**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**

**“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”**

**ТЕПЛОЕНЕРГЕТИЧНИЙ ФАКУЛЬТЕТ**

**КАФЕДРА АВТОМАТИЗАЦІЇ ПРОЕКТУВАННЯ ЕНЕРГЕТИЧНИХ ПРОЦЕСІВ І СИСТЕМ**

**КУРСОВА РОБОТА**

З кредитного модулю:

**Компоненти програмної інженерії 3.**

на тему: **система шкільного обліку класів та учнів**

Студента III курсу, групи ТІ-92

**Черноусова Дениса Ігоровича**

Спеціальність 121 – Інженерія програмного забезпечення

Освітня програма Інженерія програмного забезпечення інтелектуальних кібер-фізичних систем і веб-технологій

Керівник доцент, к.т.н., Д. С. Смаковський

Національна оцінка\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_\_ оцінка: ECTS\_\_\_\_\_\_

Члени

комісії: \_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(вчене звання, науковий ступінь, прізвище та ініціали)

\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(вчене звання, науковий ступінь, прізвище та ініціали)

Засвідчую, що у цій курсовій роботі немає запозичень праць інших авторів без відповідних посилань

Студент \_\_\_\_\_\_\_\_\_\_ Д. І. Черноусов

**КИЇВ 2021**

|  |
| --- |
| **НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**  **“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”**  **ТЕПЛОЕНЕРГЕТИЧНИЙ ФАКУЛЬТЕТ**  **КАФЕДРА АВТОМАТИЗАЦІЇ ПРОЕКТУВАННЯ ЕНЕРГЕТИЧНИХ ПРОЦЕСІВ І**  **СИСТЕМ**        **ПОЯСНЮВАЛЬНА ЗАПИСКА**    до курсової роботи на тему:    **система шкільного обліку класів та учнів**    Спеціальність 121 – Інженерія програмного забезпечення  Освітня програма Інженерія програмного забезпечення інтелектуальних кібер-фізичних систем і веб-технологій    з кредитного модулю:  «Компоненти програмної інженерії 3. Курсова робота»      Виконав студент групи ТІ-92 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Д. І. Черноусов  (підпис, дата)    Керівник роботи, к.т.н, доц. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Д. С. Смаковський  (підпис, дата)      Київ 2021 |

ЗМІСТ

[ВСТУП 4](#_Toc1)

[1. ПОСТАНОВКА ЗАВДАННЯ 5](#_Toc2)

[2 ЗАГАЛЬНА АРХІТЕКТУРА ВЕБ-СИСТЕМИ 6](#_Toc3)

[2.1 Загальний принцип роботи 6](#_Toc4)

[2.2 Загальний опис сайту 6](#_Toc5)

[3 ОПИС ЗАСОБІВ ВИКОНАННЯ РОБОТИ 8](#_Toc6)

[3.1 Java & Spring 8](#_Toc7)

[3.2 Bootstrap 8](#_Toc8)

[3.3 Intellij idea ultimate edition 9](#_Toc9)

[4 ОПИС ПРОЦЕСУ ВИКОНАННЯ ТА РЕЗУЛЬТАТІВ 10](#_Toc10)

[4.1 База даних 10](#_Toc11)

[4.2 Бекенд 11](#_Toc12)

[4.3 Фронтенд 11](#_Toc13)

[ВИСНОВКИ 12](#_Toc14)

[ЛІТЕРАТУРА 13](#_Toc15)

[Додаток 14](#_Toc16)

# ВСТУП

На сьогодні журнали в шкільних закладах є достатньо застарілим засобом обліком учнів та викладачів. На заміну приходять електроні журнали, щоденники та розклади, які декілька раз безпечніші та зручніші за традиційні методи. В Україні нещодавно у зв’язку з пандемією COVID-19 розпочали активно впроваджувати подібні технології в державних навчальних закладах. Проте в приватних школах електроні журнали використовуються вже давно. Серед найпопулярніших програмних продуктів можна зустріти : e-journal та e-schools.

Зазвичай такі інтелектуальні системи включають в себе :

* Електроні журнали. Заклади можуть створювати документи тимчасового (до 10 років включно) строку зберігання (у тому числі класний журнал) в електронній формі
* Електроні щоденники. Функціонал дозволяє учасникам освітнього процесу за допомогою зручного онлайн інструменту мати постійний доступ до цифрового аналога звичайного щоденника
* Електронний розклад. Зручний доступ до актуального переліку уроків в режимі онлайн з мобільних пристроїв або персонального комп'ютера
* Освітня статистика. Використання індивідуальних даних учнів та педагогічних працівників для аналізу та покращення навчального процесу
* Захист системи. Дані системи захищені від доступу третіх лиць, що підтверджується атестатом відповідності комплексної системи захисту інформації

# 1. ПОСТАНОВКА ЗАВДАННЯ

Завдання веб-системи полягає в тому, щоб надати користувачу доступ до зручного, добре оформленого, правильно та стабільно працюючої й красивої електронної системи обліку. В системі повинні підтримуватися режими обліку класів, учнів в них, викладачів, реєстрації нового учня або вчителя, обліку відвідування занять і оцінок учнів, генерації звітів по успішності учнів.

База даних повинна містити близько 8 таблиць, у вигляді 3 нормальної форми. Створити скрипт і саму базу даних, продумати, які індекси необхідні для пошуків та обмеження на поля. Завантажити невеликою кількістю тестових даних відповідно до предметної області та заповнити таблиці-словники.

Для обраної предметної області потрібно створити REST контролери із набором методів GET, POST, PUT, DELETE для таблиці з учнями, вчителями, розкладами, предметами, класами, записами в журналі . Web-додаток розроблятиметься як набір веб-сторінок, згенерованих на сервері, для відображення списків об’єктів, їх редагування, видалення та створення нових.

# 2 ЗАГАЛЬНА АРХІТЕКТУРА ВЕБ-СИСТЕМИ

## 2.1 Загальний принцип роботи

Веб система складається з трьох основних блоків : власне сайту, де розташовані сторінки з певним наповненням, дизайном, частковою адаптивністю та валідацією частини даних, що можна описати як frontend. База даних написана на mysql, що містить інформацію про учнів, класи, викладачів, предметів, оцінки, розклад, уроки. Посередником між базою даних та фронтендом є бекенд, що відповідає за логіку веб- додатку. В ньому прописані перевірки вхідних даних, класи та методи, що необхідні для цієї конкретної предметної області.

