znajomych:

Radosław Kopeć invites you to firends

Accept

Not Accept

Możemy również przejrzeć informacje o swoim profilu:



Radosław Kopeć
Random description

Age: 12

Job AGH

[Facebook](#) [Instagram](#) [LinkedIn](#)

Oczywiście możemy również za pomocą przycisku Friends przejrzeć swoich znajomych:

HumRel [Profil](#) [Friends](#) [Notifications](#) [Settings](#) [Logout](#)

Search friends

Search

pawel plawcki [View Profile](#)

Aleksandra Jakubiak [View Profile](#)

Aleksandra Grabarczyk [View Profile](#)

Bartosz Szukala [View Profile](#)

