НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Кафедра автоматизації енергетичних процесів та систем

(повна назва кафедри)

**КУРСОВА РОБОТА**

з дисципліни **«**Компоненти програмної інженерії 3. Архітектура програмного забезпечення**»**

 (назва дисципліни)

на тему: «Інтелектуальна система шкільного обліку класів та учнів»

Студента курсу  ОС «Бакалавр»

групи

зі спеціальності

121 «Програмна інженерія»

                                                                                                Черноусова Д. І.

(прізвище та ініціали)

Керівник: асистент Беспала О. М.

(посада, науковий ступінь, вчене звання, прізвище та ініціали)

Оцінка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

                                                                                                Кількість балів: \_\_\_\_\_\_\_\_\_\_

               Члени комісії:  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_          к.т.н, доц. Смаковський Д. С.

                                                                                                         (підпис)                         (посада, науковий ступінь, прізвище та ініціали)

Київ - 2021 рік

**Національний технічний університет України**

**“Київський політехнічний інститут імені Ігоря Сікорського”**

Факультет \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ТЕПЛОЕНЕРГЕТИЧНИЙ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кафедра \_Автоматизації проектування енергетичних процесів і систем\_\_\_\_\_

Напрям підготовки \_\_\_\_\_\_\_121 Програмна інженерія\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ЗАВДАННЯ**

**НА КУРСОВУ РОБОТУ СТУДЕНТУ**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Черноусова Дениса Ігорович\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище, ім’я, по батькові)

1.Тема роботи Інтелектуальна система шкільного обліку класів та учнів

керівник курсової роботи – ас. Беспала О. М.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

2. Строк подання студентом роботи 15 грудня 2021 року\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3. Вихідні дані до роботи: мова програмування Java, система керування базами даних MySQL

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) – \_розробка та створення повнофункціонального веб-додатку мовою програмування Java з використанням фреймворку Spring та реляційної бази даних MySQL

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| №  з/п | Назва етапів виконання дипломного  проекту (роботи) | Строк виконання етапів проекту  ( роботи ) | Примітка |
| 1. | Затвердження обраної теми роботи | 20.11.21 |  |
| 2. | Узгодження технічного завдання з замовником | 25.11.21 |  |
| 3. | Створення макетів адаптивного дизайну майбутнього проекту | 27.11.21 |  |
| 4. | Розробка бета-версії веб-додатку | 12.12.21 |  |
| 5. | Налагодження та тестування програми | 17.12.21 |  |
| 6. | Виправлення знайдених помилок | 19.12.21 |  |
| 7. | Оформлення пояснювальної записки | 22.12.21 |  |

**Студент** \_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (прізвище та ініціали)

**Керівник курсової роботи** \_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

АННОТАЦІЯ

Головною метою цієї курсової роботи є створення веб-додатку для обліку учнів у школі та формування звітів про успішність навчання кожного з них.

Для реалізації цієї цілі було створене програмне забезпечення, що керує даними про викладачів, класи, предмети, уроки, записи у журналі та розклади занять.

Програмний продукт створений за допомогою стеку технологій, що складається з Java, Spring, Thymeleaf, Bootstrap, HTML, CSS, JavaScript, MySQL, lombok.

Обсяг пояснювальної записки 36 аркушів, кількість ілюстрацій - 22, 1 додаток.

ANNOTATION

The main purpose of this course work is to create a web application for accounting of students at school and generating reports on the success of each of them.

To achieve this goal, it was created the software that manages data on teachers, classes, subjects, lessons, journal entries and class schedules.

The software product is created using a technology stack consisting of Java, Spring, Thymeleaf, Bootstrap, HTML, CSS, JavaScript, MySQL, lombok.

The volume of the explanatory note is 36 sheets, the number of illustrations is 22, 1 appendix.

ЗМІСТ

[ВСТУП 5](#_Toc2)

[1. ПОСТАНОВКА ЗАВДАННЯ 7](#_Toc3)

[2 ЗАГАЛЬНА АРХІТЕКТУРА ВЕБ-СИСТЕМИ 9](#_Toc4)

[2.1 Загальний принцип роботи 9](#_Toc5)

[2.2 Загальний опис сторінок сайту 9](#_Toc6)

[3 ОПИС ЗАСОБІВ ВИКОНАННЯ РОБОТИ 12](#_Toc7)

[3.1 Spring 12](#_Toc8)

[3.2 Bootstrap 13](#_Toc9)

[3.3 Intellij idea ultimate edition 13](#_Toc10)

[3.4 Thymeleaf 15](#_Toc11)

[4 ОПИС ПРОЦЕСУ ВИКОНАННЯ ТА РЕЗУЛЬТАТІВ 17](#_Toc12)

[4.1 База даних 17](#_Toc13)

[4.2 Серверна частина 18](#_Toc14)

[4.3 Клієнтська частина 22](#_Toc15)

[ВИСНОВКИ 29](#_Toc16)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 30](#_Toc17)

[Додаток 1 31](#_Toc18)

# 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-journa, e-schools, тощо.

Електронні журнали та щоденники це інструменти, які забезпечують базову діджиталізацію системи шкільної освіти, а також безперервну та ефективну взаємодію між чотирма групами учасників процесу навчання – адміністрацією школи, вчителями, учнями та батьками.

Вчитель під час або після уроку вносить дані в електронну систему – теми уроків, оцінки, домашні завдання та посилання на навчальні матеріали.

Учень, використовуючи особистий кабінет за допомогою комп’ютера, смартфона або іншого гаджета, отримує інформацію про власні оцінки та домашні завдання.

Батьки у своєму персональному кабінеті можуть відслідковувати успішність навчання дітей, їх відвідування, контролювати якість виконання завдань та комунікувати напряму з учителем та адміністрацією школи.

Адміністрація школи має можливість аналізувати показники ефективності освітнього процесу в автоматичному режимі без необхідності витягів із паперових носіїв кожного класу та предмету.

Учні отримують такі переваги :

* зручний доступ до навчальних матеріалів і домашніх завдань;
* участь в онлайн-вебінарах і конференціях, організованих школою;
* можливість перегляду матеріалів уроків у зручний час (у т. ч. на період відсутності в школі);
* можливість самостійного контролю успішності навчання.

Для батьків :

* можливість оперативно отримувати інформацію про успішність та відвідування дітей;
* ефективний контроль засвоєння знань та виконання домашніх завдань;

Для педагогів :

* звільнення від надлишкової паперової роботи;
* простий доступ до актуального розкладу у смартфоні або комп'ютері;
* можливість завантажувати навчальні матеріали для ознайомлення та допомоги у підготовці домашніх завдань;
* економія часу для підготовки до уроків;
* зручний поділ класів на групи без паперових журналів;
* просте автоматичне формування складних звітів за підсумками семестру або навчального року;

Для адміністрації школі :

* можливість оперативно готувати освітню звітність, діаграми успішності за класами і предметами;
* аналіз результативності роботи педагогів;
* облік відвідування;
* можливість відслідковувати динаміку успішності учнів, класів, школи;

# 1. ПОСТАНОВКА ЗАВДАННЯ

Завдання веб-системи полягає в тому, щоб надати користувачу доступ до зручного, добре оформленого, стабільної й графічно красивої інтелектуальної шкільної 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.

Був проведений аналіз аналогічних програм, які вже розроблені й мають немалий попит : e-journa та e-schools. Виявлено, що такі інтелектуальні системи зазвичай включають в собі :

* Інтуїтивно зрозумілий інтерфейс. Більша частина користувачів таких програм - це люди, які потребують не тільки простий інтерфейс, а можливо й навіть консультацію, щодо користування веб-додатком.
* Електроні журнали. Можливість створювати документи та журнали тимчасового строку зберігання 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

Для вичерпної реалізації функціональної частини веб-додатку потребується створити базу даних, що містить близько 8 таблиць, у вигляді 3 нормальної форми. Необхідно також 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 проведення тестування.

Для обраної предметної області потрібно створити REST контролери із набором методів GET, POST, PUT, DELETE для таблиць “students”, “teachers”, “lessons”, “subjects”, “classes” та інших. Web-додаток має бути представлений 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-додаток складається з трьох головних частин : клієнтською частиною (фронтендом), серверною частиною (бекендом) та базою даних. Фронтенд - це та сайту, де розташовані сторінки з певним наповненням, дизайном, частковою адаптивністю та валідацією частини даних. База даних розроблена за допомогою мови запитів mysql, що містить інформацію про учнів, класи, викладачів, предметів, оцінки, розклад, уроки. Посередником між базою даних та клієнтською частиною є серверна частина, 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

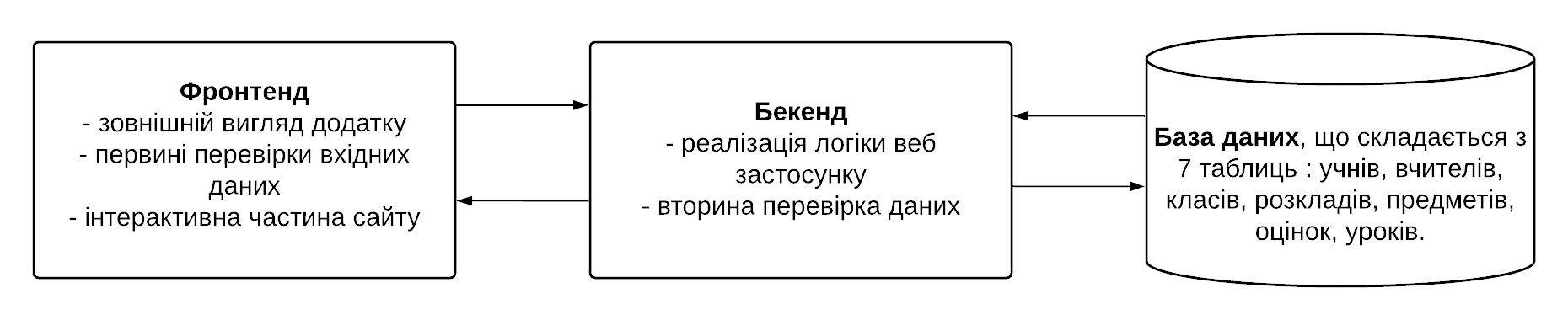


Рисунок 2.1.1 – Загальна схема роботи веб-додатку

## 2.2 Загальний опис сторінок 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 заголовку (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 для взаємодії з відповідними таблицями. На цих двох сторінках можна створити запити на оновлення рядку та створення нового рядку в базу даних, тобто заповнити форми. Вони перевіряють коректність даних за допомогою різних інструментів типу regex, звичайного числового обмеження або через скрипти, що прописані на мові javascript. Помилки висвітлюються одразу.

