

Jegyzőkönyv

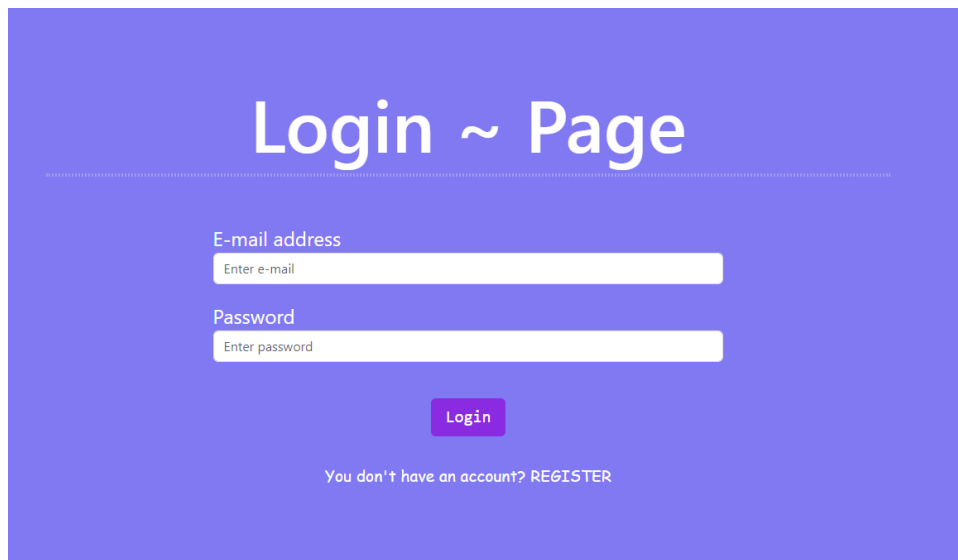
WEB technológiák 2

BEADANDÓ

Készítette: Rontó Eszter

Neptun kód: QTFL19

Az én nyilvántartó rendszerem animéket tart számon a nevük, a készítőjük és az epizódjaik alapján. Angol nyelven készítettem a weboldalt.

A screenshot of a login page with a purple background. The title "Login ~ Page" is centered at the top. Below it, there are two input fields: "E-mail address" with a placeholder "Enter e-mail" and "Password" with a placeholder "Enter password". A purple "Login" button is centered below the fields. At the bottom, there is a link that says "You don't have an account? REGISTER".

Login ~ Page

E-mail address
Enter e-mail

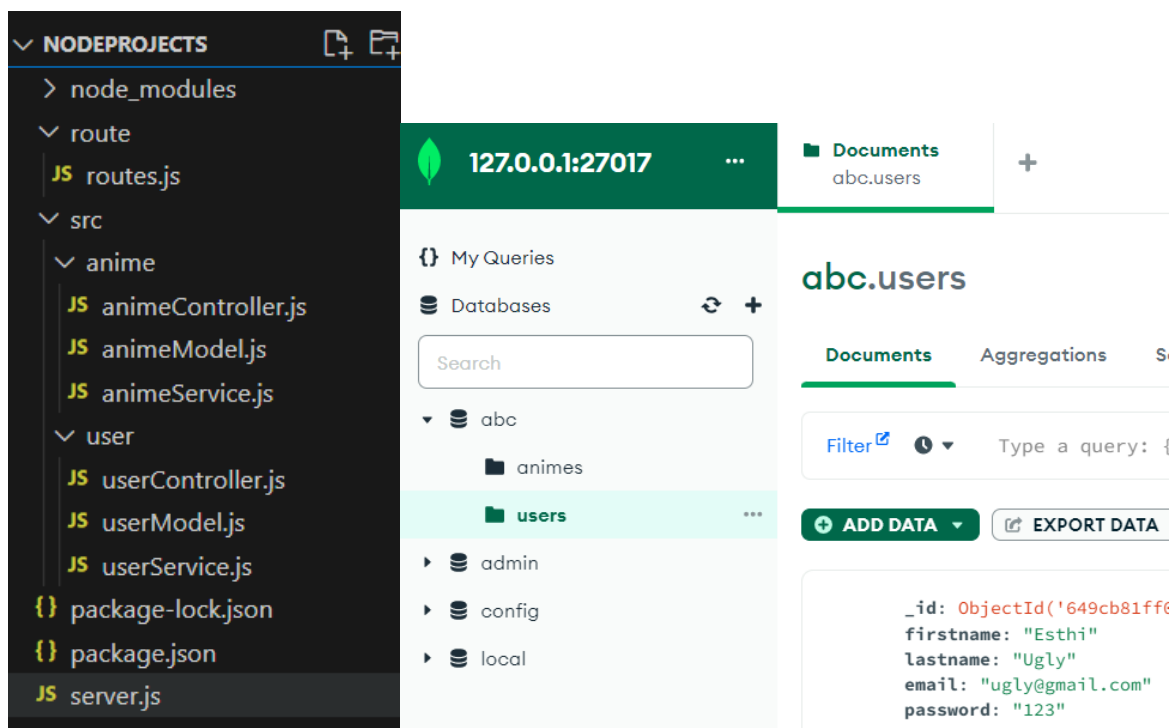
Password
Enter password

Login

You don't have an account? [REGISTER](#)

Amint belépünk a weboldalra rögtön a login felülettel kezd, ahonnan csak úgy lehet tovább haladni, ha helyes email címet és jelszót add meg.

