



МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

Факультет прикладної математики
Кафедра програмного забезпечення комп’ютерних систем

Лабораторна робота № 8
з дисципліни “Основи програмування”
тема “Створення REST API сервера”

Виконав(ла)
студент(ка) І курсу
групи КП-____

Кривенко Віталій Ігорович
(прізвище, ім'я, по батькові)

варіант № 5

Перевірів
“ ____ ” “ ____ ” 20 ____ р.
викладач

Гадиняк Руслан Анатолійович
(прізвище, ім'я, по батькові)

Київ 2018

Мета роботи

Реалізувати JSON REST API для доступу до ресурсів веб-сервера.

Постановка завдання

1. У серверній частині додати JSON REST API для виконання CRUD операцій над ресурсами сервера (Users, {Entities}, {Composites}).
2. Отримувати об'єкти можна одиночно по ідентифікатору, або декілька списком. Отримання об'єктів списком дозволяє використовувати URL аргументи для фільтрації результату і обов'язково має мати пагінацію (не повертати всі результати одразу, а розбивати їх на підписки фіксованої довжини - "сторінки").
3. Всі URL для доступу до API мають починатись з /api/v1 (наприклад, /api/v1/users, /api/v1/users/13 і т.д). При моделюванні шляхів використовувати загальні рекомендації.
4. Запит GET /developer/v1 має повертати HTML сторінку, у якій коротко задокументувати реалізований REST API версії v1:
 1. Доступні ресурси, їх URL, методи доступу до них (які параметри мають, який результат повертається)
 2. Спосіб авторизації для доступу до ресурсів
 3. Формат і види помилок

Тексти коду програм

frontend/src/app/users/register/register.component.ts

```
import { Component, OnInit } from '@angular/core';
import { HttpResponse } from '@angular/common/http';
import { Router } from '@angular/router';
import { MatDialog } from '@angular/material/dialog';
import { MessageBoxData, MessageBoxComponent } from '@vee/components/message-box';
import * as Api from '@public-api/v1';
import * as Vts from 'vee-type-safe';
import { escapeStringRegExp } from '@utils/helpers';
import { ErrorHandlingService } from '@services/error-handling';
import { PageHeaderService } from '@services/page-header';
import { RoutingService } from '@services/routing';
import { SessionService } from '@services/session';
import { AuthService } from '@services/auth';
import RegisterResponse = Api.Auth.Register.Post.Response;
import RegisterRequest = Api.Auth.Register.Post.Request;

@Component({
  selector: 'app-register',
  templateUrl: './register.component.html',
  styleUrls: ['./register.component.scss']
})
export class RegisterComponent implements OnInit {
  credentials: RegisterRequest = {
    login: '',
    password: '',
    fullname: ''
  };

  Api = Api.Data.User;
  escapeRegExp = escapeStringRegExp;
  constructor(
    private error: ErrorHandlingService,
    private pageHeader: PageHeaderService,
    private rt: RoutingService,
    private session: SessionService,
    private auth: AuthService,
    private dialog: MatDialog,
    private router: Router
  ) { }

  ngOnInit() {
    this.pageHeader.setHeader({
      title: 'Register'
    });
  }

  onFormSubmit() {
    this.auth.register(this.credentials).subscribe(
      res => {
        this.session.rawToken = res.jwt;
        void this.router.navigateByUrl(this.rt.to.home());
      },
      err => {
        if (!(err instanceof HttpResponse) ||
          !Vts.conforms<RegisterResponse.Failure>(err.error,
            RegisterResponse.FailureTD)) {
```

```

        return this.error.handle(err);
    }
    this.dialog.open<MessageBoxComponent, MessageBoxData>(
        MessageBoxComponent, {
            data: {
                contentHtml: `Failure: ${err.error.failure}`,
                titleHtml: `Registration failure`
            }
        });
    }
    });
}
}
}

```

frontend/src/app/users/register/register.component.html

```

<mat-card class="col-md-6 mx-auto">
  <mat-card-header>
    <mat-card-title>Register</mat-card-title>
  </mat-card-header>
  <mat-card-content>
    <form
      (ngSubmit)="onFormSubmit();"
      #registerForm="ngForm"
    >
      <mat-form-field>
        <input #fullnameEl
          matInput
          [(ngModel)]="credentials.fullname"
          [pattern]="Api.Fullname.regex"
          type="text"
          placeholder="Fullname"
          name="fullname"
          required
        >
        <mat-hint align="start">
          <strong>{{Api.Fullname.formatHint}}</strong>
        </mat-hint>
        <mat-hint align="end">
          {{fullnameEl.value.length}} / {{Api.Fullname.maxLength}}
        </mat-hint>
      </mat-form-field>
      <br>
      <mat-form-field>
        <input #loginEl matInput
          [(ngModel)]="credentials.login"
          [pattern]="Api.Login.regex"
          type="text"
          placeholder="Login"
          name="login"
          required
        >
        <mat-hint align="start">
          <strong>{{Api.Login.formatHint}}</strong>
        </mat-hint>
        <mat-hint align="end">
          {{loginEl.value.length}} / {{Api.Login.maxLength}}
        </mat-hint>
      </mat-form-field>
    </form>
  </mat-card-content>
</mat-card>