Такі перевірки на клієнтській покривають більшу частину множину можливих запитів, що йдуть на вхід, проте, наприклад, не забезпечують унікальність рядків у базі даних, що є необхідним для реалізації логіки програми. Цим займається серверна частина. Отримані дані з форми ініціалізуються як об’єкт відповідного класу. Далі відбувається перевірка чи існує такий об’єкт в базі даних.

Залежно від класу відбувається також перевірка на унікальність пари або групи полів. Аби пояснити раціональність використання такої перевірки, доречно навести наступний приклад. У таблиці “students” знаходяться дані про учнів, такі як імена та прізвища. Потрібно, аби кожен учень мав унікальне прізвище та ім’я. Якщо ввести обмеження на ці два поля окремо, то неможливо буде вставити учня з таким самим іменем, але інакшим прізвищем, що є неприпустимою неполадкою. В цій ситуації найкраще застосувати пару унікальних полів, що складатими прізвище та ім’я.

У відповідь на помилку валідації в серверній частині сайту відбувається повернення на попередню сторінку з висвітленням повідомлення помилки. В протилежному випадку зміни додаються до бази даних та відбувається перехід на сторінку-таблицю (див. рисунок 2.2.1).

# 3 ОПИС ЗАСОБІВ ВИКОНАННЯ РОБОТИ

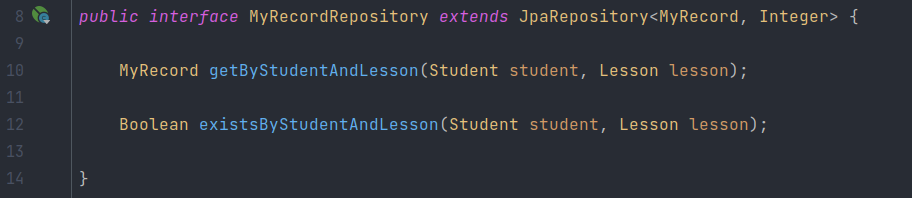
## 3.1 Spring

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.

В проекті цей фреймворк займає ключове місце в побудові логіки веб-додатку - бекенду. За допомогою нього, а конкретно залежності Spring Data Jpa, виконуються усі запити до бази даних. Окрім стандартних запитів, наприклад, на знаходження рядка таблиці даних за її унікальним індексом або на зберігання вибраного об’єкту в базі даних, можливо прописати свої власні у вигляді методів інтерфейсів (див. Рисунок 3.1.1), що не раз прискорювало процес розробки проекту. Такі запити повинні відповідати певним шаблонам, які були задані фреймворком Spring Data Jpa. Для специфічних запитів використовують анотацію @Query

Рисунок 3.1.1 Запити до таблиці “records” у Spring Data Jpa



## 3.2 Bootstrap

Bootstrap — це фреймворк CSS з відкритим вихідним кодом, спрямований на розробку графічних інтерфейсів. Він містить шаблони дизайну на основі CSS для кнопок, форм, навігації, посилань та інших компонентів інтерфейсу.  
 Результатом є уніфікований вигляд таблиць і елементів форм на всіх сторінках веб-браузера. Наприклад, у Bootstrap передбачені таблиці різних кольорів, заголовки сторінок, більші лапки та текст із виділенням.

Також для розробки графічного інтерфейсу використовувався чистий сss код. Більше того, були частково перезаписані класичні шаблони Bootstrap для надання графічному інтерфейсу специфічного стилю. До проекту підключена CDN версія Bootstrap 5.1.3 (див. рисунок 3.2.1), тобто при відсутності інтернету будуть неполадки з графічним інтерфейсом програмного продукту.

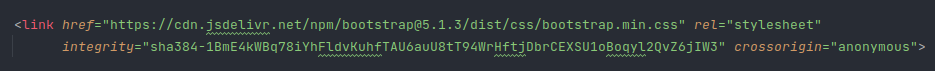


Рисунок 3.2.1 Підключення Bootstrap CDN до html файлу проекту

Далі, аби html файлі скористатися готовими стилями, розробленими Bootstrap, достатньо в атрибуті class прописати відповідний клас або класи. На рисунку 3.2.2 зображені теги для яких виконується призначення стилей фреймворку.

Рисунок 3.2.2 Використання стилей для елементів html сторінки



## 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 різних мовах програмування.

IntelliJ надає певні переваги, такі як авто 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 вікно пропозицій.  
 IntelliJ забезпечує інтеграцію з інструментами збірки та пакування, такими як maven, gradle і SBT. Він підтримує систему контролю версій Git, що широко використовувалася для зберігання різних версій проекту на репозиторії в Github. У версії IntelliJ IDEA Ultimate відкривається доступ до зручного інтерактивного інтерфейсу для роботи з базами даних на будь яких реляційних мовах включно з MySQL, що використовувався для написання макету бази даних.

IntelliJ IDEA Ultimate підтримує різноманітні фреймворки. Разом з Spring, вони в рази полегшують написання коду. Наприклад, середовище розробки автоматично відслідковує біни - об'єкти, якими керує контейнер Spring IoC. Інакший приклад - авто доповнення коду - Intellij враховує особливості фреймворків, що дозволяє застосовувати специфічне авто завершення коду шляхом аналізу контексту (див. Рисунок 3.3.1)

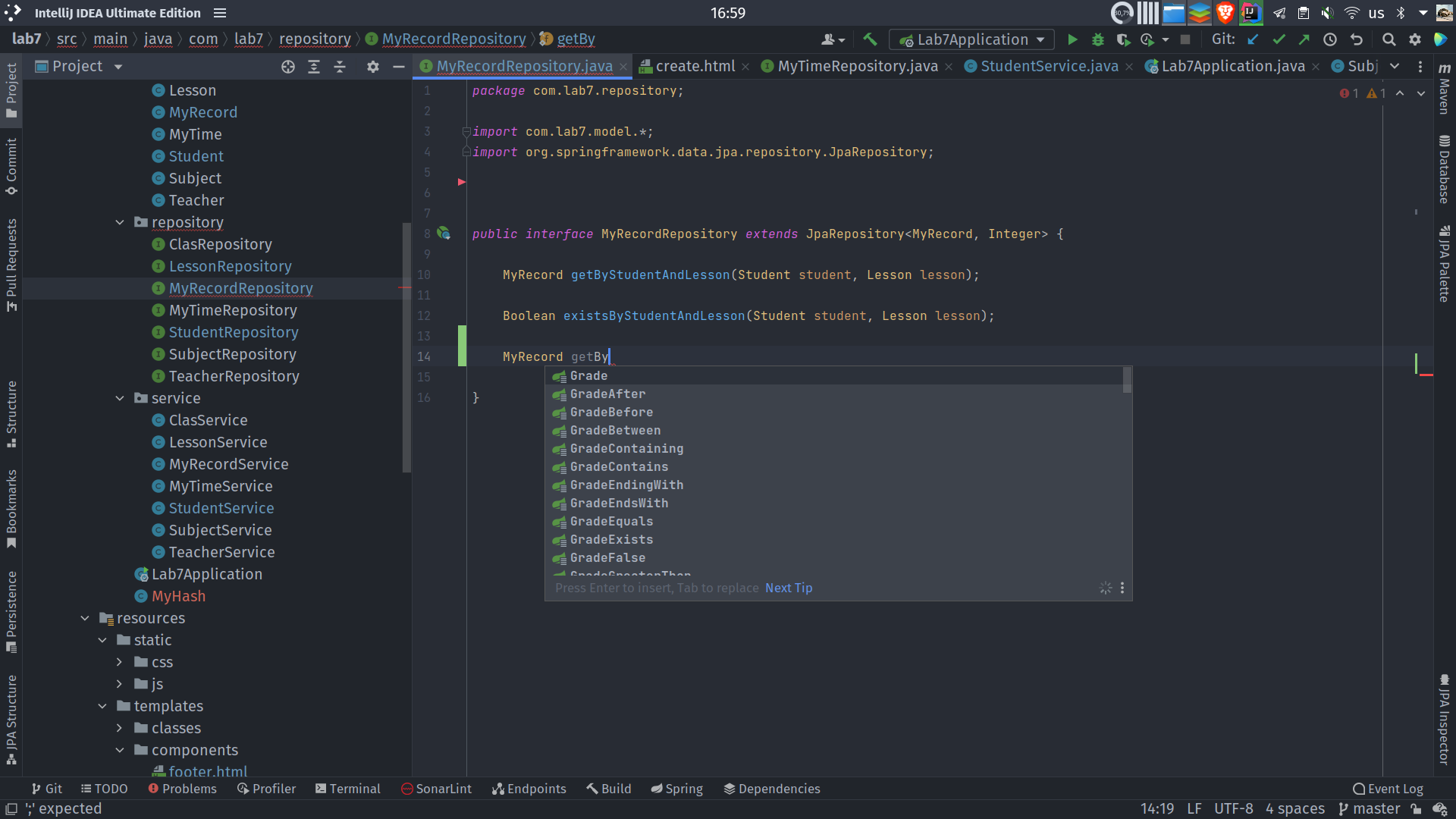


Рисунок 3.3.1 Intellij надає варіанти коду для фреймворку Spring

## 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 десктопних додатках. Конкретно в цьому проекті Thymeleaf дозволяє виділити такі компоненти сторінки як header та footer, а саме створити для кожного компонента окремий файл (див. Рисунок 3.4.1) та вставки цього блоку на інших сторінках (див. Рисунок 3.4.2).

