sМІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: 121 «Інженерія програмного забезпечення»

Освітня програма: «Розробка програмного забезпечення»

Група: 4РП-07

Дипломний проєкт

здобувача освіти денної форми навчання

РП.07.16.000.ДП

НІКОЛАЄВА ВАЛЕНТИНА ПЕТРОВИЧА

м. Одеса
2024 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: **121 «Комп'ютерна інженерія»**

Освітньо-професійна програма: **«Інженерія програмного забезпечення»**

Група: **4РП-07**

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проєкту на тему:

Розробка Back-End частини автоматизованої веб-системи обліку руху контингенту здобувачів відділення комп'ютерних систем

Проєктний матеріал складається з пояснювальної записки на **92** сторінках та графічного (презентаційного) матеріалу на **20** аркушах (слайдах).

Дипломник _____ (Ніколаєв В. П.)

Керівник _____ (Жадан А. С.)

Консультанти:

з економічного розділу _____ (Іванченков В. С.)

з розділу охорони праці та техніки безпеки _____ (Чорновол Н. І.)

з нормоконтролю _____ (Петрашова В. І.)

старший консультант _____ (Кривченко Ю. В.)

До захисту допущений

Голова циклової комісії _____ (Кривченко Ю. В.)

Завідувач відділення _____ (Скорнякова О. В.)

Захист «25» 06 2024 р. Протокол ДКК № 2

Оцінка ДКК 5 (відмінно) / 95 б

Секретар ДКК _____

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Відділення *комп'ютерних систем* Комісія *КТ та ПІ*

Спеціальність *121 «Інженерія програмного забезпечення»*

Освітня програма *«Розробка програмного забезпечення»*

ЗАТВЕРДЖУЮ:

Заст. дир. з НВР *Беркань І. В.*

"17" 01 *2024 року*

ЗАВДАННЯ

на дипломний проєкт

Здобувачеві освіти *Ніколаєву Валентину Петровичу*

1. Тема проєкту *Розробка Back-End частини автоматизованої веб-системи обліку руху контингенту здобувачів відділення комп'ютерних систем*

Затверджена наказом по коледжу від "02" *листопада* *2023 р.*, наказ № *244-А2-ОД*

2 Термін здачі закінченого проєкту

3. Вихідні дані до проєкту

1. Використовувати СУБД MySQL

2. Застосовувати фреймворк Laravel

3. Застосовувати принципи ООП – SOLID та Design Patterns

4. Зміст розрахунково-пояснювальної записки (перелік питань, які необхідно розробити)

1. Аналіз предметної області. 2. Технології та засоби розробки (проєктування).

3. Проєктування архітектури веб-застосунку. 4. Розробка Back-End частини MVC-застосунку.

5. Тестування створеного веб-застосунку.

6. Економічний розрахунок. 7. Аспекти охорони праці та техніки безпеки

5. Перелік графічного (презентаційного) матеріалу (з точним зазначенням обов'язкових креслень, кількості слайдів)

Презентація Power Point – 20 слайдів

(Схема бази даних; Архітектура веб-системи; Docker-схема; Створення міграцій; Створення моделей; Створення роутів; Створення контролерів; Тестування веб-застосунку)

6. Консультанти по проєкту, із зазначенням розділів проєкту, що їх стосується

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Основний розділ	*Жадан А. С.*		
Економічний розділ	*Іванченков В. С.*		
Розділ охорони праці	*Чорновол Н. І.*		
Нормоконтроль	*Петрашова В. І.*		
Старший консультант	*Кривченко Ю. В.*		

7. Дата видачі завдання 15 січня 2024 р.

Керівник *Жадан А. С.* (підпис)

Завдання прийняв до виконання *Ніколаєв В. П.* (підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/р	Назва етапів дипломного проєкту (роботи)	Термін виконання етапів дипломного проєкту (роботи)	Відмітка про виконання
1	*Формування вступу*	*29.04.24*	виконано
2	*Аналіз предметної області*	*10.05.24*	виконано
3	*Підбір технічної літератури*	*19.05.24*	виконано
4	*Вибір технологій та засобів розробки (проєктування)*	*20.05.24*	виконано
5	*Проєктування архітектури веб-застосунку*	*24.05.24*	виконано
6	*Реалізація Back-End частини MVC-застосунку*	*27.05.24*	виконано
7	*Тестування створеного веб-застосунку*	*29.05.24*	виконано
8	*Оформлення пояснювальної записки*	*31.05.24*	виконано
9	*Оформлення графічної (презентаційної) частини*	*01.06.24*	виконано
10	*Економічний розрахунок*	*02.06.24*	виконано
11	*Опис охорони праці та техніки безпеки*	*09.06.24*	виконано
12	*Аналіз результатів проєктування*	*13.06.24*	виконано
13	*Підготовка доповіді для захисту*	*16.06.24*	виконано

Дипломник (підпис)

Керівник (підпис)

Формат	Зона	Поз.	Позначення	Назва	Кіл.	Примітка
				Документація		
			РП 07. 16 000. 00 ДП	Дипломний проєкт		
А4			РП 07. 16 000. 00 ДП ПЗ	Пояснювальна записка	1	

Зм.	Арк.	№ докум.	Підпис	Дата	РП 07. 16 000. 00 ДП ПЗ		
Розроб.		Ніколаєв В.П.		10.06	Розробка Back-End частини автоматизованої веб-системи обліку руху контингенту здобувачів відділення комп'ютерних систем	Літ.	Аркуш 4 / Аркушів 92
Перевірив		Жадан А.С.		10.06		Н Д П	
Н. Контр.		Петрашова В.І.		10.06		ВСП "ОТФК ОНТУ" гр.4РП-07	
Затверд.		Кривченко Ю.В.		10.06			

ЗМІСТ

ВСТУП

Інформаційні веб-системи для автоматизації учбових процесів у навчальних закладах набувають все більшої важливості. Навчальні установи постійно шукають способи підвищення продуктивності та ефективності, а також мінімізації витрат, і веб-системи стають ключовим інструментом у досягненні цих цілей. Інформаційні веб-системи дозволяють ефективно керувати даними, включаючи облік руху контингенту, організацію навчального процесу та забезпечення потрібною інформацією відповідальних осіб.

Метою даної роботи є розробка інформаційної веб-системи для автоматизації учбових процесів в адміністративному відділі коледжу, зокрема, для управління даними та документами, пов'язаними з рухом контингенту здобувачів відділення комп'ютерних систем. Основна задача розробки полягає в створенні системи управління контентом, що дозволить адміністраторам ефективно керувати процесами обліку та аналізувати потрібну інформацію для прийняття управлінських рішень.

Для досягнення поставлених завдань застосовувався системний підхід. При аналізі основних проблем предметної області та дослідженні аналогів інструментів розробки використовувався наступний стек технологій: Laravel 10, PHP 8.2, що забезпечує високу продуктивність та безпеку програмного забезпечення. Для зберігання та обробки даних використовується реляційна база даних MySQL 8.0. Також використовується Docker, який дозволяє забезпечити портативність та масштабованість розгорнутих застосунків та за потреби швидко змінити оточення.

Результатом роботи стала інформаційна веб-система, яка призначена для внутрішнього використання адміністраторами ОТФК ОНТУ. Ця система допоможе автоматизувати процеси обліку та управління даними студентів, спростить робочі процеси та покращить ефективність управління в адміністративному відділі.

1 ОСНОВНИЙ РОЗДІЛ

1.1 Аналіз предметної області

1.1.1 Огляд існуючих рішень

Станом на 2024 рік на ринку програмного забезпечення у контексті управління даними та документами в навчальних закладах, можна виявити тенденцію зростання веб-систем з функціоналом, який покращує досвід роботи з даними. Розглянемо декілька аналогів:

Аналог 1 – SIS (Student Information System):

На рис. 1.1 зображено інтерфейс застосунку SIS, який є одним з найпоширеніших рішень у сфері управління даними про студентів. Це комплексна система, яка включає в себе модулі для реєстрації студентів, обліку присутності, виставлення оцінок, статистики успішності тощо. Вона дозволяє автоматизувати багато аспектів управління студентським контингентом.