Emellett fel ajánlja, hogy ha még nem regisztrált az ügyfél akkor add rá lehetőséget.

A composite screenshot showing three parts of the development environment. On the left is a file explorer for "NODEPROJECTS" showing a directory structure with files like routes.js, animeController.js, animeModel.js, animeService.js, userController.js, userModel.js, userService.js, package-lock.json, package.json, and server.js. In the center is the MongoDB Compass interface showing a database named "abc" with a collection named "users". On the right is the Postman interface showing a GET request to "abc.users" with a JSON response containing user details.

node_modules
route
JS routes.js
src
anime
JS animeController.js
JS animeModel.js
JS animeService.js
user
JS userController.js
JS userModel.js
JS userService.js
package-lock.json
package.json
JS server.js

127.0.0.1:27017

Documents
abc.users

My Queries
Databases
Search
abc
animes
users
admin
config
local

abc.users

Documents Aggregations S...

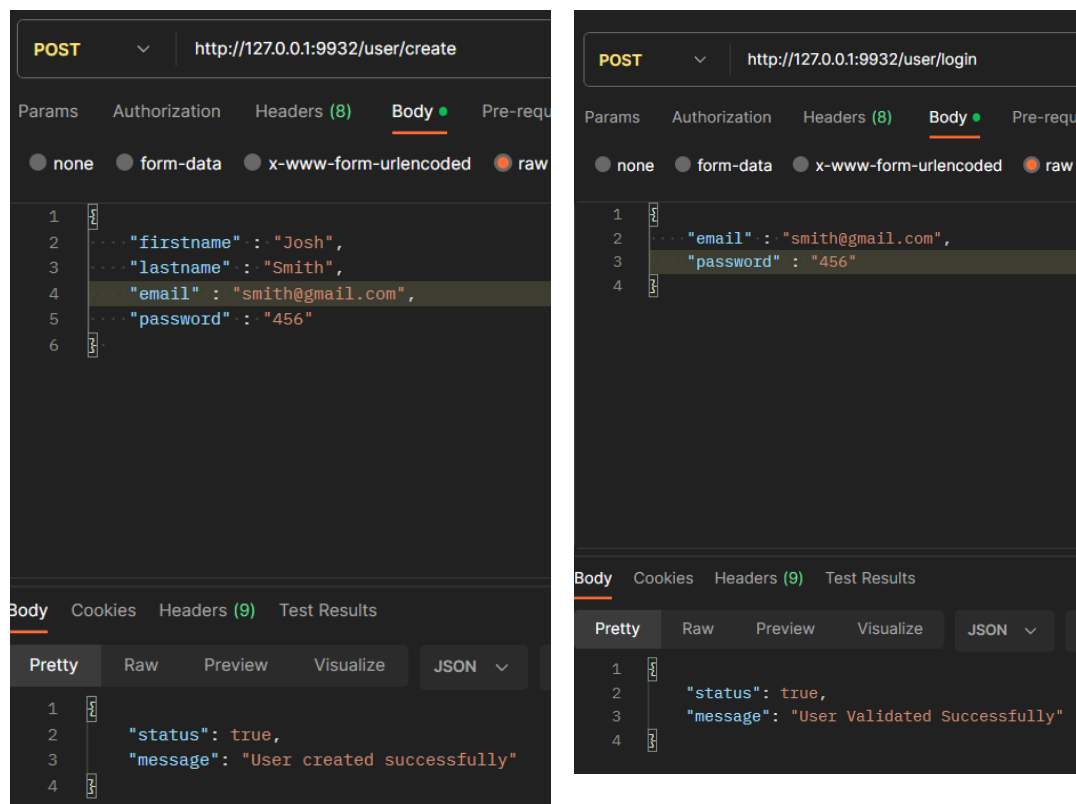
Filter Filter Type a query: {

ADD DATA EXPORT DATA

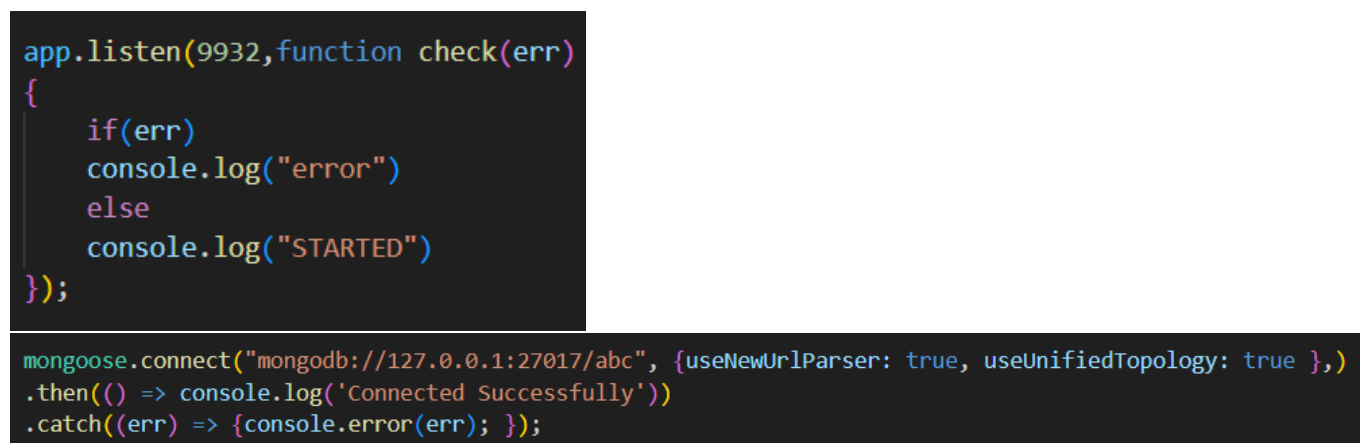
```
{
  "_id": ObjectId('649cb81ff6...'),
  "firstname": "Esthi",
  "lastname": "Ugly",
  "email": "ugly@gmail.com",
  "password": "123"
}
```

Annak érdekében, hogy itt a backend-em ("nodeprojects") működőképes legyen használtam a MongoDB-t és a PostMan-t.

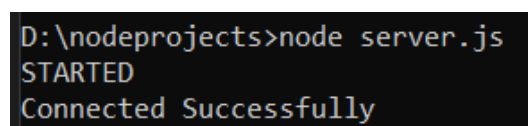
A PostMan segített ellenőrizni a hozzáadást amíg nem volt frontend-em. Emellet a login-t is le tudtam ellenőrizni.



Persze először ellenőriznem kellett, hogy a server elindul-e és ha igen csatlakozik-e a MongoDB-hez.



Ezt cmd-be (parancssorba) írtam ki ellenőrzés gyanánt.



A userModel-ben meg adtam az adatokat, amiket használni fog a felhasználó (amit kérek tőle). Ezt ugyan úgy meg tettem az animékkal (animeModel.js).

```
var mongoose = require('mongoose');
var Schema = mongoose.Schema;

var userSchema = new Schema({
  firstname: {
    type: String,
    required: true
  },
  lastname: {
    type: String,
    required: true
  },
  email: {
    type: String,
    required: true
  },
  password: {
    type: String,
    required: true
  }
});

module.exports = mongoose.model('user', userSchema);
```

```