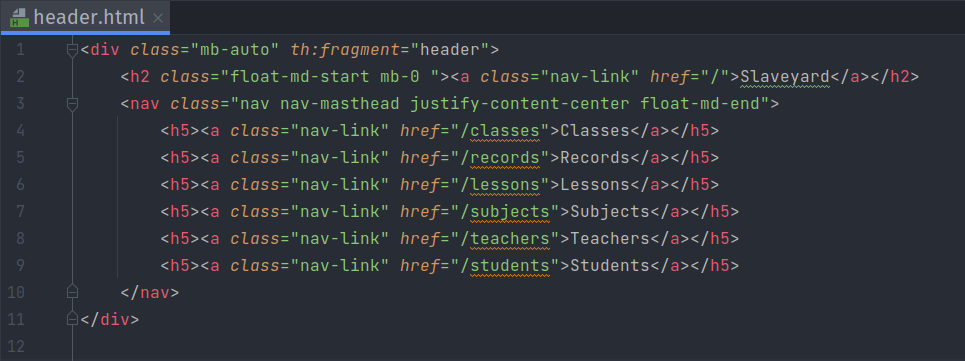


Рисунок 3.4.1 однаковий для усіх сторінок сайту елемент - header

Рисунок 3.4.2 однаковий для усіх сторінок сайту елемент - header



Головна мета Thymeleaf – надати чітко сформований спосіб створення шаблонів на серверній стороні додатку. 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.

Атрибути, що надає thymeleaf, неважко помітити, оскільки вони починаються з частинки “th:”. В більшості саме атрибути були використані в проекті при створені сторінок-шаблонів. Серед них :

* умовні оператори: th:switch, th:case та th:if
* Атрибут виведення інформації в документ th:text
* Ітераційний атрибут th:each
* Атрибут для отримання об’єкту на основі відправленої форми th:object. Далі об’єкт передається в контролер
* Та інші

Рисунок 3.4.3 фрагмент html коду з використанням Thymeleaf



Завдяки модулям для Spring Framework, Thymeleaf ідеально підходить для сучасної веб-розробки HTML5 сторінок-шаблонів, що було використано для розробки програмного продукту.

# 4 ОПИС ПРОЦЕСУ ВИКОНАННЯ ТА РЕЗУЛЬТАТІВ

## 4.1 База даних

У ході вивчення предметної області та наявного технічних засобів визначено, що для реалізації проекту оптимально використовувати база даних, що 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 клієнтській та серверній частині сайту, що значно прискорює роботу веб-додатку.

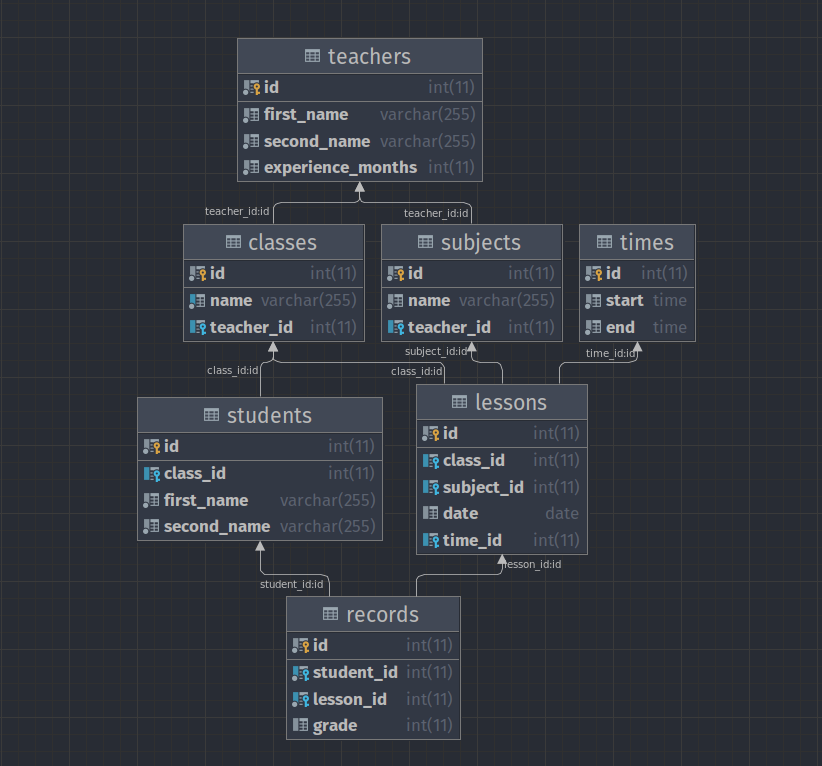


Рисунок 4.1.1 Модель бази даних

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 Серверна частина

Серверна частина, або ж іншими словами 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.

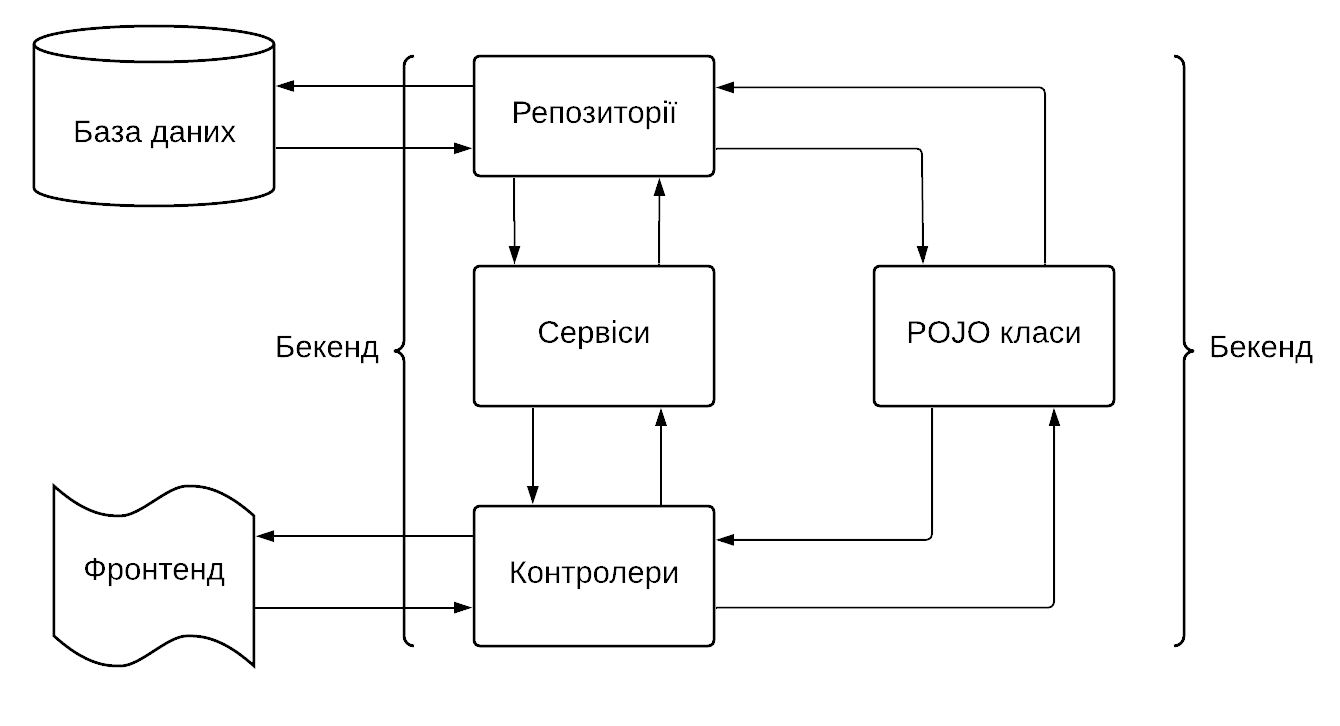


Рисунок 4.2.1 Структура бекенду у вигляді шарів

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).

Запити для контролеру таблиці “students” зображена у вигляді таблиці 5.4.1. Для інших таблиць запити є подібними. Винятком є останній запит, що переходить на сторінку на якій генерується звіт про успішність учня. Для маніпуляцій даними про успішність використовується спеціальний клас MyHash, що наслідує колекцію HashMap<String, List<Integer>>. MyHash містить методи для знаходження середньої, мінімальної, максимальної оцінки, а також дані про присутність за назвою предмета, значення, якого співпадає з ключем. Значення всередині List<Integer> - оцінки. 0 це знак відсутності учня на уроці.

|  |  |  |  |
| --- | --- | --- | --- |
| **Метод** | **Запит** | **Операція** | **Призначення** |
| GET | /students | index | Перехід на сторінку-таблицю |
| GET | /students/create | show | Перехід на сторінку створення даних про учня |
| POST | /students/create | create | Створення нового учня |
| GET | /students/update/{index} | show | Перехід на сторінку оновлення даних про учня за індексом |
| POST | /students/update | update | Оновлення даних про учня |
| GET | /students/delete/{index} | delete | Видалення рядка за індексом |
| GET | /students/details/{index} | show | Перехід на сторінку-звіт діяльності учня за індексом |

Таблиця 5.4.1 – запити класу-контролеру StudentController

Саме в контролерах відбувається валідація даних на сервері, 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 уже існує. При підтверджені, що такий об’єкт існує на клієнтську частину передається об’єкт типу Boolean.