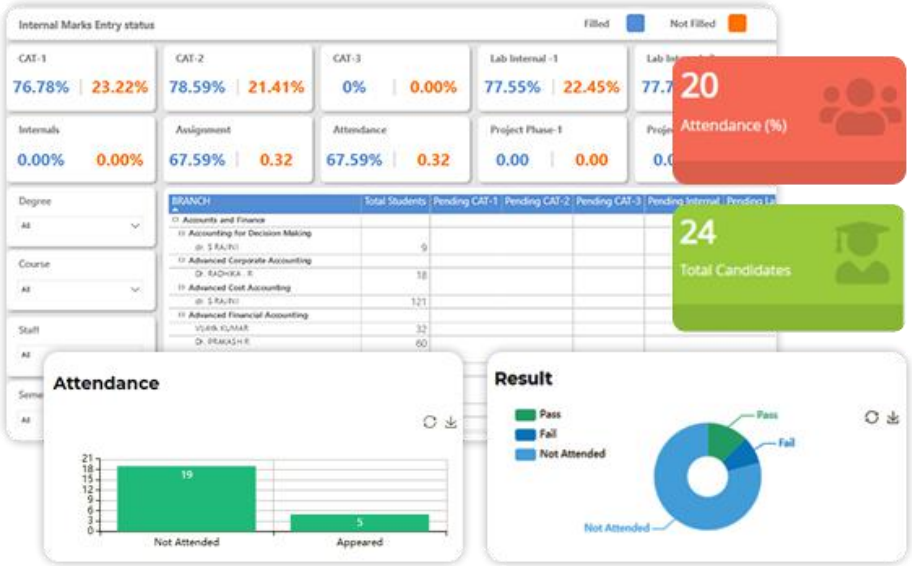


Рисунок 1.1. Приклад функціоналу застосунку SIS

Переваги SIS:

- Широкий функціонал, що охоплює багато аспектів управління студентами.
- Досвід використання та надійність, оскільки SIS використовується в багатьох навчальних закладах.

Недоліки SIS:

- Високі вартості впровадження та підтримки.
- Може бути складно налаштовуватися під конкретні потреби закладу.

Аналог 2 – OpenEduCat:

Наступний приклад на рис. 1.2 – OpenEduCat, який є відкритим програмним забезпеченням для управління навчальними закладами, яке включає в себе різні модулі, включаючи облік студентів, розклад занять, бібліотеку тощо. Цей застосунок спрямований на навчальні заклади різного рівня, від шкіл до вищих навчальних закладів.

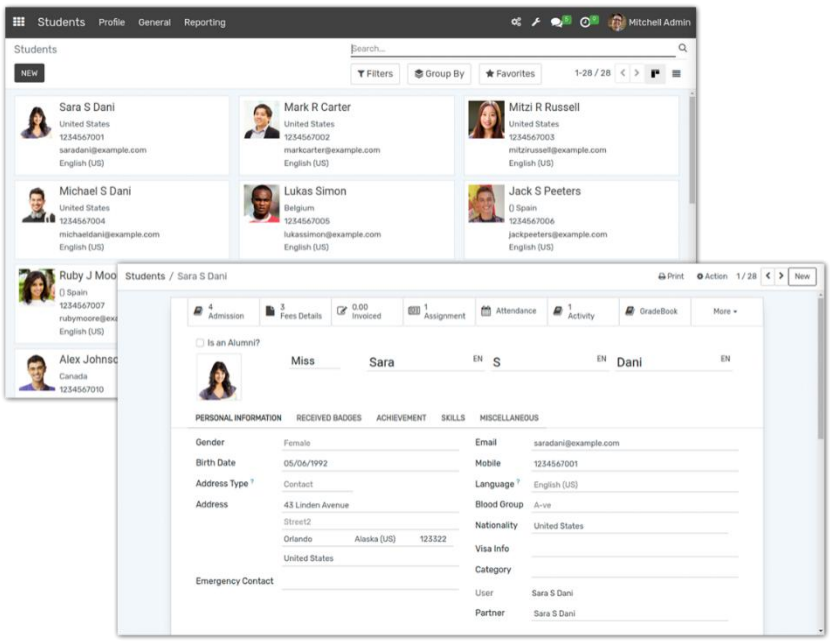


Рисунок 1.2. Приклад інтерфейсу застосунку OpenEduCat

Переваги OpenEduCat:

- Відкрите програмне забезпечення, що дозволяє змінювати та адаптувати систему під потреби закладу.
- Велика спільнота користувачів та розробників, що забезпечує підтримку та постійний розвиток системи.

Недоліки OpenEduCat:

- Можливі проблеми зі сумісністю модулів при оновленнях.
- Потребує достатньої технічної експертизи для налаштування та підтримки.

Аналог 3 – Fedena:

Fedena є ще одним популярним рішенням для управління навчальними закладами. Вона включає в себе модулі для обліку студентів, вчителів, фінансів, бібліотеки, розкладу тощо. Fedena також надає можливість використовувати різні додаткові модулі та розширення. На рис. 1.3 можна переглянути зовнішній вигляд цього веб застосунку.

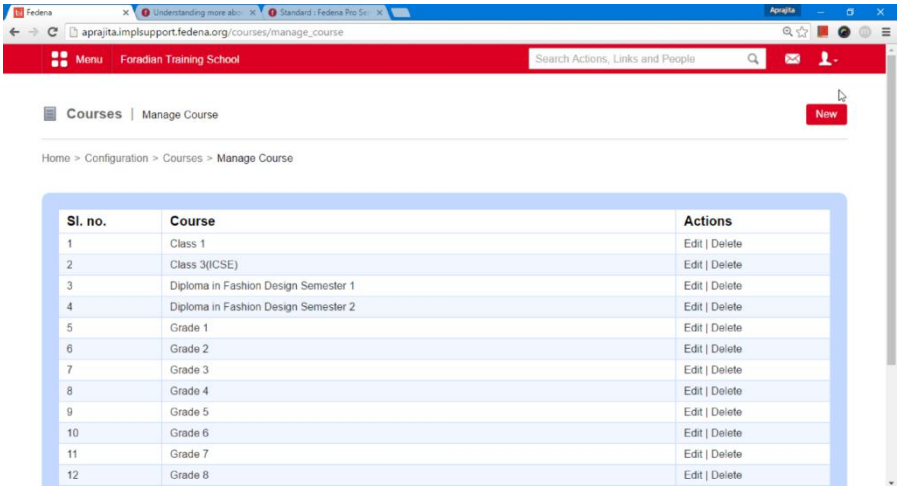


Рисунок 1.3. Інтерфейс застосунку Fedena

Переваги Fedena:

- Простий у використанні та налаштуванні.
- Має ряд готових модулів та розширень для різних потреб навчальних закладів.

Недоліки Fedena:

- Може бути обмежений у функціоналі порівняно з іншими рішеннями.
- Можливість обслуговування та підтримки може бути обмеженою залежно від постачальника.

Порівняльну характеристику розглянутих аналогів та їх функціоналу наведено у табл. 1.1.

Таблиця 1.1. Порівняльна характеристика аналогів

Характеристика / Рішення	**SIS**	**OpenEduCat**	**Fedena**
Відкрите програмне забезпечення	Ні	Так	Ні
Функціонал	Широкий, охоплює багато аспектів управління студентами	Різноманітні модулі, включаючи облік студентів, розклад занять тощо	Має модулі для обліку студентів, вчителів, фінансів, бібліотеки тощо
Спільнота користувачів та розробників	Велика	Велика	Можливо не така велика
Вартість	Висока	Залежить від конфігурації та підтримки	Залежить від постачальника
Складність налаштування	Може бути складно	Потребує достатньої технічної експертизи	Простий у використанні та налаштуванні
Підтримка	Доступна	Залежить від спільноти користувачів та розробників	Може бути обмеженою залежно від постачальника

Продовження таблиці 1.1

Характеристика / Рішення	**SIS**	**OpenEduCat**	**Fedena**
Гнучкість	Обмежена	Велика	Можливо обмежена
Надійність	Висока	Залежить від встановлення та налаштування	Може бути обмеженою залежно від постачальника

1.1.2 Технології та засоби розробки

Для реалізації Back-End частини автоматизованої веб-системи обліку руху контингенту здобувачів відділення комп'ютерних систем були обрані наступні технології:

PHP являє собою скриптову мова програмування загального призначення, яка використовується для розробки веб-застосунків та динамічних веб-сайтів. Вибір версії PHP 8.2 обумовлений його високою продуктивністю, покращеною безпекою та новими можливостями, які полегшують розробку та підтримку коду. Ця версія є останньою стабільною та підтримується розробниками, що забезпечує надійність та безпеку програмного забезпечення [1].