var mongoose = require('mongoose');
var Schema = mongoose.Schema;

var animeSchema = new Schema({
  name: {
    type: String,
    required: true
  },
  creator: {
    type: String,
    required: true
  },
  episode: {
    type: String,
    required: true
  }
});

module.exports = mongoose.model('anime', animeSchema);
```

A userService, userController-ben és az animeService, animeController-ben néztem át a felhasználó által megadott adatokat. Hogy ha létrehozásról volt szó ellenőriztem, hogy mindent megadott-e helyesen string-nél string-et, int-nél int-et stb, és ki írtam, hogy ha sikeresen hozzá tudta adni vagy akár sikertelen lett a folyamat. Keresésnél pedig ellenőriztem, hogy a beírt érték szintén helyes-e és ha igen akkor meg található-e az adatbázisban szintén ki írtam, ha nem volt megtalálható az adatbázisban.

```
module.exports.createUserDBService = (userDetails) => {
  return new Promise(function myFn(resolve, reject) {
    var userModelData = new userModel();

    userModelData.firstname = userDetails.firstname;
    userModelData.lastname = userDetails.lastname;
    userModelData.email = userDetails.email;
    userModelData.password = userDetails.password;
    var encrypted = encryptor.encrypt(userDetails.password);
    userModelData.password = encrypted;

    userModelData.save(function resultHandle(error, result) {
      if(error) {
        reject(false);
      }
      else {
        resolve(true);
      }
    });
  });
}
```

```

module.exports.loginUserDBService = (userDetails) =>
{
  return new Promise(function myFn(resolve, reject)
  {
    userModel.findOne({ email: userDetails.email}, function getResult(errorvalue, result)
    {
      if(errorvalue)
      {
        reject({status: false, msg: "Invalid Data"});
      }
      else
      {
        if(result != undefined && result != null)
        {
          var decrypted = encryptor.decrypt(result.password);

          if(decrypted == userDetails.password)
          {
            resolve({status: true, msg: "User Validated Successfully"});
          }
          else
          {
            reject({status: false, msg: "User Validated Failed"});
          }
        }
        else
        {
          reject({status: false, msg: "Invalid User Details"});
        }
      }
    }
  )
}