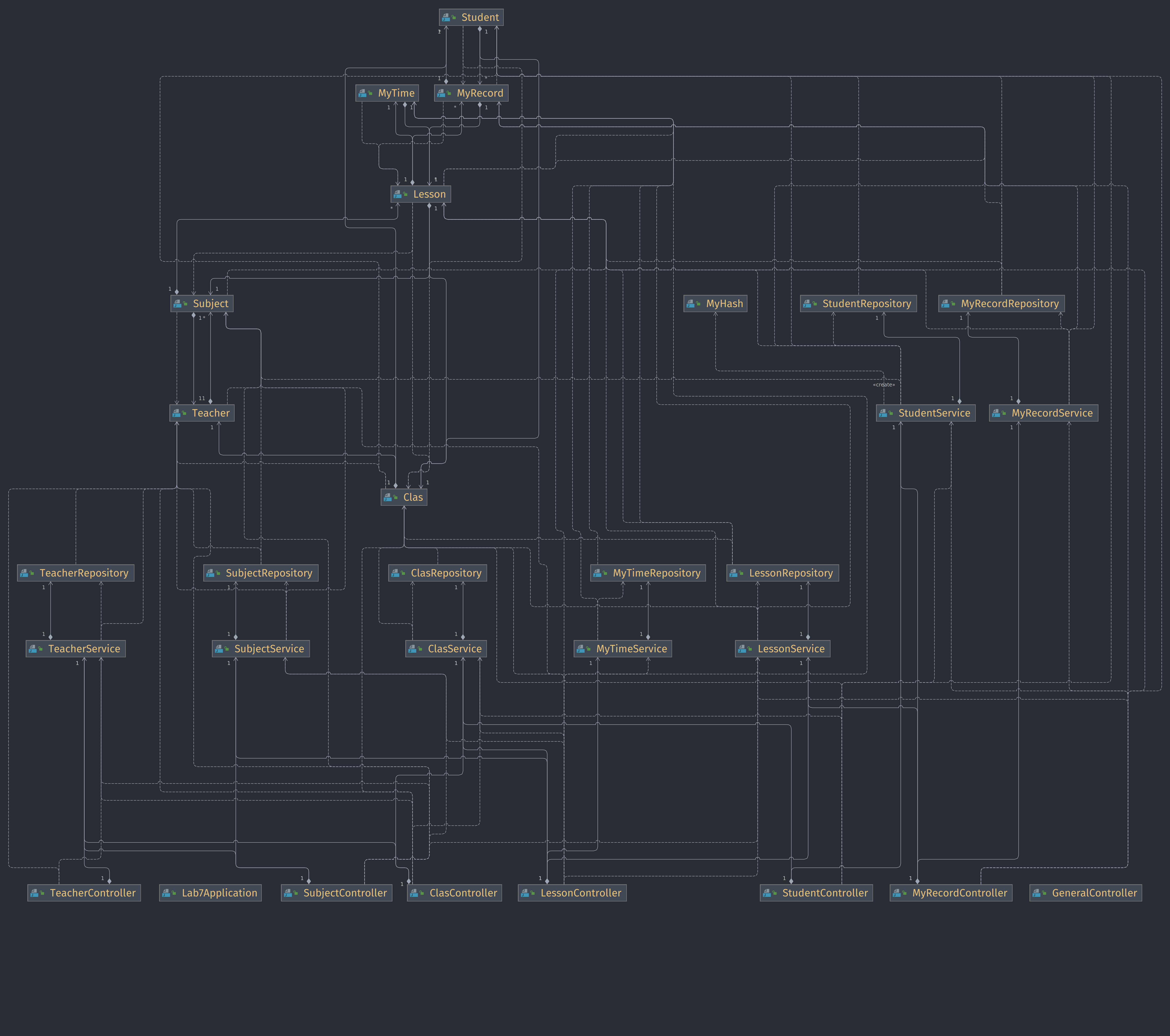
POJO класи - це класи-сутності бази даних. Тобто такий клас представляє собою таблицю відповідну таблицю. Очевидно, що клас складається з таких самих полів, що й таблиця та методів типу getter та setter для кожного поля. Для позначення взаємозв’язку класів і таблиць використовуються анотації. Вони також 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 клієнтську частину.

В усіх класах контролерів назва складається з назви сутності, якою контролер керується, та слова Controller. У репозиторіїв - це назва сутності та слово Repository. У сервісів закінчення Service. Згідно цих даних можна побудувати uml діаграму класів та вказати залежності. (див. Рисунок 4.2.2).

В цілому, найчастіше для класів використовується композиція, що вказано на схемі. У контролерах використовуються не тільки відповідні сервіси та POJO класи , а ще й інші. Такі залежності потрібні для реалізації у програмі списку можливих варіантів полів при створені або оновлені таблиці.

Клас GeneralController призначений для маршрутизації на головну сторінку. Клас Lab7Application - це точка входу спрінг-додатку.

Рисунок 4.2.2 Структура серверної частини сайту у вигляді uml схеми класів



## 4.3 Клієнтська частина

Головна сторінка, як й інші, складаються з хедера з посиланнями на сторінки-таблиці, тіла та однакового футера. Відкрити цю сторінку можна перейшовши за посиланням localhost:8080 ( див. Рисунок 4.3.1 ).

Сторінки-таблиці ( див. Рисунок 4.3.2 ) містять таблиці з даними. На цих сторінках рядки в таблиці можна динамічно фільтрувати за допомогою кількох полів. Ввівши якесь значення у видільний для цього рядок, одразу виводяться рядки, що містять у собі це значення. Пошук проводиться по всім елементам рядка. Ця пошукова система написана на мові програмування javascript.

Суть використання декількох пошукових рядків полягає в знаходженні рядків, що задовольняють усі фільтри, тобто шукається перетин множин рядків виведених фільтрами. Якщо шукати учнів, що учаться в 10 класі, таких людей всього 2, та тих у кого прізвище Singletonenko, теж всього 2, то отримаємо перетин у вигляді лише одного запису, що зображено на рисунку 4.3.3.

Видалення відбувається шляхом натискання на посилання, що називається “Delete”. Дані також можна оновлювати та додавати. Для кожної сторінки-таблиці існують свої сторінки типу create ( див. Рисунок 4.3.4 ) та update ( див. Рисунок 4.3.5 ) для відповідних POST запитів, що передають дані, введені у форму, на серверну частину сайту. Саме на цих сторінках відбувається валідація даних на стороні клієнта, а саме перевірка чи ім’я відповідає regex або чи число входить в допустимий інтервал значень. ( див. Рисунок 4.3.6 ). На клієнтській стороні приписана перевірка дати, що дозволяє назначити дату проведення уроку лише на будні дні.

При неправильно введеній адресі, відбувається перехід на сторінку помилки. ( див. Рисунок 4.3.7 ). Прописані подібні сторінки ще для помилок з кодами 500 ( внутрішня помилка серверу ) та 400 ( поганий запит ).

При невдалій валідації даних на серверній частині, як, наприклад, спробі внести дані, що вже є в таблиці, сторінка сайту відображає помилку ( див Рисунок. 4.2.8 ).

Згідно постановки задачі, було створено індивідуальні сторінки-звіти для кожного студента. Перейти на таку сторінку можна при нажаті посилання з назвою “Details” будь-якого рядка на сторінці-таблиці учнів. У звіт складається з рядків, що містять дані про один предмет, який відвідував учень на уроках. Рядок містить назву предмету, середній бал, найвищу та найнижчу оцінки, процентне значення присутності, кількість пропущених уроків та загальну їхню кількість ( див. Рисунок 4.2.9 ).

У разі відсутності даних про відвідування учнем уроків, відображається відповідна сторінка, що на рисунку 4.2.10. Більше того, сторінки-таблиці теж мають таку особливість. Тобто, якщо у таблиці “students” не буде жодного учня, то появиться подібне повідомлення, а внизу буде кнопка для створення нового учня.