NGINX є високопродуктивним веб-сервером та проксі-сервером з відкритим вихідним кодом, який широко використовується для обробки статичних файлів, обслуговування веб-сайтів, балансування навантаження, зворотного проксі, кешування та інших завдань веб-інфраструктури. Він відомий своєю ефективністю, надійністю та масштабованістю, що робить його популярним вибором для веб-розробників та системних адміністраторів. Веб-сервер NGINX був обраний через його високу швидкодію та здатність ефективно обробляти великі обсяги запитів. Він є популярним рішенням у сфері веб-розробки через свою надійність та зручність в налаштуванні [2].

В якості реляційної бази даних було обрано MySQL 8.0 через його високу швидкодію, надійність та розширені можливості для оптимізації запитів. Версія 8.0 має нові функції та покращення в порівнянні з попередніми версіями, що полегшує роботу з базою даних [3].

Docker обраний для створення та управління контейнерами з метою забезпечення портативності та масштабованості розгорнутих застосунків. Використання Docker дозволяє легко переносити розроблене ПЗ між різними середовищами та швидко масштабувати інфраструктуру за потреби [4].

У якості фреймворку для розробки Back-End частини веб-системи був обраний Laravel. Тому що він є високорівневим фреймворком для розробки веб-застосунків з відкритим вихідним кодом, який використовує архітектурний шаблон MVC (Model-View-Controller). Він забезпечує розробникам зручний та ефективний спосіб створення веб-застосунків, розподіляючи логіку застосунку на три основні складові: модель (Model), представлення (View) та контролер (Controller). Laravel надає ряд інструментів та компонентів для швидкої розробки, таких як маршрутизація, робота з базами даних, шаблонізація та багато інших [5].

Ці технології були обрані через їхню популярність у веб-розробці, широкий функціонал, високу продуктивність та активну підтримку спільноти розробників, що можна побачити на рис. 1.4 графіку з рейтингом мов програмування-2024, який був взятий з сайту dev.ua.

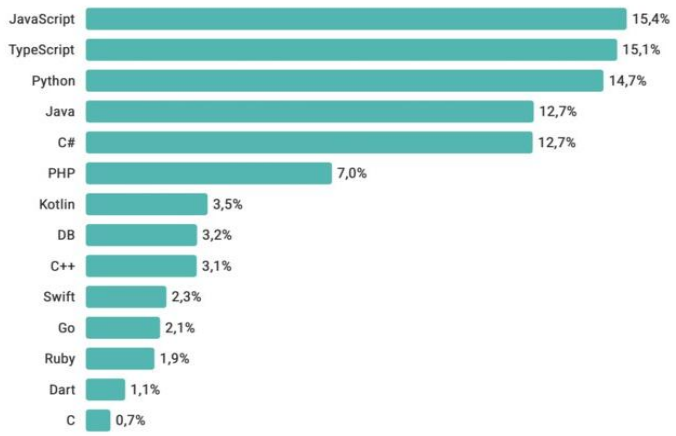


Рисунок 1.4. Рейтинг мов програмування у 2024 році

1.2 Проєктування веб-системи

1.2.1 Технічне завдання на розробку

Технічне завдання є важливим при розробці будь-якого проєкту, тому що воно забезпечує чітке розуміння вимог, цілей і технологій проєкту, що є ключовим для його успішного завершення. Тому для розробки Back-End частини автоматизованої веб-системи обліку руху контингенту здобувачів відділення комп'ютерних систем були поставлені певні функціональні вимоги.

Авторизація та аутентифікація:

Авторизація та аутентифікація є першим етапом при постановці технічного завдання. Важливість цих дій полягає у забезпеченні безпеки системи та захисті конфіденційності даних. Для цього передбачається створення декількох записів адміністраторів у базі даних та реалізація можливості авторизації у веб-застосунку за такими даними як: email та пароль. Такий підхід забезпечить безпеку та конфіденційність даних, а також попередить небажаний доступ до системи осіб, які не мають відповідних дозволів або авторизації.

CRUD-системи сутностей:

Система також має реалізувати CRUD-функціонал для керування цими сутностями. А саме забезпечити створення (Create), читання (Read), оновлення (Update) та видалення (Delete) даних, що дозволяє легко керувати даними у системі [7]. Відповідно веб-система, яка розробляється, повинна мати можливість додавати, переглядати, редагувати та видаляти інформацію про різні сутності, такі як студенти, групи, співробітники, накази, типи наказів, посади, навчальні заклади, батьки, спеціальності, освітні програми, галузі знань та пільги.

Система роутів:

Веб-сервіс вимагає створення певних роутів для управління ресурсами та сутностями у системі. Для основних сутностей системи мають бути передбачені окремі роути для пошуку за певними критеріями, щоб здійснювати динамічну

вибірку даних у формі через AJAX. Також мають бути передбачені роути для генерації файлів та їх завантаження.

Генерація файлів:

Наступними функціональними можливостями веб-системи є автоматизована генерація файлів у реальному часі, по останнім внесеним даним у сервіс, документів для студентів та групи, а саме таких документів як: картка студента, індивідуальний план, таблиці груп. Завантаження фотографій студентів та валідація всіх даних для сутностей.

Валідація:

Всі вхідні дані, будуть перевірятися на валідність та на відповідність вимогам безпеки за допомогою валідаційних правил Laravel на рівні реквестів.

Це включає:

- перевірка на коректність формату електронної пошти;
- перевірка на наявність спеціальних символів у паролі;
- перевірка на правильність введення числових значень;
- перевірка MIME-типів файлів;
- перевірка на розмір вхідних даних, а саме - розмір рядкових та числових даних, розміри файлів;
- перевірка на дійсність та існування даних у базі, задля забезпечення цілісності цих даних, тощо.

Налаштування доступів:

Сервіс має під собою наступне налаштування доступів для авторизованих та незареєстрованих користувачів на сайті, а саме – повна обмеженість неавторизованих користувачів. Всі роути для сторінок системи мають бути захищені на рівні мідлвару та перенаправляти користувача на сторінку авторизації, для них має бути недоступний перегляд файлів системи, які доступні для авторизованих користувачів системи. Авторизовані користувачі системи в свою чергу матимуть можливість вільно переходити за всіма роутами, передбаченими системою, та мати можливість генерувати файли для вивантаження особистих даних студентів та даних груп. Для неіснуючих

сторінок має бути передбачено повернення статусу 404 та редирект на відповідну сторінку веб-застосунку.

1.2.2 Проєктування структури бази даних веб-застосунку

Першим кроком у проєктуванні структури бази даних є аналіз вимог до системи. Необхідно детально вивчити функціональні та нефункціональні вимоги до бази даних, що включають в себе:

- опис основних сутностей та їх атрибутів;
- визначення взаємозв'язків між сутностями;
- вимоги до забезпечення безпеки даних;
- потреби у забезпеченні швидкості та продуктивності бази даних;
- інші специфічні вимоги, що впливають на структуру бази даних [8].

Почнемо з розгляду основної сутності студента та одразу визначимо, які атрибути необхідні цієї сутності. Студент повинен мати обов’язкові поля, такі як ім’я, прізвище, по батькові та стать, як будь-яка інша особа. Також таблиця студента має зберігати наступні особисті дані: номер телефону, електронну пошту. Крім цього, важливим аспектом є зберігання даних про місце проживання, громадянство та дату народження студента. Усі ці поля є критично важливими для забезпечення точного обліку та ідентифікації студентів.