В кінцевому результаті отримуємо схему, зображену на рисунку 2.1.1

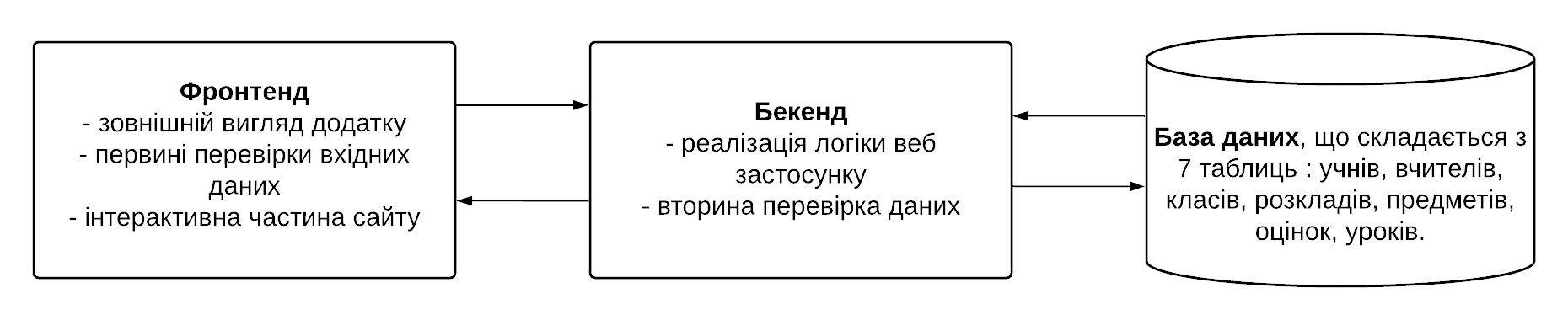


Рисунок 2.1.1 – Загальна схема роботи веб-системи

## 2.2 Загальний опис сайту

На сайті користувачу доступні головна сторінка, сторінки з таблицями : учнів, викладачів, класів, розкладів, оцінок та предметів. При цьому перехід між цими сторінками можна переходити за допомогою посилань у хедері, що знаходиться на кожній сторінці. Усі сторінки з таблицями включають посилання на дві додаткові сторінки для додавання нового та оновлення рядку. Видалення рядків відбувається на тій ж сторінці в таблиці.

Схема навігації по сайту та логіки сторінки зображена на рисунку 2.2.1.



Рисунок 2.2.1 – Навігація по сайту та логіка сторінки

Головна сторінка містить вікно вітання. Через хедер будь якої сторінки можна перейти на інші сторінки, окрім сторінок типу Update та Create, які створені для кожної окремої сторінки-таблиці. В цих сторінках можна заповнити дані на запит в базу даних, тобто форму. Вони перевіряють коректність даних за допомогою різних інструментів, наприклад, regex. Далі запит перевіряється згідного логіці програми на бекенді.

# 3 ОПИС ЗАСОБІВ ВИКОНАННЯ РОБОТИ

## 3.1 Java & Spring

Spring Framework — це java фреймворк, створений на основі інверсії контролю контейнерів. Фреймворк не передбачає конкретної моделі програмування, проте він став популярним у спільноті Java як доповнення до моделі Enterprise JavaBeans (EJB).  
 Spring робить програмування на мові Java швидшим, простішим і безпечнішим для всіх. Зосередженість Spring на швидкості, простоті та продуктивності зробила його найпопулярнішим фреймворком Java у світі.

Гнучким бібліотекам Spring довіряють розробники з усього світу. Spring забезпечує чудові враження мільйонам кінцевих користувачів щодня — будь то потокове телебачення, інтернет-магазини чи безліч інших інноваційних рішень. Spring також має внесок від усіх великих імен у сфері технологій, такі як Yandex, Amazon, Google, Microsoft тощо.

В проекті цей фреймворк займає ключове місце в побудові логіки веб-додатку - бекунду. За допомогою нього, а конкретно залежності Spring Data Jpa, виконуються усі запити до бази даних.

## 3.2 Bootstrap

Bootstrap — це фреймворк CSS з відкритим вихідним кодом, спрямований на веб-розробку графічних інтерфейсів. Він містить шаблони дизайну на основі CSS для кнопок, форм, навігації та інших компонентів інтерфейсу.  
 Результатом є уніфікований вигляд прози, таблиць і елементів форм у всіх веб-браузерах. Крім того, розробники можуть скористатися перевагами класів CSS, визначених у Bootstrap, щоб додатково налаштувати зовнішній вигляд вмісту. Наприклад, у Bootstrap передбачені таблиці різних кольорів, заголовки сторінок, більші лапки та текст із виділенням.

## 3.3 Intellij idea ultimate edition

IntelliJ IDEA Ultimate edition — це інтегроване середовище розробки, що написане на Java для розробки комп’ютерного програмного забезпечення на Java та інших мовах.

IntelliJ надає певні переваги, такі як завершення коду шляхом аналізу контексту, навігація по коду, яка дозволяє безпосередньо переходити до класу або оголошення в коді, рефакторинг коду, налагодження коду та виправлення невідповідностей через пропозиції.  
 IntelliJ забезпечує інтеграцію з інструментами збірки та пакування, такими як maven, gradle і SBT. Він підтримує системи контролю версій Git, Perforce і SVN. Можна отримати доступ у версії Ultimate до таких баз даних, як NoSQL, Oracle, PostgreSQL, SQLite та MySQL.

IntelliJ підтримує плагіни, за допомогою яких можна додати додаткові функції до програмного середовища. Плагіни можна завантажити та встановити з веб-сайту сховища плагінів IntelliJ, а також через вбудовану функцію пошуку та встановлення плагінів. Кожне видання має своє сховище плагінів, і станом на 2020 рік у виданнях Community та Ultimate є понад 3000 плагінів.

## 3.4 Thymeleaf

Thymeleaf - це механізм шаблонізації XML/HTML/HTML5 файлів, здатний застосувати набір перетворень у файли шаблонів для відображення даних та тексту, створеного програмами.

Він краще підходить для обслуговування XHTML/HTML5 у веб-застосунках, але може обробляти будь-який файл XML, будь то в Інтернеті або в автономних додатках.

Основна мета Thymeleaf – надати елегантний та добре сформований спосіб створення шаблонів. Щоб досягти цього, він заснований на тегах і атрибутах XML, що визначають виконання попередньо визначеної логіки в DOM (об'єктна модель документа), замість явно записувати цю логіку як код всередині шаблону.

Завдяки модулям для Spring Framework, Thymeleaf ідеально підходить для сучасної веб-розробки HTML5 JVM.

# 4 ОПИС ПРОЦЕСУ ВИКОНАННЯ ТА РЕЗУЛЬТАТІВ

## 4.1 База даних

База даних складається з 7 таблиць (див. рисунок 4.1.1). Було прийняте рішення не накладати обмеження на саму базу даних оскільки валідацію даних можна здійснити на фронтенді та бекенді, що значно прискорює роботу веб-додатку.