```

```

var animeService = require('./animeService');

var findOneAnimeController = async (req, res) =>
{
  console.log(req.params.name );
  var result = await animeService.findOneAnimeDBService(req.params.name );

  if (result) {
    res.send({ "status": true, "data": result} );
  } else {
    res.send({ "status": false, "data": "Anime not found" });
  }
}

var createAnimeControllerFn = async (req, res) =>
{
  try
  {
    console.log(req.body);
    var status = await animeService.createAnimeDBService(req.body);
    console.log(status);

    if(status) {
      res.send({ "status": true, "message": "Anime added successfully"
    })
    }
    else {
      res.send({ "status": false, "message": "Error added anime" });
    }
  }
  catch(err)
  {
    console.log(err);
  }
}

module.exports = { findOneAnimeController, createAnimeControllerFn};

```

```

var animeModel = require('./animeModel');

module.exports.findOneAnimeDBService = (animeDetails) =>
{
  return new Promise(function myFn(resolve, reject)
  {
    animeModel.findOne({ name: animeDetails }, function returnData(error, result)
    {
      if(error)
      {
        reject({status: false, msg: "There is no anime like"});
      }
      else
      {
        resolve(result);
      }
    }
  )
}

```

```

module.exports.createAnimeDBService = (animeDetails) => {

  return new Promise(function myFn(resolve, reject) {

    var animeModelData = new animeModel();

    animeModelData.name = animeDetails.name;
    animeModelData.creator = animeDetails.creator;
    animeModelData.episode = animeDetails.episode;

    animeModelData.save(function resultHandle(error, result) {

      if(error) {
        reject(false);
      }
      else {
        resolve(true);
      }
    }
  )
}

```

Az animénél is használtam a PostMan-t amíg nem volt frontendem, hogy ellenőrizem a backendet.

```

GET http://127.0.0.1:9032/anime/findOne/Attack-on-Titan
"status": true,
"data": {
  "_id": "649ccc53272c97b7efc41f93",
  "name": "Attack-on-Titan",
  "creator": "Hajime Isayama",
  "episode": "88",

```

A routes.js-ben pedig összekötöttem mindent. A package.json-ban megtalálható az általam letöltött csomagok.

```

var express = require('express');

var userController = require('../src/user/userController');
var animeController = require('../src/anime/animeController');

const router = express.Router();

router.route('/user/login').post(userController.loginUserControllerFn);
router.route('/user/create').post(userController.createUserControllerFn);

router.route('/anime/findOne/:name').get(animeController.findOneAnimeController);
router.route('/anime/create').post(animeController.createAnimeControllerFn);

module.exports = router;

```

```

{
  "name": "nodeprojects",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \\\"Error: no test\\\"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "cors": "^2.8.5",
    "express": "^4.18.2",
    "mongodb": "^5.6.0",
    "mongoose": "^6.9.0",
    "simple-encryptor": "^4.0.0"
  },
  "devDependencies": {
    "nodemon": "^2.0.21"
  }
}

```

Frontend indítása

```
D:\frontend\frontend>ng serve
✓ Browser application bundle generation complete.

Initial Chunk Files | Names | Raw Size |
vendor.js           | vendor | 2.57 MB |
polyfills.js        | polyfills | 333.15 kB |
styles.css, styles.js | styles | 230.91 kB |
main.js             | main | 52.01 kB |
runtime.js          | runtime | 6.51 kB |
                    | Initial Total | 3.17 MB |

Build at: 2023-06-29T14:24:42.252Z - Hash: f00aee51c7f4b6ed - Time: 6631ms

** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **

✓ Compiled successfully.
```

- Login

Ha esetleg hibás adatokat adnak meg a felhasználók:

Login ~ Page

E-mail address

nemjoe@gmail.com

Password

...

Login

You don't have an account? REGISTER

```
"status": false,
"message": "Invalid User Details"
```

Ha helyesen adják meg és benne van az adatbázisban (ez esetben át is lép a home page-re):

Login ~ Page

E-mail address

smith@gmail.com

Password

...