Для забезпечення ефективного управління академічною діяльністю студентів, таблиця містить поля, що вказують на групу, до якої належить студент, форму фінансування та форму навчання, тип зарахування, дату вступу та курс, на який студент зарахований, освітню програму та освітній ступінь. Враховано специфічні потреби студентів в гуртожитку, академічна відпустка та статус старости групи. Зберігання інформації про наявність пільг дозволяє ефективно керувати та контролювати використання цих ресурсів. Також, для поліпшення взаємодії та комунікації, зберігаються унікальні контактні дані кожного студента.

З метою забезпечення цілісності та взаємозв'язку даних у базі, таблиця студентів містить зовнішні ключі, що посилаються на таблиці груп, навчальних

закладів та пільг. Ці зовнішні ключі не тільки підтримують цілісність даних, але й забезпечують можливість каскадного оновлення та видалення пов'язаних записів, що є важливим аспектом при адмініструванні бази даних.

Ця структура таблиці студентів, що зображена у табл. 1.2 не лише забезпечує повний та детальний облік кожного студента, що зберігаються в системі.

Таблиця 1.2. Структура таблиці “Студенти”

Назва поля	**Тип даних**	**Унікальне?**	**Може бути пустим?**	**Ключ / Індекс**
ID	BIGINT	Так	Ні	Первинний
group_id	BIGINT	Ні	Ні	Зовнішній
educational_institute_id	BIGINT	Ні	Ні	Зовнішній
educational_program_id	BIGINT	Ні	Ні	Зовнішній
educational_degrees	INT	Ні	Ні	
benefits_id	BIGINT	Ні	Так	Зовнішній
firstname	CHAR	Ні	Ні	Індекс
lastname	CHAR	Ні	Ні	Індекс
surname	CHAR	Ні	Ні	Індекс
phone	CHAR	Так	Ні	
email	CHAR	Так	Ні	
is_group_leader	BOOLEAN	Ні	Ні	
is_academic_vacation	BOOLEAN	Ні	Ні	
need_dormitory	BOOLEAN	Ні	Ні	
entry_date	DATE	Ні	Ні	
enrolled_in_the_course	INT	Ні	Ні	
financing_form	INT	Ні	Ні	
education_form	INT	Ні	Ні	

Продовження таблиці 1.2

Назва поля	Тип даних	Унікальне?	Може бути пустим?	Ключ / Індекс
enrollment_type	INT	Ні	Ні	
birthday	DATE	Ні	Ні	
gender	INT	Ні	Ні	
citizenship	CHAR	Ні	Ні	
place_of_residence	CHAR	Ні	Ні	
graduation_date	DATE	Ні	Ні	
language	INT	Ні	Ні	
timestamps	DATETIME	Ні	Так	

Всі документи студентів, окрім студентський квитків, зберігаються у відповідній таблиці “Документи” – табл. 1.3. А студентські квитки в свою чергу у табл. 1.4.

Таблиця 1.3 Структура таблиці “Документи студента”

Назва поля	Тип даних	Унікальне?	Може бути пустим?	Ключ / Індекс
ID	BIGINT	Так	Ні	Первинний
student_id	BIGINT	Ні	Ні	Зовнішній
document_type	INT	Ні	Ні	Індекс
document_number	CHAR	Ні	Ні	Індекс
document_series	CHAR	Ні	Так	Індекс
issued_by	INT	Ні	Так	
issue_date	DATE	Ні	Так	
valid_until_date	DATE	Ні	Так	
timestamps	DATETIME	Ні	Так	

Таблиця 1.4 Структура таблиці “Студентські білети”

Назва поля	Тип даних	Унікальне?	Може бути пустим?	Ключ / Індекс
ID	BIGINT	Так	Ні	Первинний
student_id	BIGINT	Ні	Ні	Зовнішній
document_number	CHAR	Ні	Ні	Індекс
issue_date	DATE	Ні	Так	
valid_until_date	DATE	Ні	Так	
timestamps	DATETIME	Ні	Так	

Кожен батько, мати або опікун мають бути зареєстровані у системі. Таблиця батьків у табл. 1.5 містить важливі контактні дані, такі як ПІБ окремо, номер телефону та електронну пошту, для забезпечення ефективної комунікації. Також зберігаються дані про місце проживання кожного з батьків, що важливо для екстрених контактів та адміністративних цілей.

З метою забезпечення цілісності та взаємозв'язку даних у базі, встановлюється зв'язок між таблицями студентів та батьків через зв'язок багато до багатьох. Це означає, що кожен студент може мати декілька батьків, а кожен батько або мати може бути зареєстрований для декількох студентів. Саме тому потрібно розуміти ким приходиться особа для поточного студента. Тому в проміжковий таблиці – табл. 1.6 є додаткове поле статус, задля забезпечення такої можливості.

Таблиця 1.5. Структура таблиці “Батьки”

Назва поля	Тип даних	Унікальне?	Може бути пустим?	Ключ / Індекс
ID	BIGINT	Так	Ні	Первинний
firstname	CHAR	Ні	Ні	Індекс
lastname	CHAR	Ні	Ні	Індекс
surname	CHAR	Ні	Ні	Індекс

Продовження таблиці 1.5

Назва поля	Тип даних	Унікальне?	Може бути пустим?	Ключ / Індекс
phone	CHAR	Так	Ні	
email	CHAR	Так	Так	
gender	INT	Ні	Ні	Індекс
place_of_residecne	CHAR	Ні	Ні	
timestamps	DATETIME	Ні	Так	

Таблиця 1.6. Структура таблиці “Батьки-Студенти”

Назва поля	Тип даних	Унікальне?	Може бути пустим?	Ключ / Індекс
ID	BIGINT	Так	Ні	Первинний
student_id	BIGINT	Ні	Ні	Зовнішній
parent_id	BIGINT	Ні	Ні	Зовнішній
status	INT	Ні	Ні	Індекс
timestamps	DATETIME	Ні	Так	

Всі та подальші таблиці відповідають певним нормальним формам, які допомагають структурувати дані для забезпечення їх цілісності та зменшення надмірності, а саме:

Перша нормальна форма, що вимагає, щоб усі атрибути таблиці були атомарними значеннями. У моїх попередніх і всіх наступних таблицях - кожне поле містить одне значення, тому таблиці знаходиться в першій нормальній формі.

Друга нормальна форма вимагає, щоб таблиця була в першій нормальній формі і всі неключові атрибути були повністю залежними від усього первинного ключа. Первинниими ключами в більшості таблиць є саме поле id, і всі інші поля залежать безпосередньо від нього, що відповідає другій нормальній формі.

Третя нормальна форма вимагає, щоб таблиця була в другій нормальній формі, і всі неключові атрибути не мали транзитивної залежності від первинного ключа. Тобто, атрибут не повинен залежати від іншого атрибуту, який, у свою чергу, залежить від первинного ключа. У таблицях всі неключові поля залежать безпосередньо від первинного ключа id і не мають транзитивної залежності. Отже, таблиці знаходиться в третій нормальній формі.

Перейдемо безпосередньо до навчальної частини системи та тепер почнемо розглядати, починаючи з найголовнішої таблиці, а саме табл. 1.7 – "Галузі знань".

Таблиця "Галузі знань" зберігає інформацію про різні галузі знань, що викладаються в навчальних закладах. Ця таблиця має наступні поля - унікальний ідентифікатор галузі знань, назва галузі знань, номер галузі знань. Унікальні поля name та number у таблиці "Галузі знань" гарантують, що кожна галузь знань буде представлена лише один раз у базі даних. Це відповідає певним нормальним формам, які допомагають структурувати дані для забезпечення їх цілісності та зменшення надмірності, а саме:

Всі поля таблиці атомарні, тому таблиця знаходиться в першій нормальній формі. Первинний ключ таблиці "Галузі знань" – це поле id, і всі інші поля, а саме name та number залежать безпосередньо від нього, що відповідає 2 нормальній формі. У таблиці "Галузі знань" поля name та number залежать безпосередньо від первинного ключа id і не мають транзитивної залежності. Отже, таблиця знаходиться в третій нормальній формі.