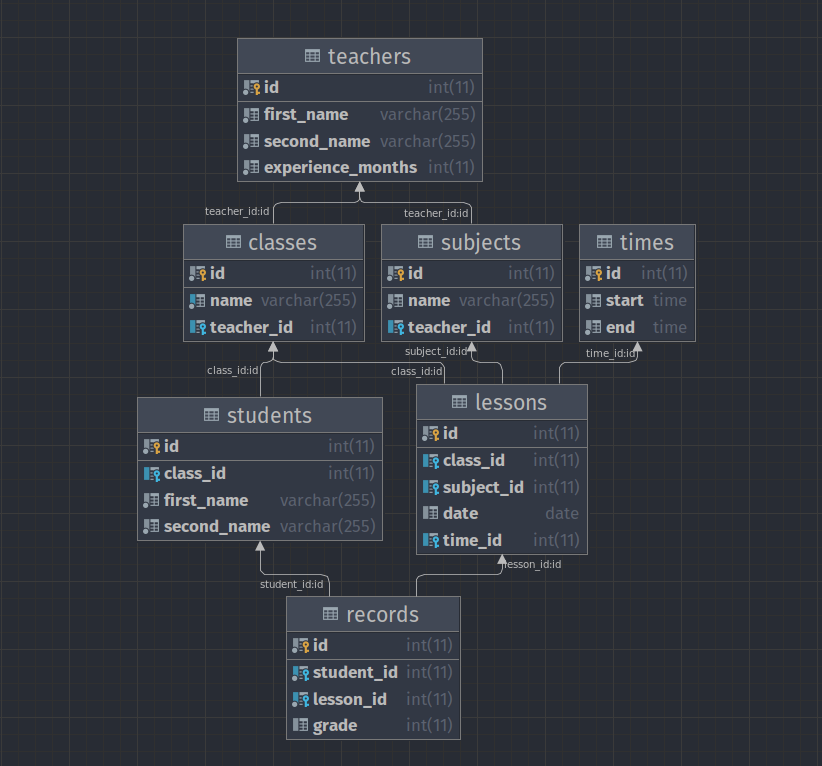


Рисунок 4.1.1 Модель бази даних

Таблиця teachers містить дані про викладачів, а саме ім’я, прізвище, що в парі мають бути унікальними, та досвід роботи в місяцях.

Таблиця classes містить дані про клас, тобто назву, що є унікальною для таблиці, та посилається через поле teacher\_id на викладача, який відповідальний за цей клас.

Таблиця subjects містить дані про предмети, наприклад математику або біологію, та посилання на викладача, що проводить даний предмет. Id викладача та назва предмету повинні бути унікальними для таблиці.

Таблиця times є незмінною таблицею-словником, тобто дані, що знаходяться в ній не можливо змінити за допомогою веб-додатку, лише за допомогою прямого втручання в базу даних через mysql. Таблиця містить дані про номери, початки та кінці уроків.

Таблиця students містить дані про учнів. Ім’я та прізвище мають бути унікальною парою. Поле class\_id відповідає за зв’язок учня з класом, до якого він належить.

Таблиця lessons містить дані про уроки, а саме клас, для якого вони проводитимуться, предмет, який викладатиметься, номер уроку та дату проведення цього уроку. Клас, дата та номер уроку в купі мають бути унікальними.

Таблиця records є, по суті, журналом, в якому записуються оцінки учнів на уроках. Пара полів id студента та уроку повинні разом бути унікальними. Оцінка варіюється між числами 0 та 12 включно, де 0 вважається знаком відсутності. Ця логіка прописана в бекенді.

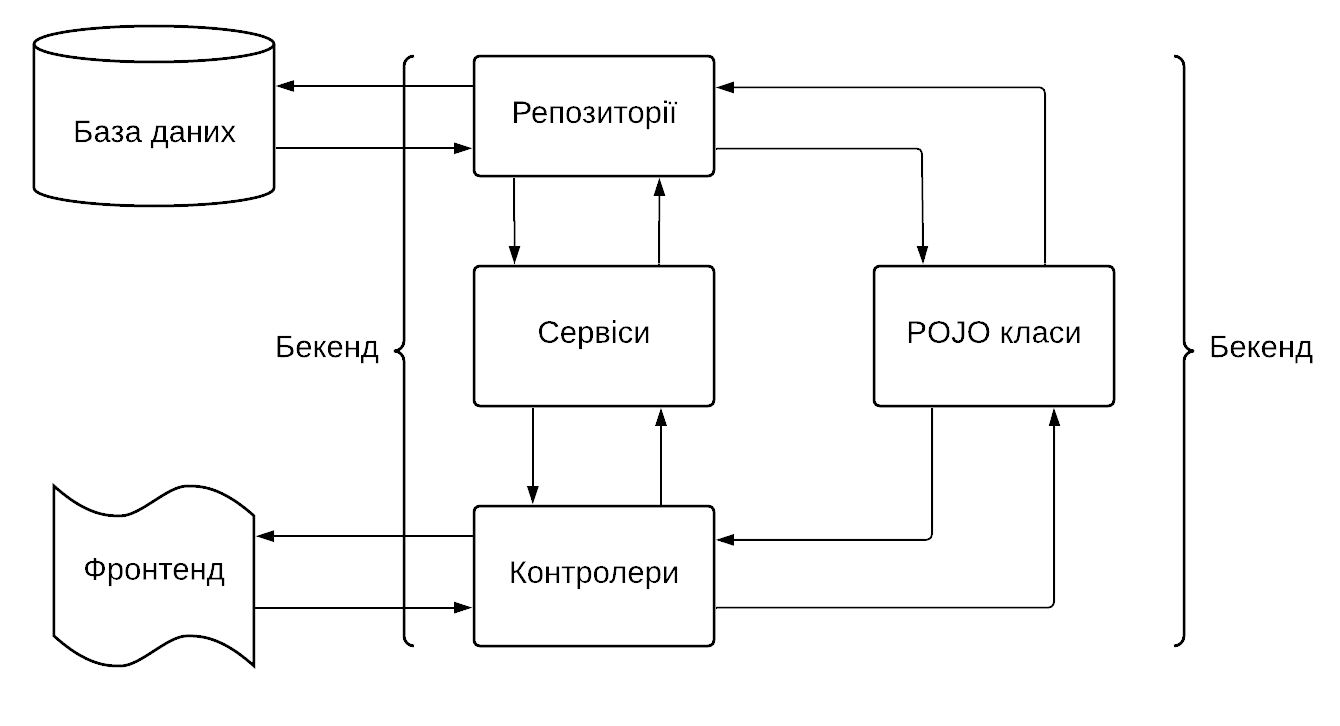
## 4.2 Бекенд

Бекенд складається з 4 шарів (див. рисунок 4.2.1)

Репозиторії - це шар інтерфейсів, що наслідують інтерфейс JpaRepository. Цей шар відповідає за прямий контакт з базою даних через стандартні sql команди, а також на цьому шарі були створені власні команди у виглядів полів інтерфейсу.