Login

You don't have an account? REGISTER

```
+ _id: ObjectId('649d6830ba5922b3e3b33704')
2  firstname: "Josh"
3  lastname: "Smith"
4  email: "smith@gmail.com"
5  password: "456"
6  ...
```

```
"status": true,
"message": "User Validated Successfully"
```

Login.component.html


```

<body> (element) h2: HTMLHeadingElement
The h2 element represents a section heading.
<div cla MDN Reference
<h2>Login ~ Page</h2>
<hr/>
</div>
<div class="row">
  <div class="col-sm-6">
    <form>
      <div class="form-group">
        <label for="exampleInputEmail">E-mail address</label>
        <input type="text" [(ngModel)]="email" [ngModelOptions]="{standalone: true}" class="form-control" id="stname" place
      </div>
      <div class="form-group">
        <label for="exampleInputPassword1">Password</label>
        <input type="password" [(ngModel)]="password" [ngModelOptions]="{standalone: true}" class="form-control" id="stname"
      </div>
      <br>
      <div class="but">
        <button type="submit" class="btn btn-primary" (click)="login()">Login</button>
      </div>
      <div>
        <ul>
          <li><a href="http://localhost:4200/register">You don't have an account? REGISTER</a></li>
        </ul>
      </div>
    </div>
  </div>

```

Login.component.ts

Login.component.css

```

app > login > ts login.component.ts > ...
import { HttpClient } from '@angular/common/http';
import { Component } from '@angular/core';
import { Router } from '@angular/router';

@Component({
  selector: 'app-login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.css']
})
export class LoginComponent {

  email: string = '';
  password: string = '';

  isLogin: boolean = true;
  errorMessage: string = '';

  constructor(private router: Router, private http: HttpClient) {}

  login() {
    console.log(this.email);
    console.log(this.password);

    let bodyData = {
      email: this.email,
      password: this.password,
    };

    this.http.post("http://localhost:9932/user/login", bodyData)
      .subscribe(resultData => {
        console.log(resultData);

        if (resultData.status) {
          this.router.navigateByUrl('/home');
        }
      }, error => {
        console.log(error);
      });
  }
}

```

```

# login.component.css
src > app > login > # login.component.css > h2
1 body {
2   min-height: 100vh;
3   margin: 0;
4   background-color: #1a202c;
5 }
6
7 h2 {
8   color: #e2e3e5;
9   text-align: center;
10  margin-top: 10px;
11  font-size: 1.5em;
12 }
13
14 hr {
15   border: 2px dotted #e2e3e5;
16 }
17
18 div {
19   text-align: center;
20   margin: auto;
21   width: 80%;
22   padding: 10px;
23 }
24
25 div.form-group {
26   color: #e2e3e5;
27   font-size: 1.2em;
28   text-align: left;
29 }
30
31 button {
32   background-color: #2d3748;
33   border: none;
34   color: #e2e3e5;
35   padding: 8px 18px;
36   text-align: center;
37   text-decoration: none;
38   display: inline-block;
39   font-size: 1.1em;
40   font-family: monospace;

```

- Registration

A registration oldalról már csak home oldalra lehet tovább jutni persze csak akkor, ha helyesen adta meg az adatokat a felhasználó.



The image shows a registration form with a purple header and four input fields: First name (Josh), Last name (Smith), E-mail (smith@gmail.com), and Password (456). A purple 'Save' button is at the bottom right. To the right of the form is a black box with a JSON response: {"status": true, "message": "User created successfully"}. Below the form is a code block showing the data being sent: { "firstname": "Josh", "lastname": "Smith", "email": "smith@gmail.com", "password": "456" }.

Registration

First name
Josh

Last name
Smith

E-mail
smith@gmail.com

Password
456

Save

```
{ "status": true, "message": "User created successfully" }
```

```
{ "firstname": "Josh", "lastname": "Smith", "email": "smith@gmail.com", "password": "456" }
```

register.component.html

```

register.component.html X
src > app > register > register.component.html > body
Go to component
1 <body>
2
3 <div class="container mt-4">
4   <div class="card">
5     <h1>Registration</h1>
6     <hr/>
7     <form>
8       <div class="form-group">
9         <label>First name</label>
10        <input type="text" [(ngModel)]="firstname" [ngModelOptions]="{standalone: true}" class="form-control" id="stnam
11      </div>
12
13      <div class="form-group">
14        <label>Last name</label>
15        <input type="text" [(ngModel)]="lastname" [ngModelOptions]="{standalone: true}" class="form-control" id="stna
16      </div>
17
18      <div class="form-group">
19        <label>E-mail</label>
20        <input type="email" [(ngModel)]="email" [ngModelOptions]="{standalone: true}" class="form-control" id="course"
21      </div>
22
23      <div class="form-group">
24        <label>Password</label>
25        <input type="password" [(ngModel)]="password" [ngModelOptions]="{standalone: true}" class="form-control" id="
26      </div>
27
28      <button type="submit" class="btn btn-primary mt-4" (click)="save()">Save</button>
29
30    </form>
  </div>
</body>