Таблиця 1.7 Структура таблиці "Галузі знань"

Назва поля	**Тип даних**	**Унікальне?**	**Може бути пустим?**	**Ключ**
ID	BIGINT	Так	Ні	Первинний
name	CHAR	Так	Ні	
number	INT	Так	Ні	
timestamps	DATETIME	Ні	Так	

Ця таблиця пов’язана з таблицею “Спеціальності” зв’язком один до багатьох, для дотримання ієрархічної системи, тому що, одна галузь знань може мати декілька спеціальностей, як в свою чергу можуть мати декілька освітніх програм. Табл. 1.8 – “Спеціальності” зберігає інформацію про спеціальності, що пропонуються різними галузями знань, містить наступні поля – унікальний ідентифікатор, унікальний ідентифікатор галузі знань, назва спеціальності номер спеціальності.

Таблиця 1.8. Структура таблиці “Спеціальності”

Назва поля	**Тип даних**	**Унікальне?**	**Може бути пустим?**	**Ключ**
ID	BIGINT	Так	Ні	Первинний
branch_knowledge_id	BIGINT	Ні	Ні	Зовнішній
name	CHAR	Так	Ні	
number	INT	Так	Ні	
timestamps	DATETIME	Ні	Так	

Наступна сутність – “Освітні програми”, має мати свою таблицю – табл. 1.9, яка містить ті самі поля, що і таблиця “Галузі знань” з єдиною відмінністю, замість поля branch_knowledge_id, яке посилається на таблицю “Галузі знань” буде поле specialty_id, яке посилається на таблицю “Спеціальності”, задля забезпечення зв’язку один до багатьох між цими двома таблицями.

Таблиця 1.9. Структура таблиці “Освітні програми”

Назва поля	**Тип даних**	**Унікальне?**	**Може бути пустим?**	**Ключ**
ID	BIGINT	Так	Ні	Первинний
branch_knowledge_id	BIGINT	Ні	Ні	Зовнішній
name	CHAR	Так	Ні	
number	INT	Так	Ні	
timestamps	DATETIME	Ні	Так	

Табл. 1.10 – “Групи” повинна зберігати унікальний ідентифікатор групи, назва групи. Унікальний код або номер групи, дату формування групи, та дату випуску групи та зовнішній ключ, який посилається на таблицю "Співробітники" та вказує на куратора групи. Кожна група має одного куратора і кожен куратор має одну групу.

Таблиця 1.10. Структура таблиці “Групи”

Назва поля	**Тип даних**	**Унікальне?**	**Може бути пустим?**	**Ключ**
ID	BIGINT	Так	Ні	Первинний
curator_id	BIGINT	Так	Ні	Зовнішній
code	CHAR	Так	Ні	
started_date	DATE	Ні	Ні	
finished_date	DATE	Ні	Ні	
timestamps	DATETIME	Ні	Так	

Як видно з таблиці “Групи”, вона містить поле curator_id, яке відповідає ідентифікатору співробітника, який керує поточну групу. Тобто групи, мають бути пов’язані з таблицею “Співробітники” зв’язком один до одного по полю curator_id. Таблиця “Співробітники” має такі самі поля як і таблиця батьків
В результаті маю таку схему для таблиці“Співробітники” – табл. 1.11.

Таблиця 1.11. Структура таблиці “Співробітники”

Назва поля	**Тип даних**	**Унікальне?**	**Може бути пустим?**	**Ключ / Індекс**
ID	BIGINT	Так	Ні	Первинний
firstname	CHAR	Ні	Ні	Індекс
lastname	CHAR	Ні	Ні	Індекс
surname	CHAR	Ні	Ні	Індекс
phone	CHAR	Так	Ні	

Продовження таблиці 1.11

Назва поля	Тип даних	Унікальне?	Може бути пустим?	Ключ / Індекс
email	CHAR	Так	Так	
gender	INT	Ні	Ні	Індекс
place_of_residecne	CHAR	Ні	Ні	
timestamps	DATETIME	Ні	Так	

Посади співробітників будуть зберігатись в окремій таблиці “Посади” та пов’язані з таблицею співробітників через проміжкову таблицю “Співробітники-посади”, задля забезпечення можливості вибору декількох посад для співробітника. Попарно поля employee_id та position_id складають унікальне значення, що означає що в одного співробітника не може бути двох записів з однаковою посадою.

Ще одна складова системи – “Накази”, які відображені у табл. 1.12, вони будуть зберігаються в доволі простій таблиці, так як здебільшого несуть базовий інформативний характер, а саме відображення номерів наказів та їх типів, які також відокремлено у окрему таблицю “Типи наказів” – табл. 1.13, для студента та груп. “Накази” зв’язані з таблицями “Студенти” через проміжкову таблицю “Студенти-накази”, яка відображена у табл. 1.14, та “Групами” через проміжкову таблицю “Групи-накази” – табл. 1.15, задля забезпечення зв’язків багато до багатьох. Також “Накази” матиме зв’язок з таблицею “Типи наказів” по полю decree_type_id – зв’язком один до багатьох.

Таблиця 1.12 Структура таблиці “Накази”

Назва поля	Тип даних	Унікальне?	Може бути пустим?	Ключ
ID	BIGINT	Так	Ні	Первинний
decree_type_id	BIGINT	Так	Ні	Зовнішній
number	CHAR	Так	Ні	

Продовження таблиці 1.12

Назва поля	Тип даних	Унікальне?	Може бути пустим?	Ключ
signing_date	DATE	Ні	Ні	
from_which_date	DATE	Ні	Ні	
timestamps	DATETIME	Ні	Так	

Таблиця 1.13 Структура таблиці “Типи наказів”

Назва поля	Тип даних	Унікальне?	Може бути пустим?	Ключ
ID	BIGINT	Так	Ні	Первинний
name	CHAR	Так	Ні	
slug	CHAR	Так	Ні	
lock	BOOLEAN	Ні	Ні	
timestamps	DATETIME	Ні	Так	

Таблиця 1.14 Структура таблиці “Студенти-накази”

Назва поля	Тип даних	Унікальне?	Може бути пустим?	Ключ
ID	BIGINT	Так	Ні	Первинний
student_id	BIGINT	Ні	Ні	Зовнішній
decree_id	BIGINT	Ні	Ні	Зовнішній
timestamps	DATETIME	Ні	Так	

Таблиця 1.15 Структура таблиці “Групи-накази”

Назва поля	Тип даних	Унікальне?	Може бути пустим?	Ключ
ID	BIGINT	Так	Ні	Первинний
group_id	BIGINT	Ні	Ні	Зовнішній
decree_id	BIGINT	Ні	Ні	Зовнішній

Продовження таблиці 1.15

Назва поля	Тип даних	Унікальне?	Може бути пустим?	Ключ
timestamps	DATETIME	Ні	Так	

На основі аналізу вимог та аналізу кожної сутності, які були описані вище для бази даних розроблена структура ER-схема “Пташина лапка”, що відображає в собі сутності та взаємозв'язки між ними, її зображено на рис. 1.5. При аналізі кожної сутності, було враховано принципи нормалізації даних для забезпечення ефективної організації інформації та приведення таблиць до перших трьох нормальних форм. В разі коли денормалізація даних була запланованою, тоді і тільки тоді можливо відійти від однієї або декількох нормальних форм.

Крім сутностей, які безпосередньо пов’язані з обліком студентів, є такі сутності, які відображені на рис. 1.6, як “Файли”, “Адміністратори”, “Таблиця міграцій”. Розберемо кожну з них:

Адміністратори повинні мати такі обов’язкові поля, як адреса електронної пошти та пароль. Крім цих даних в адміністратора можуть бути такі необов’язкові дані як: ім’я, прізвище, номер телефону.