Шар Сервіси містить класи-сервіси, що використовують інтерфейси-репозиторії для виконання запитів. Вони є посередниками між репозиторіями та контролерами. Зазвичай саме в них виконується бізнес-логіка.

Рисунок 4.2.1 Структура бекенду у вигляді шарів

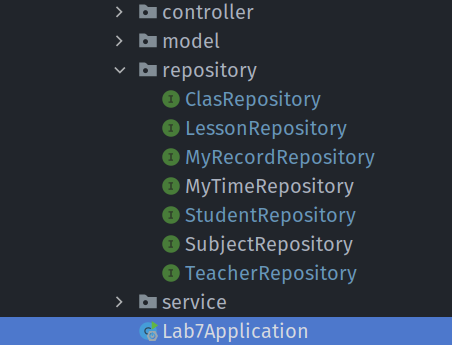


POJO класи - це сутності бази даних. Тобто такий клас представляє собою таблицю відповідну таблицю. Очевидно, що клас складається з таких самих полів, що й таблиця та методів типу getter та setter для кожного поля. Для позначення взаємозв’язку класів і таблиць використовуються анотації. Вони також використовуються в бібліотеці Lombok для атоматичної генерації тих самих необхідних методів getters та setters. Зв’язки таблиць через зовнішні ключі, наприклад багато до одного або багато до багатьох , теж описуються за допомогою анотацій. Можливо проводити валідацію даних в полях, проте в нашому веб-застосунку вигідніше робити це через фронтенд.

Контролери - це класи-елементи шаблону MVC. Їхня головна роль це - маршрутизація POJO класу (Model) до відповідного шаблону або ще як називають - виду (View). За допомогою контролерів відбувається перехід між сторінками на сайті (запити типу GET) та здійснюється зміна, додавання та видалення даних (методи типу POST). Саме в контролерах відбувається вторинна валідація даних, а саме перевіряється чи існує ідентичний об’єкт, що був переданий через фронтенд, у базі даних. Однаковими вважаються ті рядки, де унікальні поля або сукупність унікальних полів ( залежить від таблиці див. розділ 4.1 ) уже існує.

У Intellij idea вище наведені шари мають такий вигляд (див. Рисунок 4.2.1). Клас Lab7Application - це точка входу додатка.

Рисунок 4.2.1 Структура бекенду в пакетному вигляді



4.3 Фронтенд

Головна сторінка, як й інші, складаються з хедера з посиланнями на сторінки-таблиці, тіла та однакового футера ( Рисунок 4.3.1 ).

Сторінки-таблиці (див. Рисунок 4.3.2 ) містять таблиці з даними. Рядки в таблиці можна динамічно фільтрувати. Дані можна видаляти, оновлювати та додавати перейшовши за посиланням. Для кожної сторінки-таблиці існують свої сторінки типу create ( Рисунок 4.3.3 ) та update ( Рисунок 4.3.4 ) для відповідних POST запитів. Саме на цих сторінках відбувається первинна валідація даних, а саме перевірка чи ім’я відповідає regex або чи число входить в допустимий інтервал значень. ( Рисунок 4.3.5 )

При неправильно введеній адресі, відбувається перехід на сторінку помилки. ( Рисунок 4.3.6 ).