Рисунок 4.2.1 Головна сторінка

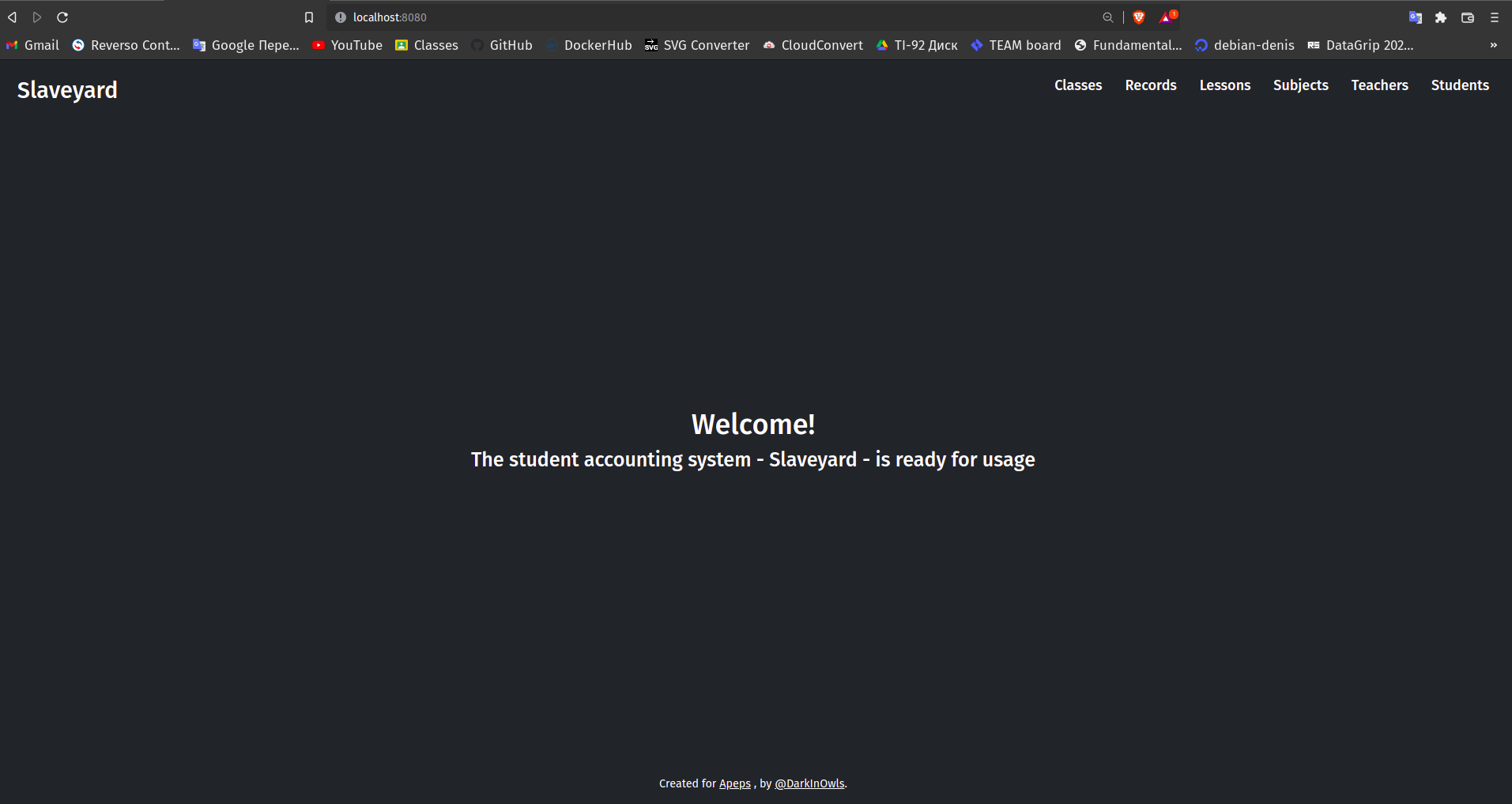


Рисунок 4.2.2 Сторінка-таблиця

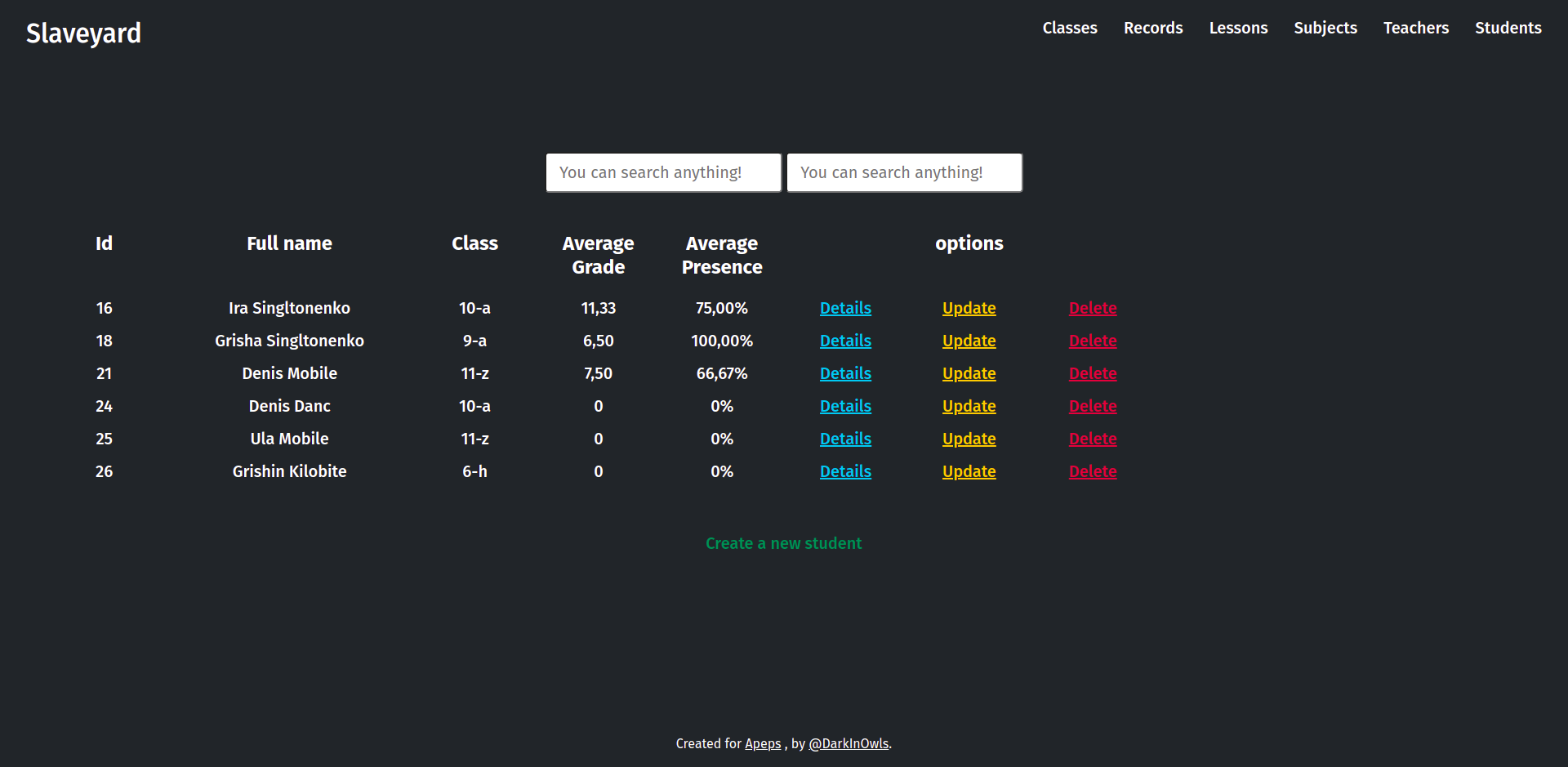


Рисунок 4.2.3 Відфільтровані дані за двома фільтрами