```

register.component.ts

register.component.css

```

import { HttpClient } from '@angular/common/http';
import { Component } from '@angular/core';

@Component({
  selector: 'app-register',
  templateUrl: './register.component.html',
  styleUrls: ['./register.component.css']
})
export class RegisterComponent {

  firstname: string = "";
  lastname: string = "";
  email: string = "";
  password: string = "";

  constructor(private http: HttpClient) {
  }

  ngOnInit(): void {
  }

  register() {
    {
      let bodyData = {
        "firstname": this.firstname,
        "lastname": this.lastname,
        "email": this.email,
        "password": this.password,
      };
      this.http.post("http://localhost:9932/user/create", bodyData).subscribe((res) => {
        console.log(resultData);
        alert("User Registered Successfully")
      });
    }
  }

  save()

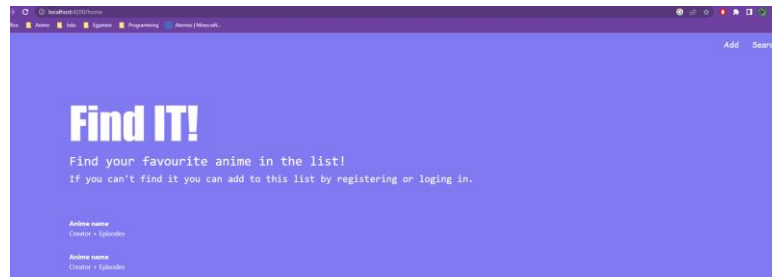
```

```

register.component.css X
> app > register > # register.component.css > div
1 body {
2   min-height: 100vh;
3   margin: 0;
4   background-color: #808080;
5   overflow: hidden;
6 }
7
8 h1 {
9   color: #808080;
10  text-align: center;
11  margin-top: 10%;
12  font-size: 550%;
13 }
14
15 div {
16   text-align: center;
17   margin: auto;
18   width: 80%;
19   border: none;
20   padding: 10px;
21   background-color: #808080;
22 }
23
24 hr {
25   border: 2px dotted #808080;
26 }
27
28 div.form-group {
29   color: #808080;
30   font-size: 150%;
31   text-align: left;
32 }
33
34 button {
35   background-color: #4169E1;
36   border: none;
37   color: #808080;
38   padding: 8px 18px;
39   text-align: center;
40   text-decoration: none;

```

- Home



A home oldalon sajnos nem tudtam végül megoldani az anime adatbázis ki íratását, azonban csináltam két gyors gombot a jobb oldali sarokba, hogy a felhasználó tudjon hozzá adni és keresni is az animék között.

home.component.html

```
home.component.html X
src > app > home > home.component.html > body
Go to component
<body>

  <ul>
    <li><a href="http://localhost:4200/search">Search</a></li>
    <li><a href="http://localhost:4200/add">Add</a></li>
  </ul>
  <h1>Find IT!</h1>
  <h2>Find your favourite anime in the list!</h2>
  <h3>If you can't find it you can add to this list by registering or logging in.</h3>

  <br><br><br>

  <dl>
    <dt>Anime name</dt>
    <dd>Creator + Episodes</dd>
    <br>
    <dt>Anime name</dt>
    <dd>Creator + Episodes</dd>
  </dl>
</body>
```

home.component.ts

```
TS home.component.ts X
src > app > home > TS home.component.ts > ...
1 import { Component } from '@angular/core';
2
3 @Component({
4   selector: 'app-home',
5   templateUrl: './home.component.html',
6   styleUrls: ['./home.component.css']
7 })
8 export class HomeComponent {
9
10 }
```

home.component.css

```
home.component.css X
src > app > home > home.component.css > h1
1 body {
2   min-height: 100vh;
3   margin: 0;
4   background-color: #121212;
5 }
6
7 h1 {
8   margin-top: 5%;
9   text-align: left;
10  margin-left: 10%;
11  color: white;
12  font-size: 670%;
13  font-family: fantasy;
14 }
15
16 h2 {
17   text-align: left;
18   margin-left: 10%;
19   color: white;
20   font-size: 200%;
21   font-family: monospace;
22 }
23
24 h3 {
25   text-align: left;
26   margin-left: 10%;
27   color: white;
28   font-size: 150%;
29   font-family: monospace;
30 }
31
32 ul {
33   list-style-type: none;
34   margin: 0;
35   padding: 0;
36   overflow: hidden;
37   font-size: 120%;
38   font-family: cursive;
39 }
```