Рисунок 4.2.1 Головна сторінка

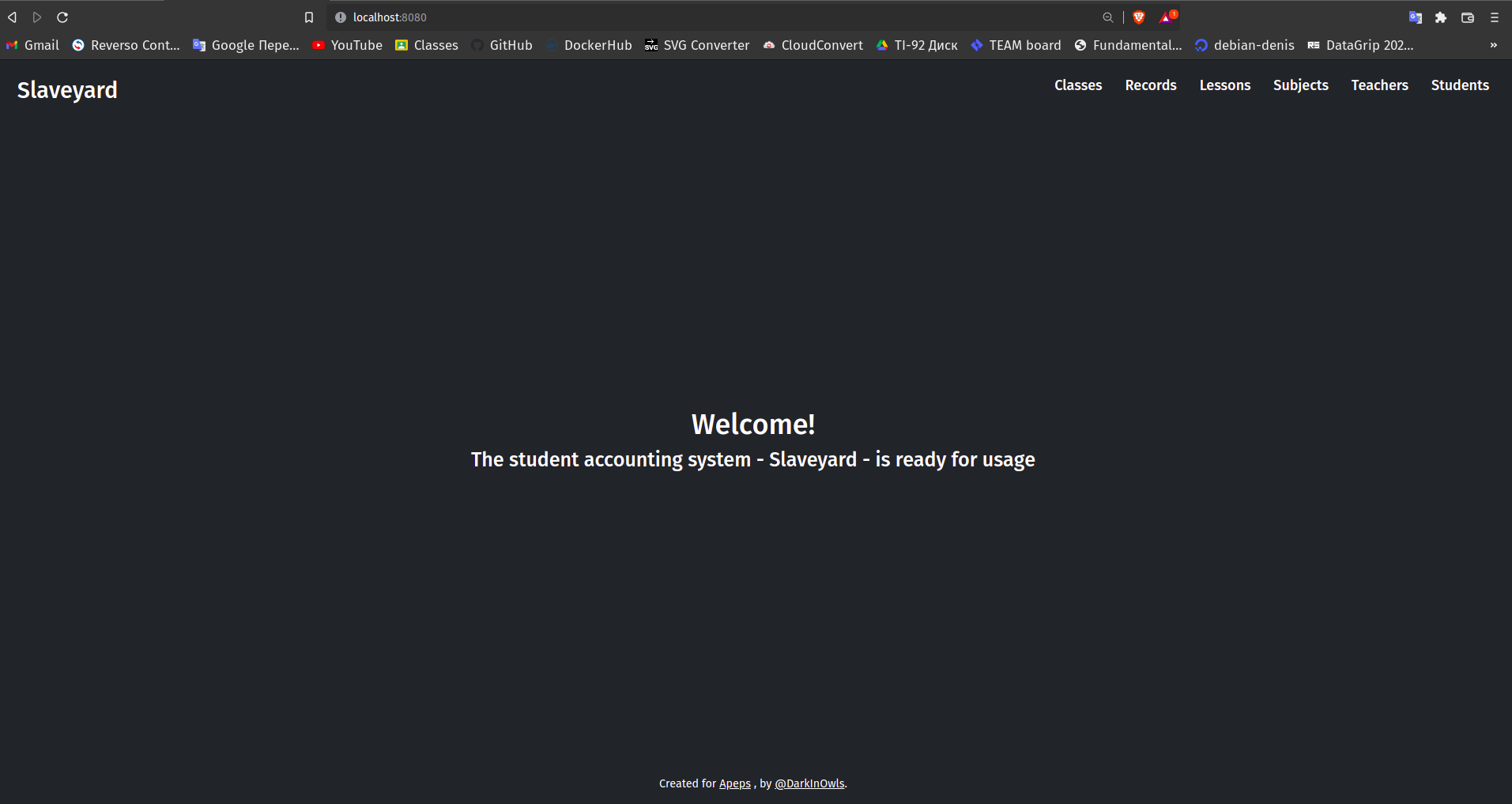


Рисунок 4.2.2 Сторінка-таблиця

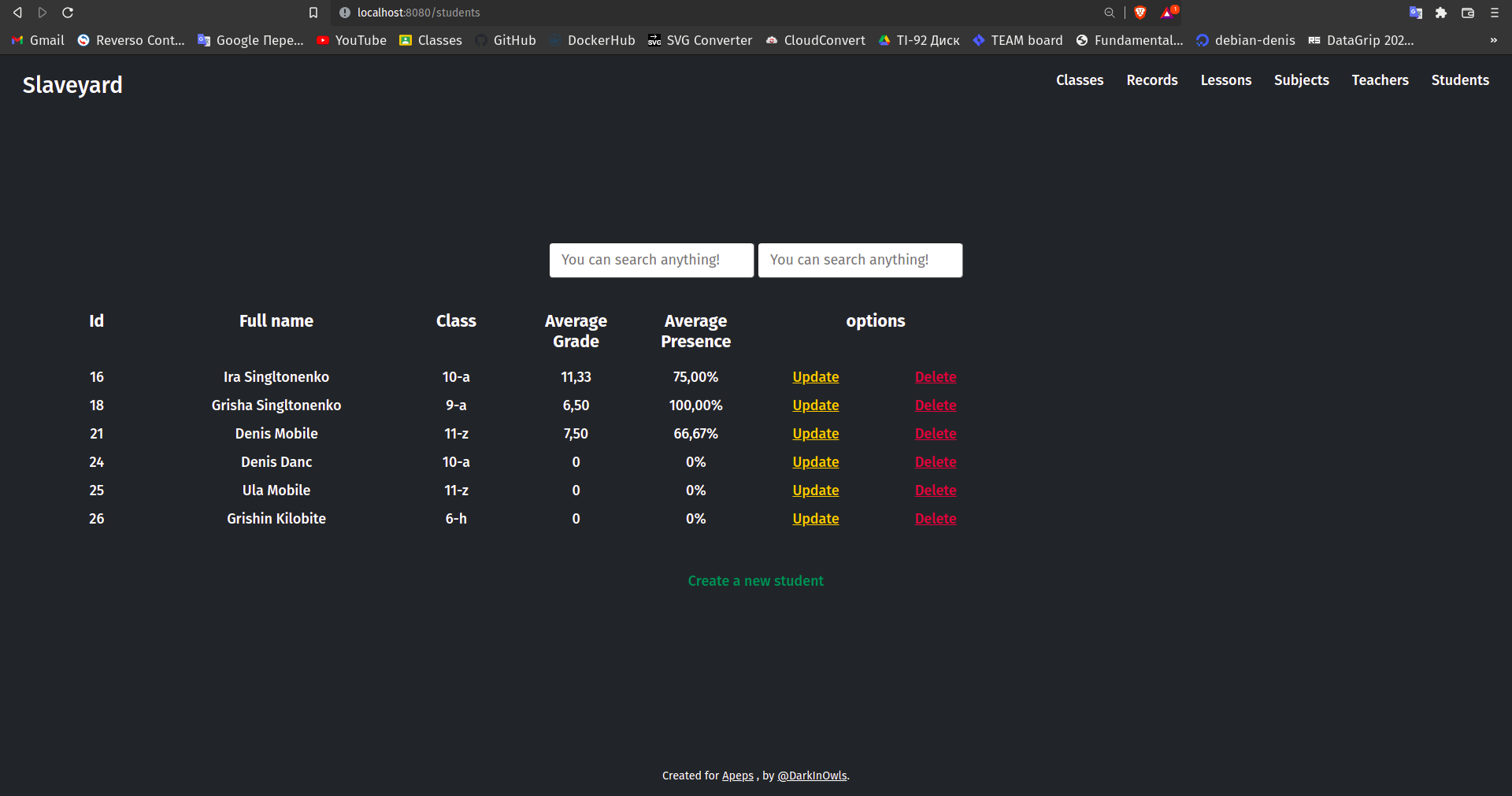


Рисунок 4.2.3 Сторінка типу create

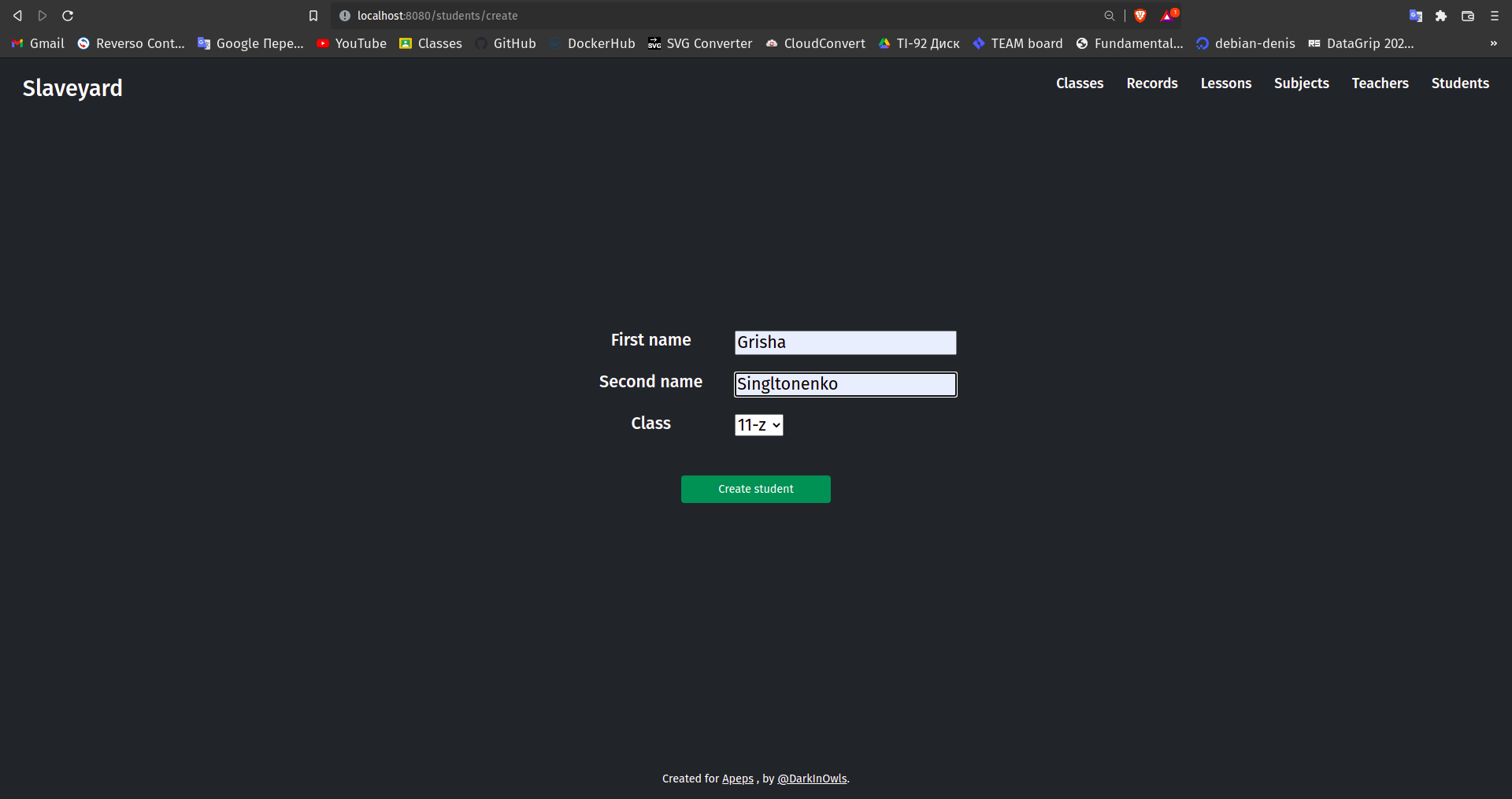


Рисунок 4.2.4 Сторінка типу update

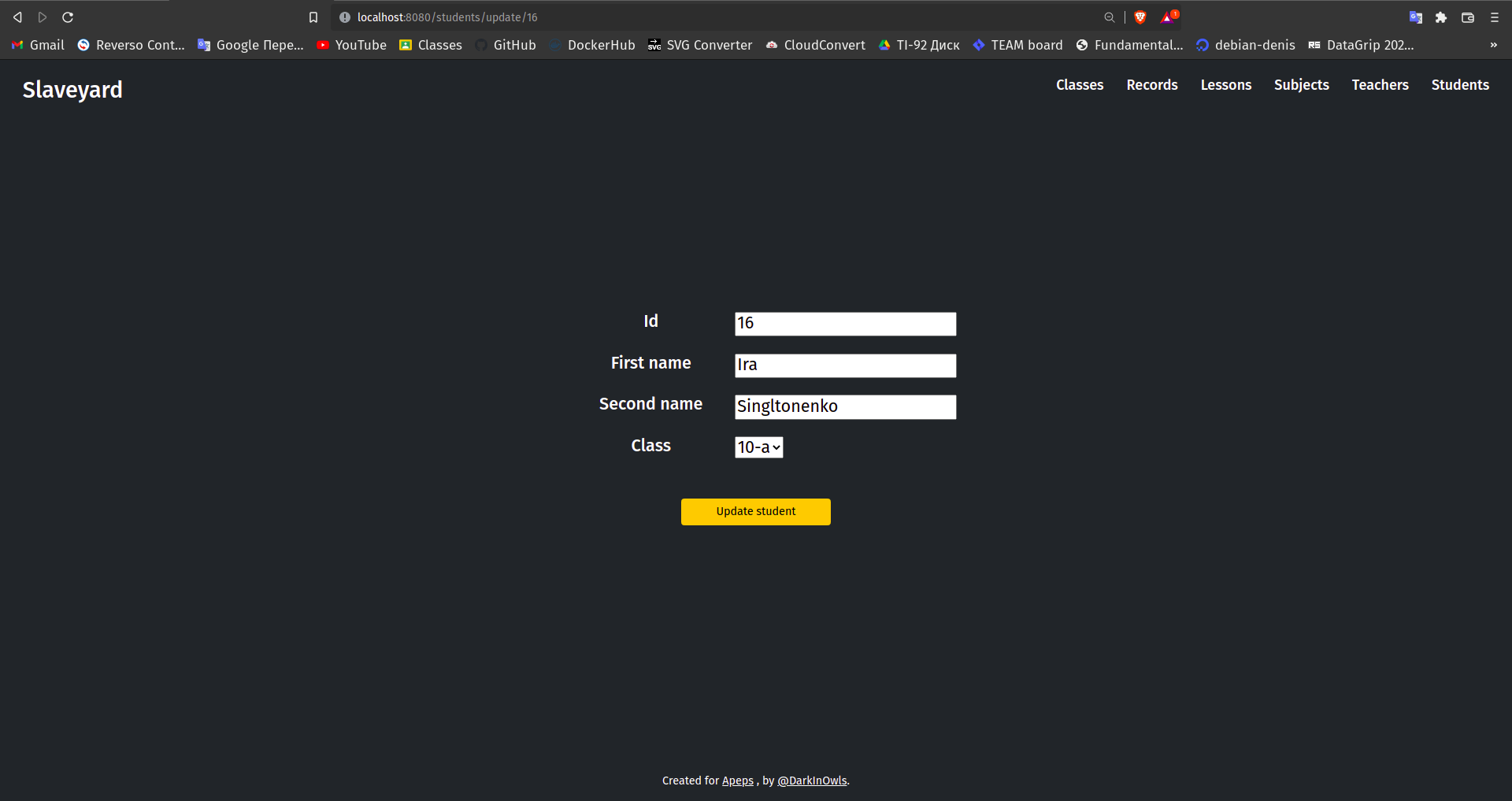


Рисунок 4.2.5 Первинна валідація даних

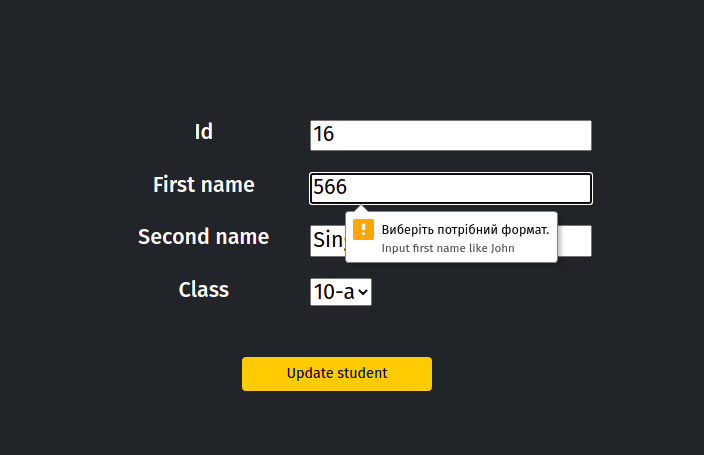
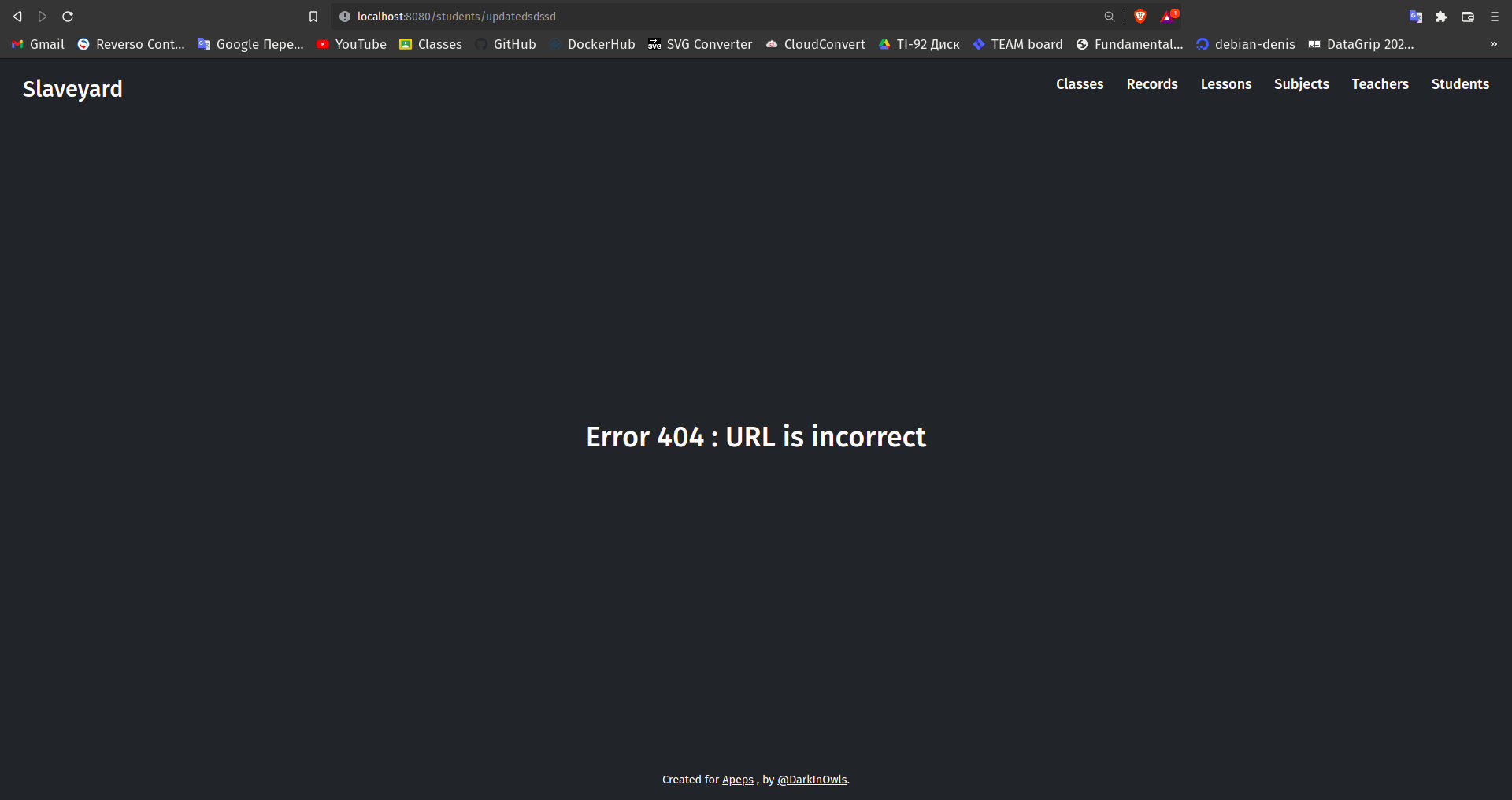


Рисунок 4.2.6 Сторінка помилки



# ВИСНОВКИ

В ході виконання курсової роботи було отримано інформацію про принципи роботи електронних журналів, запити в них, обробку та відображення інформації. Була створена власна база даних для проекту, яка через відсутність точної моделі проекту, змінювалась багато разів. Були отримані додаткові знання про можливості spring, java, thymeleaf, bootstrap, сss, mysql, javascript, html, lombok та jpaRepository .

Були стоворені REST контролери із набором методів GET, POST, PUT, DELETE для таблиці з учнями, вчителями, розкладами, предметами, класами, записами в журналі. Web-додаток розроблений як набір веб-сторінок, згенерованих на сервері, для відображення списків об’єктів-рядків, їх редагування, видалення та створення нових.

Певні навички були отримані і в пошуку помилок на сайті, що дозволяло швидко знаходити ситуації, в яких сайт функціонує неправильно. Дуже корисним було написання веб-системи від початку до кінця, що дало змогу попрацювати з кожним етапом і спостерігати, як ці етапи впливають на проект в цілому.

Зовнішній вигляд сайту є досить красивим, а основний функціонал - реалізований. Що важливо, база даних потребувала більше затрат часу та точного плану розробки, щоб мати хорошу архітектуру, з якою не виникало би пустих або зайвих комірок.