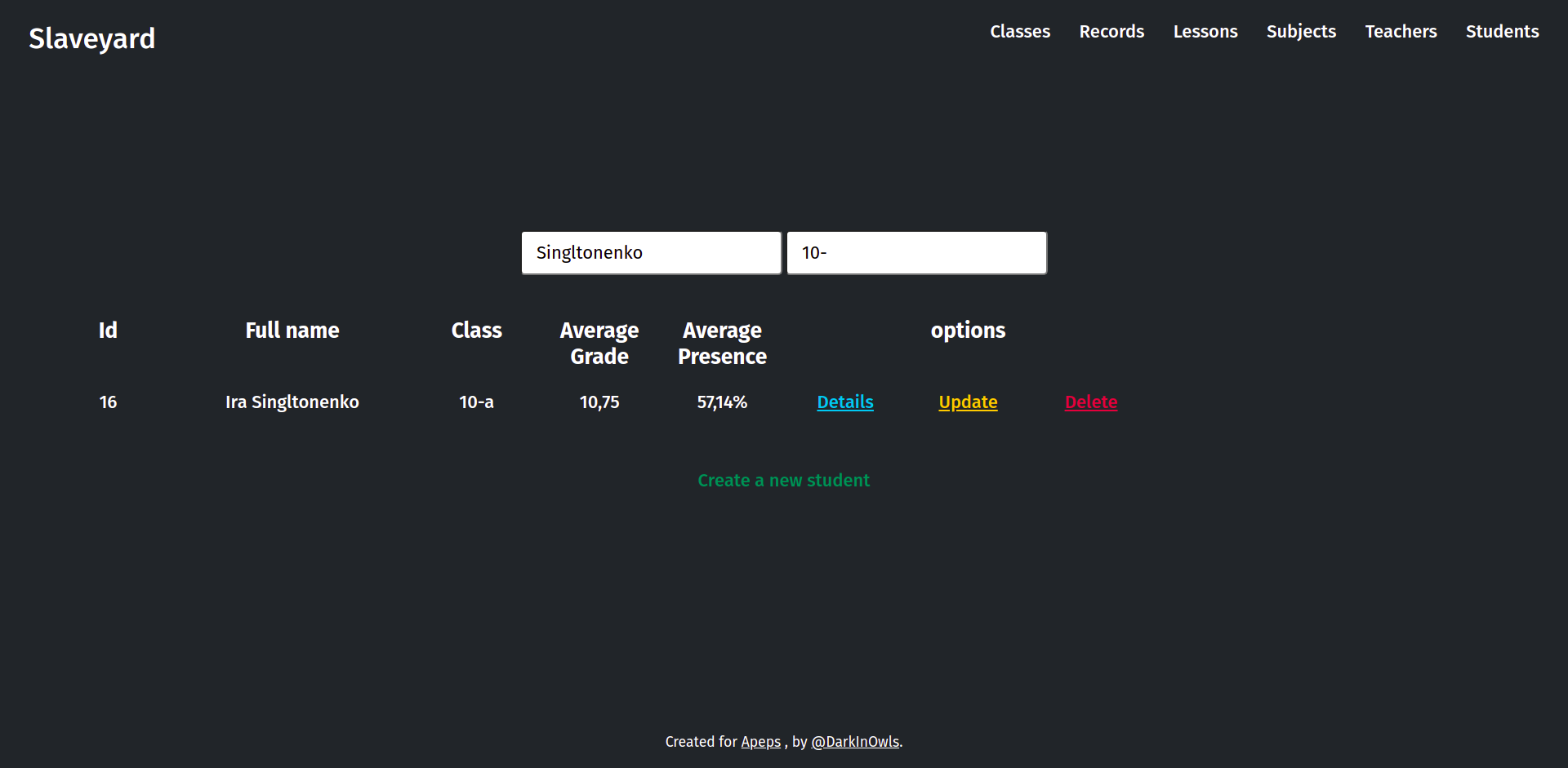


Рисунок 4.2.4 Сторінка типу create

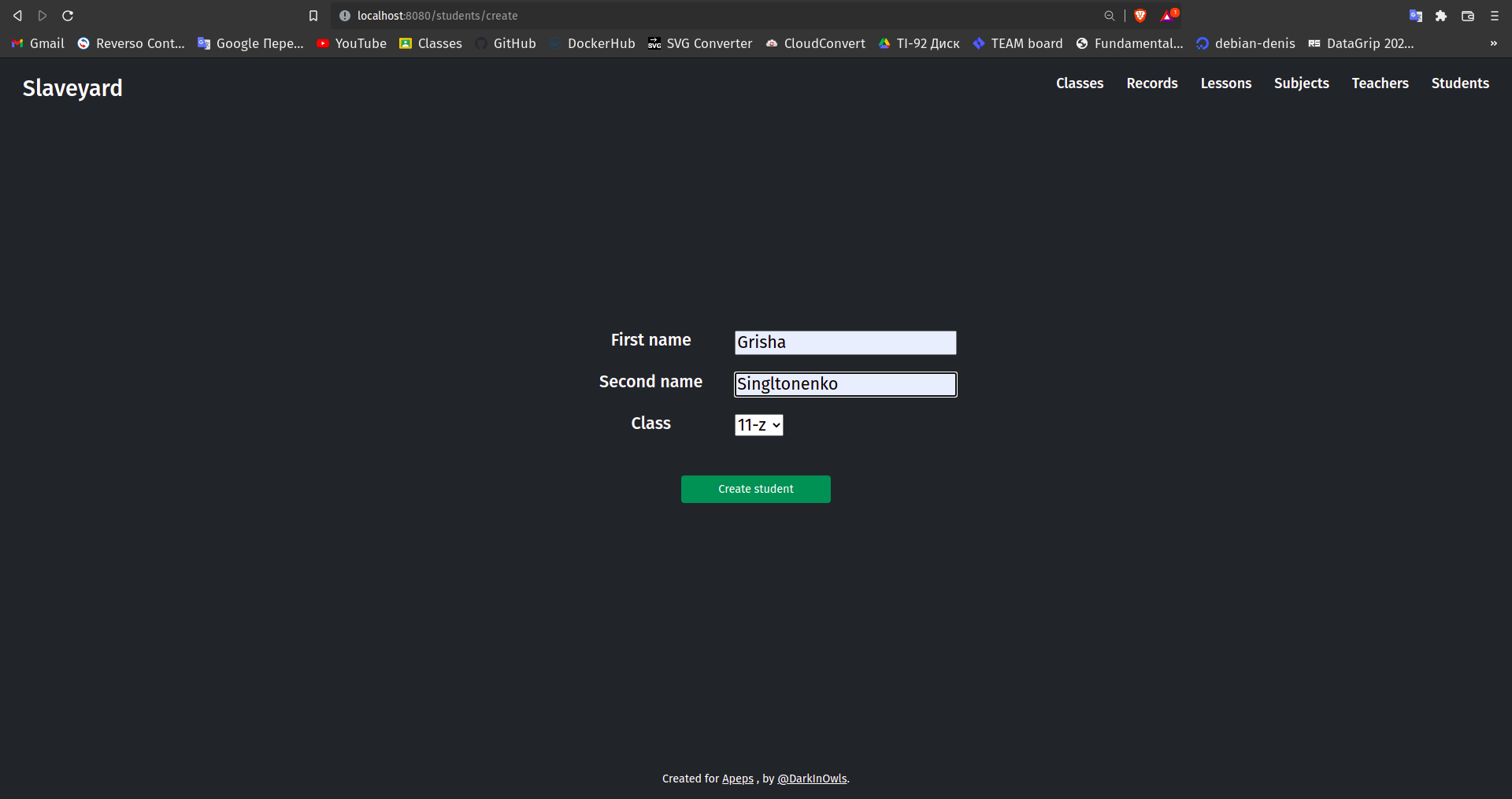


Рисунок 4.2.5 Сторінка типу update

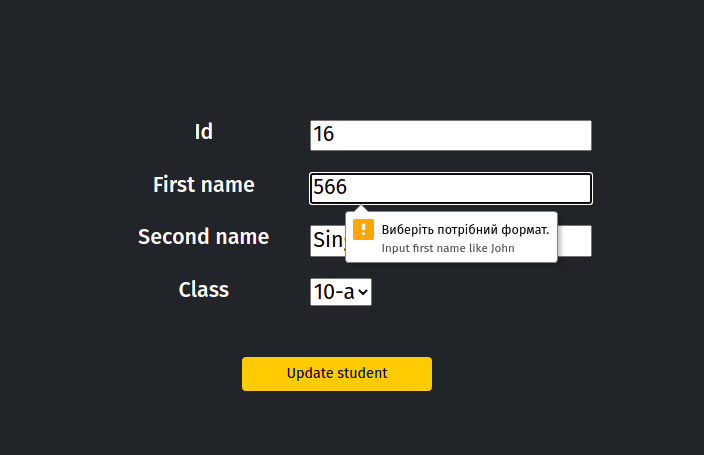
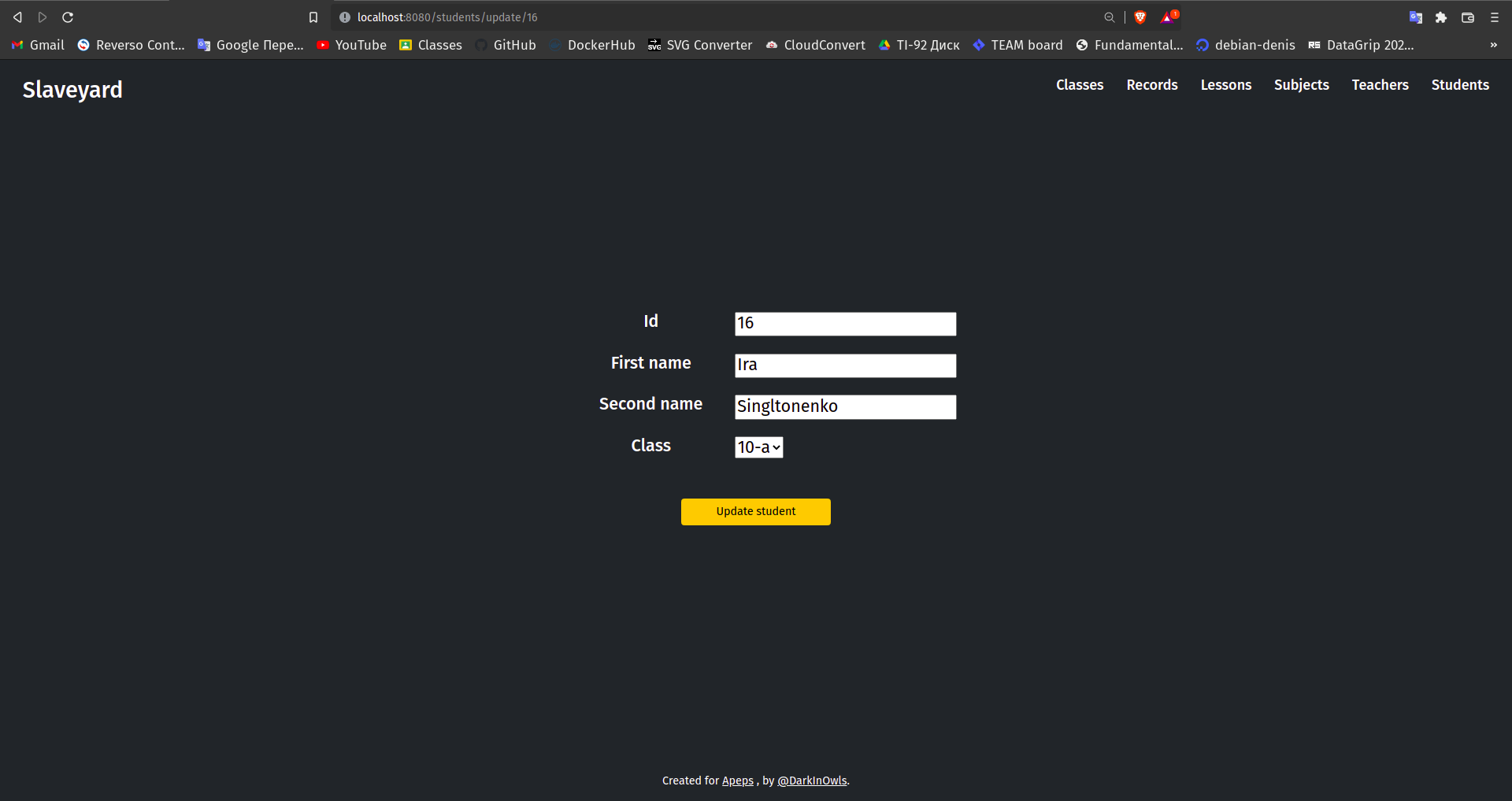