Розглянемо сутність “Файли” – табл. 1.16, для визначення списку полів, які повинні бути в сутності файлу. По перше файл, має мати свій ідентифікатор та шлях до самого файлу. Для більшої динамічності та можливості зв'язати файл з будь якою сутністю (Моделлю) зв’язок файлів з ними буде здійснюватись через морфічний зв’язок по таким полям як: model_type та model_id. Також сутність має зберігати дані про диск, через який був збережений файл. Окрім цих полів таблиця файлів має зберігати таку обов’язкову інформацію про файл: назву файлу, mime-тип та розмір. Також повинна бути можливість зберігати необов’язкові параметри, такі як опис файлу, тип файлу, відображаємо назва файлу, статус (приватний або публічний), колекція до якої він відноситься, тег для файлу, бо якому можливо буде у подальшому дістати файл, або декілька файлів, а також таймс темпи, які будуть зберігати інформацію про те, коли саме був створений або оновлений файл.

Таблиця 1.16. Структура таблиці “Файли”

Назва поля	**Тип даних**	**Унікальне?**	**Може бути пустим?**	**Ключ**
ID	BIGINT	Так	Ні	Первинний
model_type	CHAR	Ні	Ні	Індекс
model_id	BIGINT	Ні	Ні	Індекс
disk	CHAR	Ні	Ні	Індекс
storage_path	CHAR	Так	Ні	
name	CHAR	Ні	Так	
type	CHAR	Ні	Так	
file_description	CHAR	Ні	Так	
collection	CHAR	Ні	Так	
tag	CHAR	Ні	Так	
status	CHAR	Ні	Ні	
file_name	CHAR	Ні	Ні	
mime_type	CHAR	Ні	Ні	
size	CHAR	Ні	Ні	
timestamps	DATETIME	Ні	Так	

Таблиця міграції зберігає ідентифікатор міграції, назву міграції, та номер порції в яку потрапила міграція.

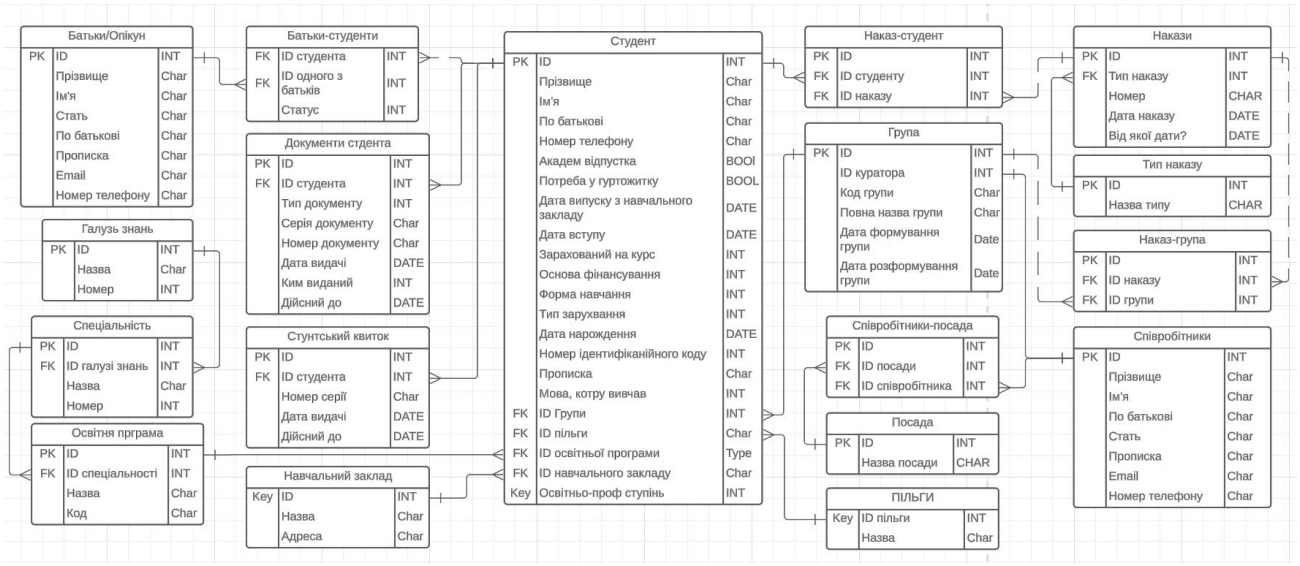


Рисунок 1.5. Схема взаємозв'язків між базовими сутностями сервісу

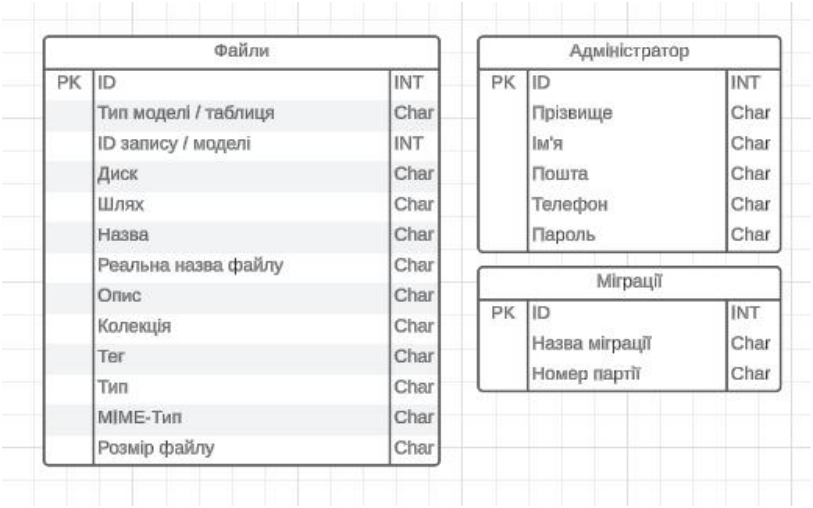


Рисунок 1.6. Схема базових таблиць веб-застосунку

1.2.3 Проєктування архітектури веб-системи

Застосунок, що розробляється, буде проєктуватися на архітектурі MVC за допомогою фреймворку Laravel [7]. Дана архітектура є однією з найпоширеніших у світі розробки веб-застосунків, оскільки вона надає простий механізм створення застосунків.

Кожен запит, що надходить до застосунку, спочатку проходить через веб-сервер Nginx, який відповідає за обробку HTTP-запитів. Nginx передає ці запити далі до PHP, де вони обробляються Laravel. Перед тим, як досягти контролера, запит проходить через шар проміжного програмного забезпечення (middleware). Middleware використовується для різних задач, таких як перевірка авторизації,

логування, кешування тощо. В даному випадку він забезпечує перевірку авторизації користувача, що є критично важливим для безпеки застосунку.

Контролери є центральною частиною обробки запитів у Laravel. Однак, у моїй архітектурі контролери не обробляють дані напряму. Вони взаємодіють з кастомними реквестами, які відповідають за валідацію даних та надають методи для їх отримання. Це дозволяє забезпечити чистоту коду та його відповідність принципу єдиного обов'язку (Single Responsibility Principle). Використання кастомних реквестів гарантує, що лише валідні дані потрапляють на рівень сервісів та моделей, запобігаючи можливим помилкам та забезпечуючи цілісність даних.

Сервіси відповідають за бізнес-логіку застосунку, отримують валідовані дані від контролерів та обробляють їх відповідно до бізнес-правил. Для взаємодії з базою даних використовуються кастомні репозиторії. В моєму контексті репозиторієм будемо називати прошарок над моделями Eloquent, що дозволяє здійснювати операції з базою даних більш контрольованим чином. Основна ідея використання моїх репозиторіїв полягає у створенні моделей з обов'язковими параметрами, що гарантує коректність створення записів у базі даних. Це важливо, оскільки Eloquent (Active Record) не завжди забезпечує необхідний рівень контролю створення моделей, так як сам по собі Active Record порушує деякі принципи ООП, та деякі принципи SOLID, а саме Single Responsibility Principle (величезна кількість методів, не всі з яких є відповідальністю моделі), Open/Closed Principle (при потребі додати нову бізнес логіку необхідно міняти модель та додавати нові методу), Interface Segregation Principle (абстракції Laravel мають величезну кількість методів, які не завжди потрібні для роботи), Dependency Inversion Principle (Laravel зав’язаний на статиці і майже все обертається навколо моделі, тому стає важко впроваджувати принцип DI). Кастомні репозиторії дозволяють централізовано керувати взаємодією з базою даних, забезпечуючи безпечне отримання та запис даних, їх стабільність, завдяки тому, що я обмежую можливі місця “пострілу у ногу”.