```


```

<mat-form-field>
  <input #passwordEl matInput
    [(ngModel)]="credentials.password"
    [pattern]="Api.Password.regex"
    type="password"
    placeholder="Password"
    name="password"
    required
  >
  <mat-hint align="start">
    <strong>{{Api.Password.formatHint}}</strong>
  </mat-hint>
  <mat-hint align="end">
    {{passwordEl.value.length}} / {{Api.Password.maxLength}}
  </mat-hint>
</mat-form-field>
<br>
<mat-form-field>
  <input #repeatPasswordEl matInput ngModel
    pattern="^{{escapeRegExp(credentials.password)}}$"
    type="text"
    placeholder="Repeat password"
    name="password-repeat"
    required
  >
  <mat-hint align="end">
    {{repeatPasswordEl.value.length}} / {{passwordEl.value.length}}
  </mat-hint>
</mat-form-field>
<br>
<button
  #submitButton
  mat-raised-button
  color="primary"
  type="submit"
  [disabled]="registerForm.invalid"
  ><mat-icon>how_to_reg</mat-icon>Submit</button>
</form>
</mat-card-content>
</mat-card>



```

Приклади результатів

Сторінка з посиланнями на пісочницю та документацію API.



API reference

 veetaha

GraphQL API

[▶ Try in playground](#) [🔗 Browse api documentation](#)

Authorization is provided via JWT.

Login

Do a RESTful **POST** request on `/auth/login` with the body, which contains your credentials.

```
{  login: "login_string",  password: "password_string"}
```

Registration

Do a RESTful **POST** request on `/auth/register` with the body, which contains your new credentials and fullname.

```
{  login: "login_string",  password: "password_string",  fullname: "fullname_string"}
```

Сторінка з документацією API.

GraphQL Schema

SCHEMA

Query

Mutation

SCALARS

BOID

Boolean

Date

Float

Int

String

ENUMS

TaskType

UserRole

__DirectiveLocation

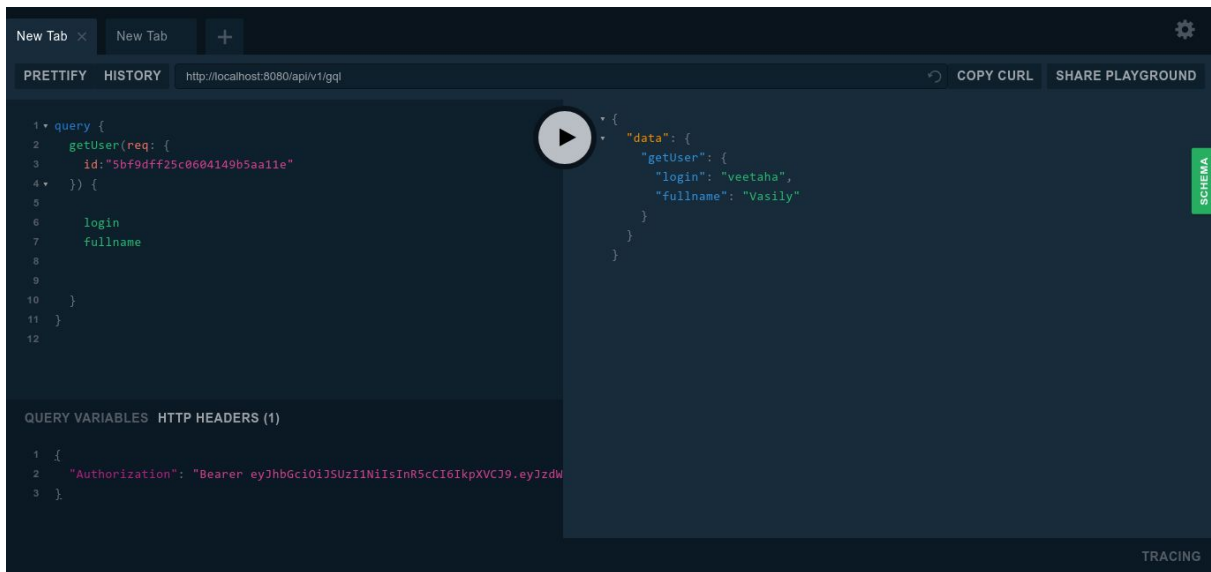
Graphql schema documentation

GraphQL Schema definition

```
1. schema {
2.
3.   # Root queries endpoint.
4.   query: Query
5.
6.   # Root mutations endpoint
7.   mutation: Mutation
8. }
```

GENERATED WITH GRAPHDOC 2.4.0

Сторінка пісочниці



Висновки

В результаті виконання даної лабораторної роботи я реалізував JSON REST API для доступу до ресурсів веб-сервера.