Рисунок 4.2.6 Помилка валідації даних на клієнтській стороні

Рисунок 4.2.7 Сторінка помилки

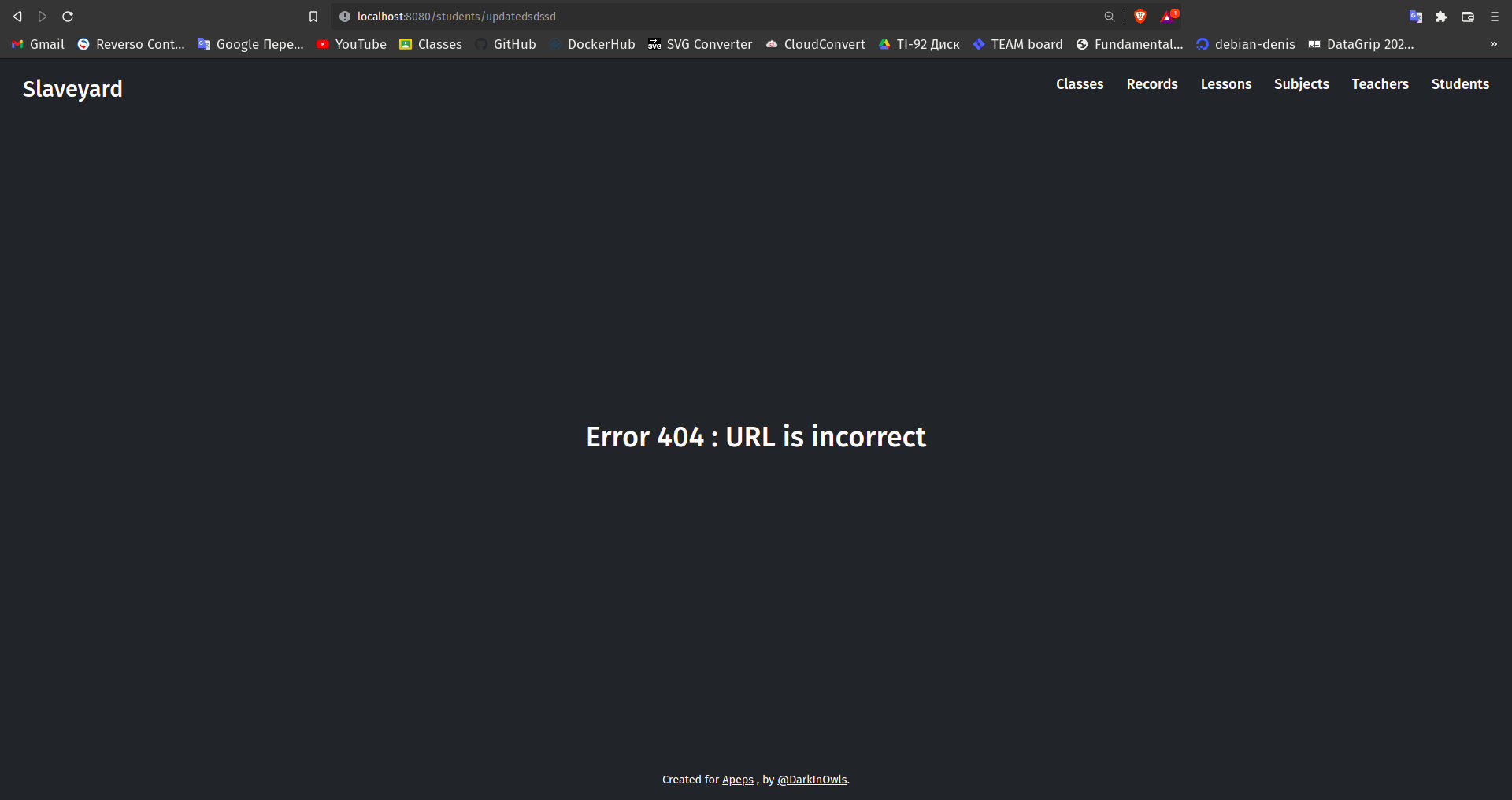


Рисунок 4.2.8 Помилка валідації даних на серверній стороні

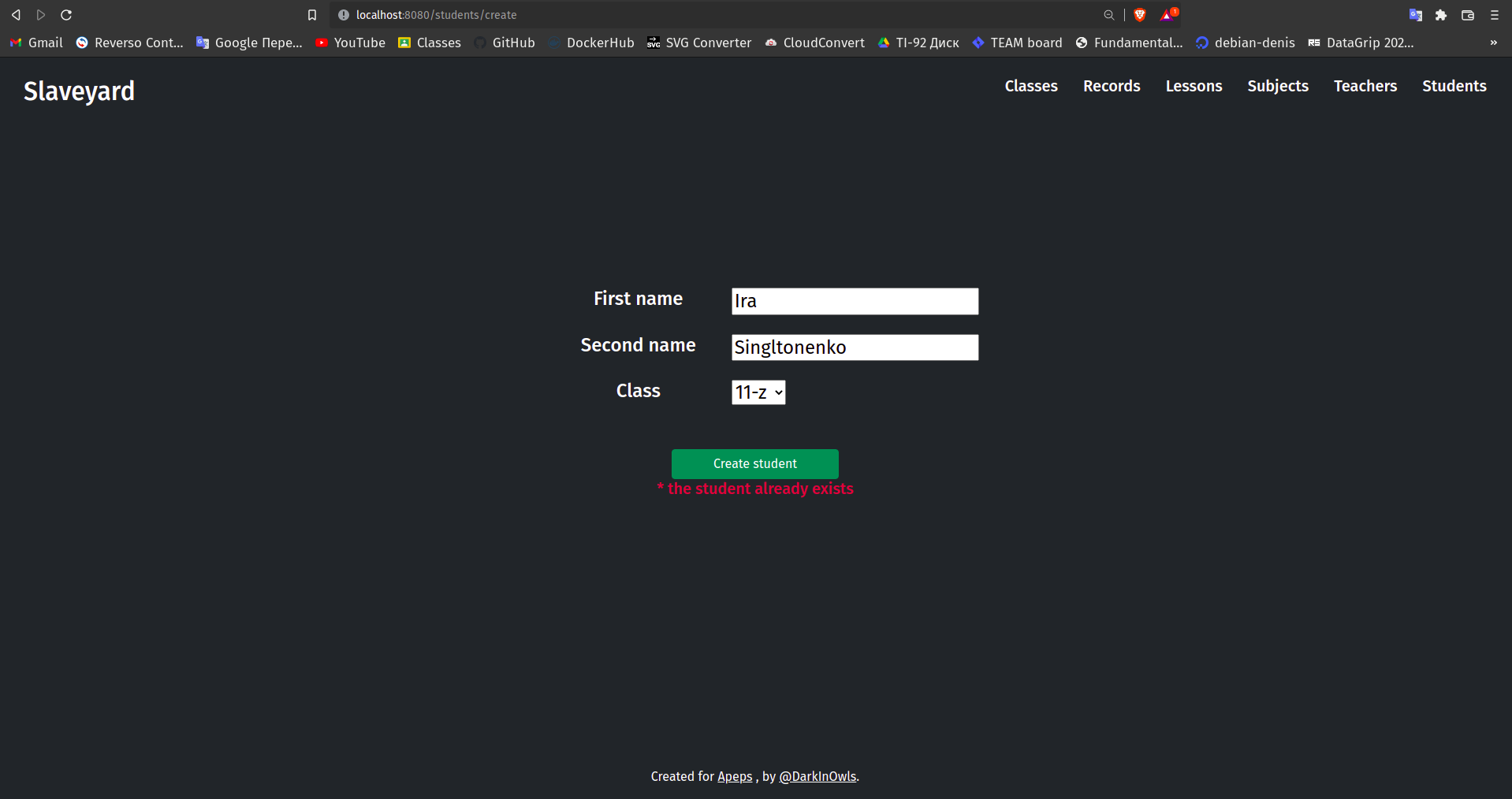


Рисунок 4.2.9 Сторінка-звіт діяльності учня

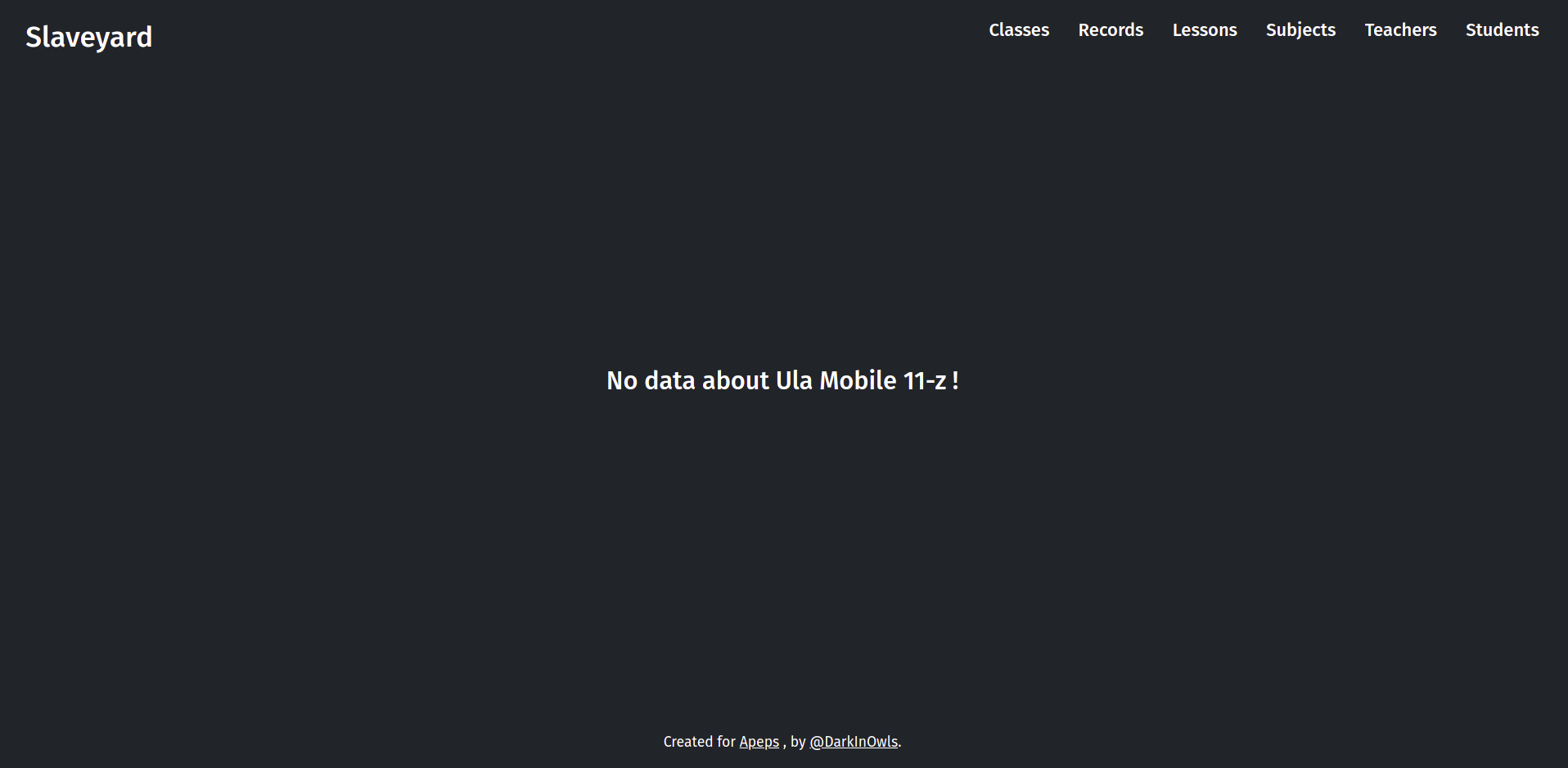
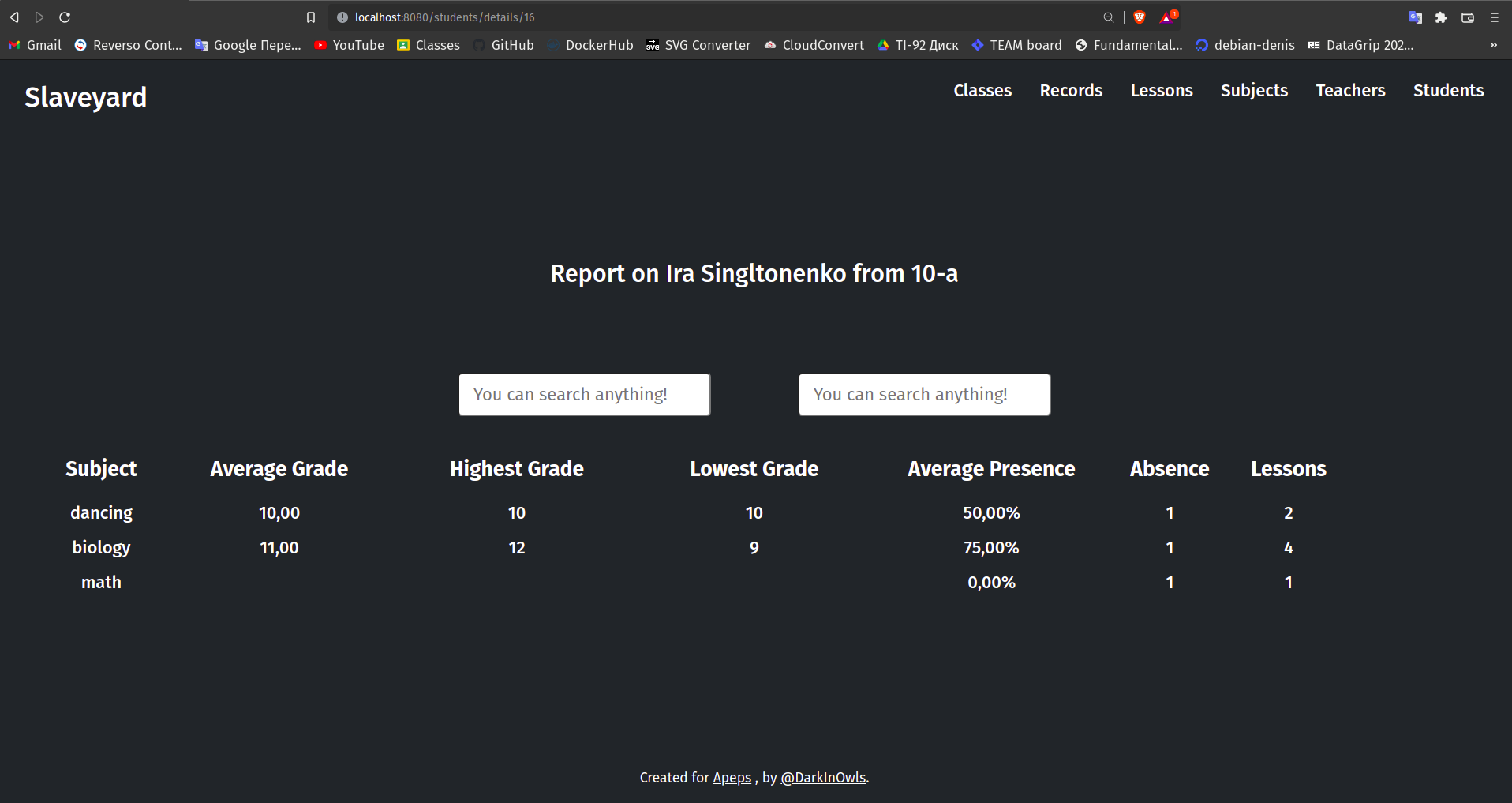


Рисунок 4.2.10 Повідомлення про відсутність даних для звітування

# 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 справді красивим і зручним, а основний функціонал - реалізований. Що важливо, база даних потребувала більше затрат часу та точного плану розробки. В результаті було отримано оптимальну базу даних шляхом поступового її зведення до третьої нормальної форми.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Java Documentation. URL : <https://docs.oracle.com/en/java/>
2. Spring Framework Documentation. URL : <https://docs.spring.io/spring-framework/docs/current/reference/html/>
3. Tutorial: Using Thymeleaf. URL : <https://www.thymeleaf.org/doc/tutorials/3.0/usingthymeleaf.html>
4. Bootstrap 4 Tutorial. URL : <https://www.w3schools.com/bootstrap4/>
5. MySQL Documentation. URL : <https://dev.mysql.com/doc/>
6. Головна сторінка сайту. URL : <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 1

Класи для обробки даних з таблиці subjects на серверній частині сайту

SubjectRepository.java

package com.lab7.repository;

import com.lab7.model.Subject;

import com.lab7.model.Teacher;

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

public interface SubjectRepository extends JpaRepository<Subject, Integer> {

Boolean existsByNameAndTeacher(String name, Teacher teacher);

}

SubjectService.java

package com.lab7.service;

import com.lab7.model.Subject;

import com.lab7.model.Teacher;

import com.lab7.repository.SubjectRepository;

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

import org.springframework.stereotype.Component;

import java.util.List;

@Component

public class SubjectService {

private final SubjectRepository subjectRepository;

@Autowired

public SubjectService(SubjectRepository subjectRepository) {

this.subjectRepository = subjectRepository;

}

public Subject findById(int id){

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

}

public List<Subject> findAll(){

return subjectRepository.findAll();

}

public Subject saveSubject(Subject subject){

return subjectRepository.save(subject);

}

public void deleteById(int id){

subjectRepository.deleteById(id);

}

public boolean existsByNameAndTeacher(String name, Teacher teacher){

return subjectRepository.existsByNameAndTeacher(name, teacher);

}

}

Subject.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 = "subjects")

public class Subject {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int id;