Моделі в Laravel реалізовані за допомогою ORM Eloquent. Вони представляють собою структури даних та їхню взаємодію з базою даних. Завдяки використанню кастомних репозиторіїв, створення та оновлення моделей здійснюється уніфіковано та з дотриманням всіх необхідних бізнес-правил.

Архітектура, яка відображена на рис. 1.7 забезпечує чітке розділення відповідальностей між різними компонентами системи. Використання middleware, кастомних реквестів, сервісів та репозиторіїв для дозволяє створювати надійний, масштабований та підтримуваний код, що полегшує його подальший розвиток і підтримку.

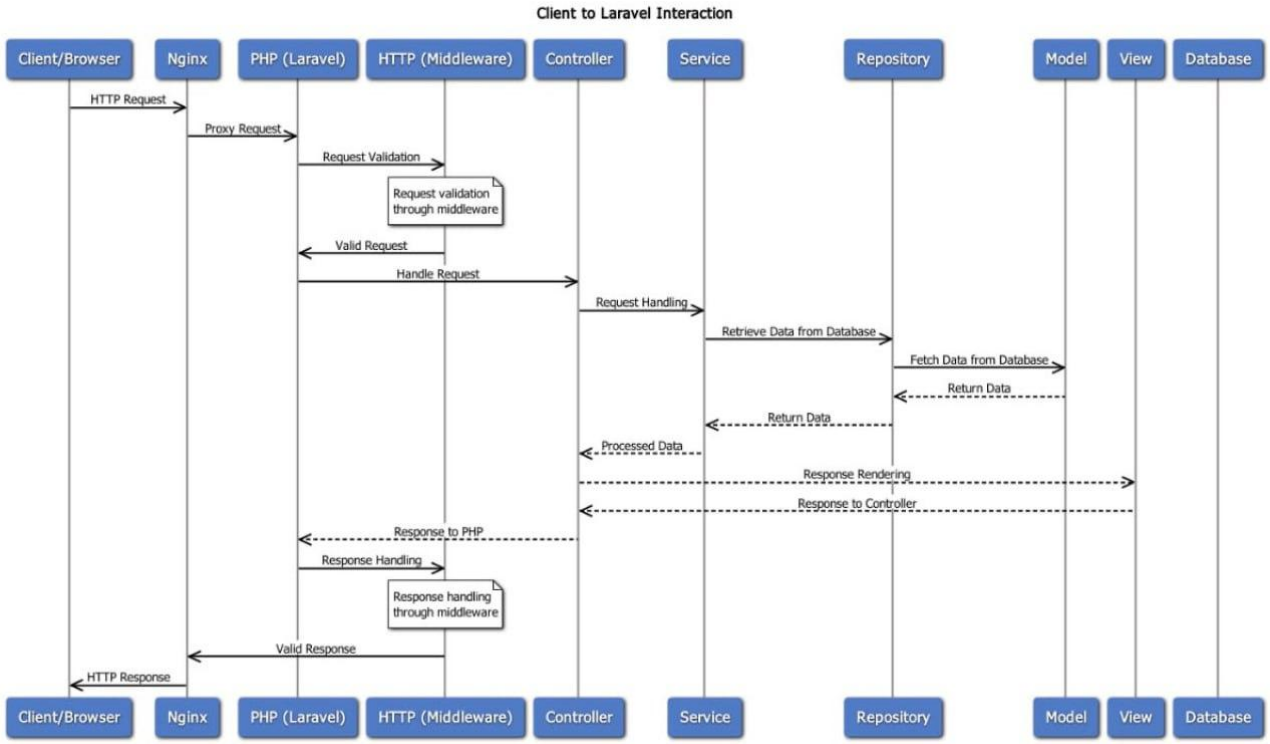


Рисунок 1.7. Архітектура веб-застосунку

1.3 Реалізація веб-системи

1.3.1 Конфігурація оточення

Для того щоб налаштувати свою Docker-збірку, яка буде піднімати такі контейнери, яка зображені на Docker-схемі на рис. 1.8, а саме – веб-сервер Nginx, мову програмування PHP, та бази даних MySQL з певними версіями, я створив

файл docker-compose.yml, який безпосередньо буде виковуватись при піднятті контейнерів.

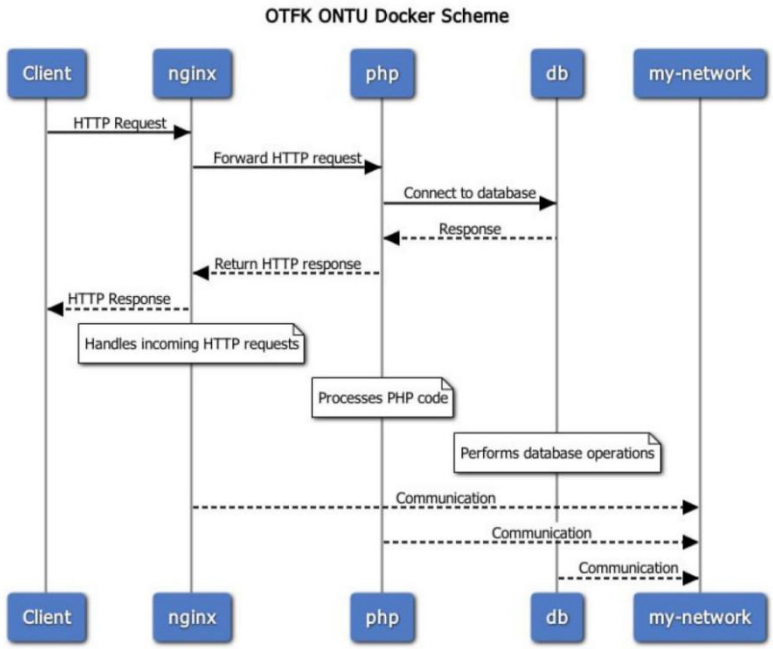


Рисунок 1.8. Docker-схема веб-застосунку

Також я створив директорію docker, яка містить наступні вкладені директорії – рис. 1.9. в яких додані певні файли для конфігурації.