# ЛІТЕРАТУРА

1. <https://docs.oracle.com/en/java/>
2. <https://docs.spring.io/spring-framework/docs/current/reference/html/>
3. <https://www.thymeleaf.org/documentation.html>
4. <https://getbootstrap.com/docs/5.1/getting-started/introduction/>
5. <https://dev.mysql.com/doc/>
6. <https://stackoverflow.com/>
7. Abeysinghe S. RESTful Java Web Services / Samisa Abeysinghe., 2008. – 222 с. – (Kindle Edition).
8. Walls C. Spring in Action / Craig Walls., - 2018. - 520 c. - (5th Edition)

# Додаток

Життєвий цикл таблиці students, моделі Student

StudentRepository.java

package com.lab7.repository;

import com.lab7.model.Lesson;

import com.lab7.model.MyRecord;

import com.lab7.model.Student;

import org.springframework.data.jpa.repository.JpaRepository;

public interface StudentRepository extends JpaRepository<Student,Integer> {

Student getByFirstNameAndSecondName(String firstName, String secondName);

Boolean existsByFirstNameAndSecondName(String firstName, String secondName);

}

StudentService.java

package com.lab7.service;

import com.lab7.model.Student;

import com.lab7.repository.StudentRepository;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Component;

import java.util.List;

@Component

public class StudentService {

private final StudentRepository studentRepository;

@Autowired

public StudentService(StudentRepository studentRepository) {

this.studentRepository = studentRepository;

}

public Student findById(int id) {

return studentRepository.findById(id).orElse(null);

}

public List<Student> findAll() {

return studentRepository.findAll();

}

public Student saveStudent(Student student) {

return studentRepository.save(student);

}

public void deleteById(int id) {

studentRepository.deleteById(id);

}

public Boolean existsByFirstNameAndSecondName(String firstName, String secondName){

return studentRepository.existsByFirstNameAndSecondName(firstName, secondName);

}

public Student getByFirstNameAndSecondName(String firstName, String secondName){

return studentRepository.getByFirstNameAndSecondName(firstName, secondName);

}

}

Student.java

package com.lab7.model;

import lombok.AllArgsConstructor;

import lombok.Getter;

import lombok.NoArgsConstructor;

import lombok.Setter;

import javax.persistence.\*;

import java.util.List;

@Getter

@Setter

@Entity

@AllArgsConstructor

@NoArgsConstructor

@Table(name = "students",

uniqueConstraints = {@UniqueConstraint(columnNames = {"first\_name", "second\_name"})})

public class Student {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int id;

@Column(name = "first\_name")

private String firstName;

@Column(name = "second\_name")

private String secondName;

@ManyToOne

@JoinColumn(name = "class\_id")

private Clas clas;

@OneToMany(mappedBy = "student")

private List<MyRecord> myRecords;

public String calculateGrade() {

float size = 0;

float sum = 0;

for (MyRecord r : myRecords) {

if (r.getGrade() > 0) {

sum += r.getGrade();

size++;

}

}

if (size == 0) return "0";

return String.format("%.2f", sum / size);

}

public String calculatePresence() {

float all = 0;

float absence = 0;

for (MyRecord r : myRecords) {

all++;

if (r.getGrade() == 0) {

absence++;

}

}

if (all == 0 || (all - absence) == 0 ) return "0";

return String.format("%.2f", (all - absence) / all \* 100 );

}

}

StudentController.java

package com.lab7.controller;

import com.lab7.model.Clas;

import com.lab7.model.MyRecord;

import com.lab7.model.Student;

import com.lab7.service.ClasService;

import com.lab7.service.StudentService;

import lombok.AllArgsConstructor;

import lombok.Getter;

import lombok.Setter;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.validation.BindingResult;

import org.springframework.validation.annotation.Validated;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.RequestBody;

import javax.validation.Valid;

import java.util.List;

import java.util.Objects;

@Controller

public class StudentController {

private final String REDIRECT\_STUDENTS = "redirect:/students";

private final StudentService studentService;

private final ClasService clasService;

@Autowired

public StudentController(StudentService studentService, ClasService clasService) {

this.studentService = studentService;

this.clasService = clasService;

}

@GetMapping("/students")

public String findAll(Model model) {

List<Student> students = studentService.findAll();

model.addAttribute("students", students);

return "students/students";

}

@GetMapping("/students/create")

public String createStudentForm(Student student, Model model) {

List<Clas> classes = clasService.findAll();

model.addAttribute("classes", classes);

return "students/create";

}

@PostMapping("/students/create")

public String createStudent(Student student, Model model) {

if (Boolean.TRUE.equals(

studentService.existsByFirstNameAndSecondName(student.getFirstName(), student.getSecondName()))) {

model.addAttribute("exist", true);

return createStudentForm(student, model);

}

studentService.saveStudent(student);

return REDIRECT\_STUDENTS;

}

@GetMapping("/students/delete/{id}")

public String deleteStudent(@PathVariable("id") int id) {

studentService.deleteById(id);

return REDIRECT\_STUDENTS;

}

@GetMapping("/students/update/{id}")

public String updateStudentForm(@PathVariable("id") int id, Model model) {

Student student = studentService.findById(id);

model.addAttribute("student", student);

List<Clas> classes = clasService.findAll();

model.addAttribute("classes", classes);

return "students/update";

}

@PostMapping("/students/update")

public String updateStudent(Student student, Model model) {

Student self = studentService.getByFirstNameAndSecondName(student.getFirstName(), student.getSecondName());

if (self == null || (self.getId() == student.getId() && self.getClas().getId() != student.getClas().getId())) {

studentService.saveStudent(student);

return REDIRECT\_STUDENTS;

}

model.addAttribute("exist", true);

return updateStudentForm(student.getId(), model);

}

}

Stydents.html

<html lang="en" class="">

<head>

<meta charset="utf-8">

<title>Slaveyard</title>

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet"

integrity="sha384-1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3" crossorigin="anonymous">

<link rel="stylesheet" type="text/css" th:href="@{/css/style.css}"/>

</head>

<body class="d-flex h-100 text-center text-white bg-dark">

<div class="cover-container d-flex w-100 h-100 p-3 mx-auto flex-column">

<header th:insert="/components/header :: header"></header>

<main class="m-auto">

<form action="#" th:action="@{/students/create}" th:object="${student}" method="post">

<table>

<div class="row mb-3">

<h4 class="col-5 themed-grid-col">First name</h4>

<h4 class="col-5 themed-grid-col">

<input type="text" th:field="\*{firstName}"

pattern="^[A-Z][a-z]{2,50}$" placeholder="First name"

title="Input first name like John" required>

</h4>

</div>

<div class="row mb-3">

<h4 class="col-5 themed-grid-col">Second name</h4>

<h4 class="col-5 themed-grid-col">

<input type="text" th:field="\*{secondName}"

pattern="^[A-Z][a-z]{2,50}$" placeholder="Second name"

title="Input second name like Smith" required>

</h4>

</div>

<div class="row mb-3">

<h4 class="col-5 themed-grid-col">Class</h4>

<h4 class="col-1 themed-grid-col">

<select th:field="\*{clas.id}">

<!-- <option value="1">select class</option>-->

<option th:each="clas : ${classes}" th:value="${clas.id}"

th:text="${clas.name}"></option>

</select>

</h4>

</div>

<div class="row my-5">

<input type="submit" class="m-auto col-5 btn btn-success" value="Create student">

<p class="text-danger" th:if="${exist}">\* the student already exists</p>

</div>

</table>

</form>

</main>

<footer th:insert="components/footer :: footer"></footer>

</div>

</body>

</html>