- Add

Add

Name of the Anime

Name of the Creator

Episodes

Add

```
"status": true,  
"message": "Anime added successfully"
```

```
_id: ObjectId('649da5fdcd0723  
name: "Naruto"  
creator: "Kisimoto Maszasi"  
episode: "220"  
v: 0
```

Sikeresen hozzá adja az adatbázishoz, azonban mindig ki kell törölni a már egyszer megadott adatokat, hogy újakat tudjon megadni.

add.component.html

```
add.component.html X
src > app > add > add.component.html > body
Go to component
1 <body>
2
3 <div class="container mt-4" >
4   <div class="card">
5     <h1>Add</h1>
6     <hr/>
7     <form>
8       <div class="form-group">
9         <label>Name of the Anime</label>
10        <input type="text" [(ngModel)]="name" [ngModelOptions]="{standalone: true}" class="form-control" id="stname" place
11      </div>
12
13      <div class="form-group">
14        <label>Name of the Creator</label>
15        <input type="text" [(ngModel)]="creator" [ngModelOptions]="{standalone: true}" class="form-control" id="stname"
16      </div>
17
18      <div class="form-group">
19        <label>Episodes</label>
20        <input type="text" [(ngModel)]="episode" [ngModelOptions]="{standalone: true}" class="form-control" id="course" pl
21      </div>
22
23      <button type="submit" class="btn btn-primary mt-4" (click)="save()">Add</button>
24
25    </form>
  </div>
</div>
```

add.component.ts

```
import { HttpClient } from '@angular/common/http';
import { Component } from '@angular/core';

@Component({
  selector: 'app-add',
  templateUrl: './add.component.html',
  styleUrls: ['./add.component.css']
})
export class AddComponent {

  name: string = "";
  creator: string = "";
  episode: string = "";

  constructor(private http: HttpClient) {
  }

  ngOnInit(): void {
  }

  add() {
    let bodyData = {
      "name": this.name,
      "creator": this.creator,
      "episode": this.episode,
    };
    this.http.post("http://localhost:9932/anime/create", bodyData)
    {
      console.log(resultData);
      alert("Anime Added Successfully")
    }
  });

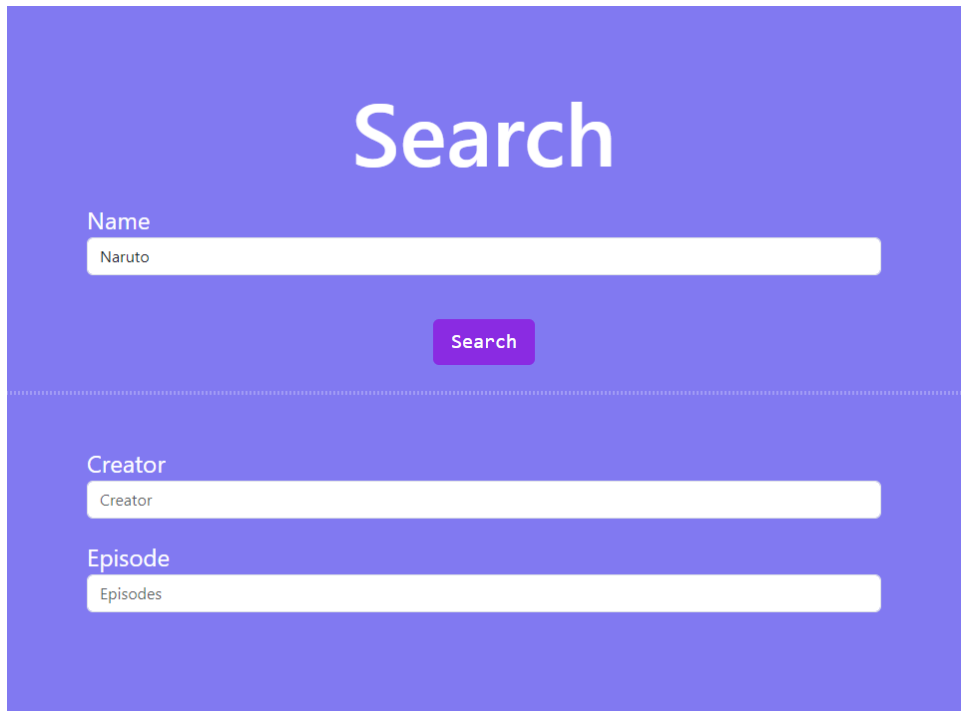
  save() {
    this.add();
  }
}
```