@Column(name = "name", unique = true)

private String name;

@ManyToOne

@JoinColumn(name = "teacher\_id")

private Teacher teacher;

@OneToMany(mappedBy = "subject")

private List<Lesson> lessons;

}

SubjectController.java

package com.lab7.controller;

import com.lab7.model.Lesson;

import com.lab7.model.Subject;

import com.lab7.model.Teacher;

import com.lab7.service.LessonService;

import com.lab7.service.SubjectService;

import com.lab7.service.TeacherService;

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

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

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

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

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

import java.util.List;

@Controller

public class SubjectController {

private final SubjectService subjectService;

private final TeacherService teacherService;

@Autowired

public SubjectController(SubjectService subjectService, TeacherService teacherService, LessonService scheduleService) {

this.subjectService = subjectService;

this.teacherService = teacherService;

}

@GetMapping("/subjects")

public String findAll(Model model) {

List<Subject> subjects = subjectService.findAll();

model.addAttribute("subjects", subjects);

return "subjects/subjects";

}

@GetMapping("/subjects/create")

public String createSubjectForm( Subject subject, Model model) {

List<Teacher> teachers = teacherService.findAll();

model.addAttribute("teachers", teachers);

return "subjects/create";

}

@PostMapping("/subjects/create")

public String createSubject(Subject subject, Model model) {

if(Boolean.TRUE.equals(subjectService.existsByNameAndTeacher(subject.getName(), subject.getTeacher()))) {

model.addAttribute("exist", true);

return createSubjectForm(subject, model);

}

subjectService.saveSubject(subject);

return "redirect:/subjects";

}

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

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

subjectService.deleteById(id);

return "redirect:/subjects";

}

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

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

Subject subject = subjectService.findById(id);

model.addAttribute("subject", subject);

List<Teacher> teachers = teacherService.findAll();

model.addAttribute("teachers", teachers);

return "subjects/update";

}

@PostMapping("/subjects/update")

public String updateSubject(Subject subject, Model model) {

if(Boolean.TRUE.equals(subjectService.existsByNameAndTeacher(subject.getName(), subject.getTeacher()))) {

model.addAttribute("exist", true);

return updateSubjectForm(subject.getId(), model);

}

subjectService.saveSubject(subject);

return "redirect:/subjects";

}

}