add.component.css

```
# add.component.css X
src > app > add > # add.component.css > body
1 body{
2   min-height: 100vh;
3   margin: 0;
4   background-color: #129, 121, 241;
5   overflow: hidden;
6 }
7
8 h1 {
9   color: #255, 255, 255;
10  text-align: center;
11  margin-top: 10%;
12  font-size: 550%;
13 }
14
15 div {
16   text-align: center;
17   margin: auto;
18   width: 80%;
19   border: none;
20   padding: 10px;
21   background-color: #129, 121, 241;
22 }
23
24
25 hr {
26   border: 2px dotted #fff;
27 }
28
29 div.form-group {
30   color: #fff;
31   font-size: 150%;
32   text-align: left;
33 }
34
35 button {
36   background-color: #4169E1;
37   border: none;
38   color: #fff;
39   padding: 8px 18px;
```

- Search

A search-öt úgy oldottam volna meg, hogy név alapján keresi ki és ha meg találja ki írhatja a "creator"-t és az "episode"-ot.



Search

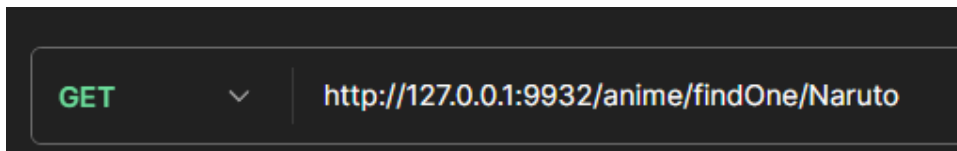
Name
Naruto

Search

Creator
Creator

Episode
Episodes

Sajnos magára az oldalra nem tudtam ki írni azonban a backend mégis működik, amit a PostMan-nel ellenőriztem.



GET http://127.0.0.1:9932/anime/findOne/Naruto

```
"status": true,  
"data": {  
  "_id": "649da5fdcd072352dd408519",  
  "name": "Naruto",  
  "creator": "Kisimoto Maszasi",  
  "episode": "220",  
  "__v": 0  
}
```

```

search.component.html
src > app > search > search.component.html > body
Go to component
1 <body>
2 <div class="container mt-4">
3   <div class="card">
4     <h1>Search</h1>
5
6     <form>
7       <div class="form-group">
8         <label>Name</label>
9
10        <input type="text" [(ngModel)]="name" [ngModelOptions]="{standalone: true}" class="form-control" id="name" placeholder="Name" />
11      </div>
12      <div>
13        <button type="submit" class="btn btn-primary mt-4" (click)="search()">Search</button>
14      </div>
15    </form>
16
17    <hr/>
18    <br/>
19    <div class="form-group">
20      <label>Creator</label>
21      <input type="text" [(ngModel)]="creator" [ngModelOptions]="{standalone: true}" class="form-control" id="creator" placeholder="Creator" />
22    </div>
23
24    <div class="form-group">
25      <label>Episode</label>
26      <input type="text" [(ngModel)]="episode" [ngModelOptions]="{standalone: true}" class="form-control" id="episode" placeholder="Episode" />
27    </div>
28  </div>
29 </div>
30 </body>

```

search.component.ts

```

app > search > TS search.components.ts > ...
import { HttpClient } from '@angular/common/http';
import { Component } from '@angular/core';

@Component({
  selector: 'app-search',
  templateUrl: './search.component.html',
  styleUrls: ['./search.component.css']
})
export class SearchComponent {

  name: string = "";
  creator: string = "";
  episode: string = "";

  constructor(private http: HttpClient) {
  }

  search() {
    this.http.post("http://localhost:9932/anime/findOne/" + this.name, {
      creator: this.creator,
      episode: this.episode
    })
    .subscribe(
      (resultData) => {
        console.log(resultData);
        if(resultData.data == 'Anime not found') {
          alert("Anime Record Not Found")
        } else {
          this.creator = resultData.data.creator;
          this.episode = resultData.data.episode;
        }
      }
    );
  }
}

```

search.component.css

```

# search.component.css
src > app > search > # search.component.css > body
1 body{
2   min-height: 100vh;
3   margin: 0;
4   background-color: #121212;
5   overflow: hidden;
6 }
7
8 h1 {
9   color: #f5f5f5;
10  text-align: center;
11  margin-top: 10px;
12  font-size: 550%;
13 }
14
15 div {
16   text-align: center;
17   margin: auto;
18   width: 80%;
19   border: none;
20   padding: 10px;
21   background-color: #121212;
22 }
23
24 hr {
25   border: 2px dotted #f5f5f5;
26 }
27
28 div.form-group {
29   color: #f5f5f5;
30   font-size: 150%;
31   text-align: left;
32 }
33
34
35 button {
36   background-color: #9c27b0;
37   border: none;
38   color: #f5f5f5;
39   padding: 8px 18px;
40   text-align: center;

```