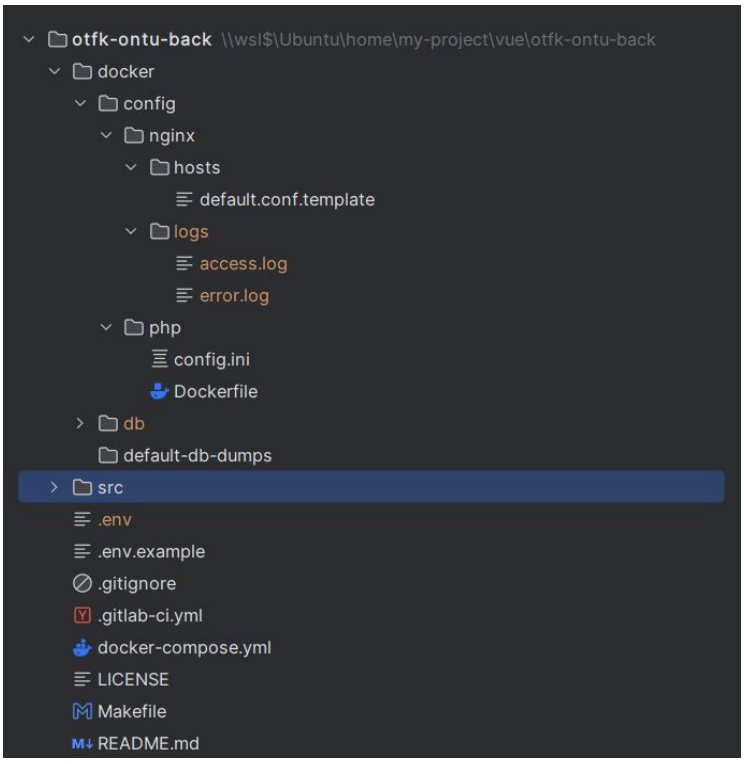


Рисунок 1.9. Базова структура директорій для налаштування Docker

Розглянемо структуру файлу docker-compose.yml, яка наведена нижче:

```
version: '3'
services:
    nginx:
        image: nginx:latest
        container_name: nginx_${COMPOSE_PROJECT_NAME}
        ports:
            - "${DOCKER_WEB_PUBLIC}:80"
        volumes:
            - ${DOCKER_CODE_FOLDER}:${APP_PATH}
            - ./docker/config/nginx/logs:/var/log/nginx
            - ./docker/config/nginx/hosts:/etc/nginx/templates/
        environment:
            WWW_ROOT: ${APP_PATH}/${WWW_FOLDER}
        networks:
            - my-network
    db:
        image: mysql:8.0
        container_name: mysql_${COMPOSE_PROJECT_NAME}
        hostname: db
        volumes:
            - "${DOCKER_DB_FILES}:/var/lib/mysql"
            - ./docker/default-db-dumps:/docker-entrypoint-initdb.d
            - /var/run/mysqld
        environment:
            MYSQL_ROOT_PASSWORD: "${DOCKER_DB_ROOT_PWD}"
            MYSQL_USER: "${DB_USERNAME}"
            MYSQL_PASSWORD: "${DB_PASSWORD}"
            MYSQL_DATABASE: "${DB_DATABASE}"
            TZ: "Europe/Kiev"
        ports:
            - "${DOCKER_DB_PUBLIC}:3306"
        networks:
            - my-network
    php:
        container_name: php_${COMPOSE_PROJECT_NAME}
        build:
            context: ./
            dockerfile: ./docker/config/php/Dockerfile
            args:
                APP_PATH: ${APP_PATH}
        volumes:
            - ${DOCKER_CODE_FOLDER}:${APP_PATH}
            - ./docker/config/php/config.ini:/usr/local/etc/php/conf.d/config.ini
        hostname: php
        environment:
            TZ: Europe/Kiev
            APP_PATH: ${APP_PATH}
            PHP_IDE_CONFIG: "serverName=Docker"
            XDEBUG_MODE: ${XDEBUG_MODE}
        networks:
            - my-network
networks:
    my-network:
        driver: bridge
```

Nginx, який використовується в якості веб-сервера, обробляє запити і відповідає на них, він використовує офіційний образ nginx:latest, що відповідає

останній стабільний версії Nginx. До контейнеру мапиться, тобто співставляється, зовнішній порт ${DOCKER_WEB_PUBLIC} хоста, який бере значення порту зі змінних оточення, на 80-ий порт контейнера Nginx. Змінна оточення WWW_ROOT вказує на кореневу директорію веб-сервера. А трохи нижче налаштований шлях до файлів конфігурації. Розглянемо файл, що містить в собі дефолтний конфіг docker/config/hosts/default.conf.template.

```
server {
    index index.php;
    error_log  /var/log/nginx/error.log;
    access_log /var/log/nginx/access.log;
    root ${WWW_ROOT};
    client_max_body_size 32M;
    location / {
        try_files $uri $uri/ /index.php$is_args$args;
    }
    location ~ \.php$ {
        try_files $uri =404;
        fastcgi_split_path_info ^(.+\.php)(/.+)$;
        fastcgi_pass php:9000;
        fastcgi_index index.php;
        include fastcgi_params;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        fastcgi_param PATH_INFO $fastcgi_path_info;
    }
}
```

У розділі налаштувань веб-сервера Nginx ми встановлюємо параметри, які регулюють його роботу. Нижче розглянемо основні моменти з коду.

Перший рядок вказує на те, що файл index.php є головним файлом, тобто файлом для входу у веб-застосунок, який сервер буде використовувати, якщо не вказано інший файл, який зберігається у директорії public. Нижче вказані шляхи до файлів логів, куди будуть записуватися помилки сервера та журнали доступів - інформація про те, які клієнти отримують доступ до сервера.

${WWW_ROOT}; визначає кореневу директорію сервера, де будуть знаходитись файли веб-сайту. Сам шлях до цієї директорії динамічно передається зі змінних оточення, що є гарною практикою при будуванні контейнерів.

У блоці “location /” задається обробка запитів до кореневої URL. За допомогою директиви try_files веб-сервер спочатку спробує знайти файл у вказаному URI ($uri), якщо не вдасться - у каталозі ($uri/). Якщо файл не

знайдено, запит буде направлено на index.php, додаючи до нього будь-які аргументи запиту (is_argsargs).

У блоці “location ~ \.php$” визначається обробка запитів до файлів PHP. Директива try_files спробує знайти запитаний файл, якщо ні - буде повернено код помилки 404. Після цього використовується fastcgi_split_path_info, яка розбиває шлях до скрипту PHP на дві частини: ім'я файлу та додаткову інформацію. Далі, запит буде переданий на сервер PHP за допомогою протоколу FastCGI, який працює на порту 9000, згідно з директивою fastcgi_pass. Для обробки цього запиту, використовується індексний файл index.php, а також включаються параметри FastCGI та передаються шлях до скрипту та додаткова інформація через змінні SCRIPT_FILENAME та PATH_INFO відповідно.

MySQL використовується в якості сервера баз даних MySQL, в межах мережі він описаний як сервіс db. Для нього використовується образ mysql:8.0 та мапиться порт ${DOCKER_DB_PUBLIC} хоста на порт 3306 контейнера MySQL. За таким самим принципом, як для Nginx, прописані volumes для зберігання даних бази даних MySQL та ініціалізації початкових даних. Зі змінних оточення для контейнеру встановлюються налаштування користувача, пароля, назви бази даних і часового поясу.

PHP використовується для виконання PHP-коду застосунку. Для конфігурації контенеру використовується Dockerfile, задля збірки власного образу PHP. Так само мапиться директорія з кодом застосунку та конфігураційний файл config.ini для PHP та встановлюються змінні оточення для налаштування часового поясу, шляху до проєкту, налаштування Xdebug та іншого.

У директорії php знаходяться два файли - config.ini, який містить в собі налаштування для PHP, та Dockerfile з безпосередньо налаштуваннями контейнеру PHP. Нижче наведено зміст цього файлу.

```
FROM php:8.2-fpm

ARG APP_PATH

RUN apt-get -y update && apt-get -y install git unzip vim libzip-dev libpng-dev
```

```
# Install PHP extensions
RUN docker-php-ext-install pdo pdo_mysql bcmath gd zip exif

# Install Xdebug
RUN pecl install xdebug \
    &&    echo    "zend_extension=$(find    /usr/local/lib/php/extensions/    -name
xdebug.so)" > /usr/local/etc/php/conf.d/xdebug.ini \
    && echo "xdebug.client_port=9000" >> /usr/local/etc/php/conf.d/xdebug.ini \
    && echo "xdebug.mode=develop,debug" >> /usr/local/etc/php/conf.d/xdebug.ini \
    &&              echo              "xdebug.start_with_request=yes"              >>
/usr/local/etc/php/conf.d/xdebug.ini \
    &&              echo              "xdebug.discover_client_host=0"              >>
/usr/local/etc/php/conf.d/xdebug.ini \
    &&          echo          "xdebug.client_host=host.docker.internal"          >>
/usr/local/etc/php/conf.d/xdebug.ini

# Install Composer
RUN  php  -r  "readfile('http://getcomposer.org/installer');"  |  php  --  --install-
dir=/usr/bin/ --filename=composer

WORKDIR $APP_PATH

CMD ["php-fpm"]
```

Тут вказано базовий образ контейнеру “FROM php:8.2-fpm:”, на основі якого буде побудований кастомний контейнер для PHP, в моєму випадку - php:8.2-fpm, що містить PHP версії 8.2 з підтримкою FastCGI Process Manager (FPM), та APP_PATH базовий шлях до проєкту, який задається в env, які я розгляну нижче. Рядок RUN apt-get вказує на те, які пакети має оновити пакетний менеджер APT для роботи з застосунком.

Наступний декілька рядків вказують на те, що потрібно встановити розширення Xdebug для PHP, яке дозволить дебажити код у реальному часі, що прискорить процес розробки. Xdebug потребує додаткового налаштування в IDE, а саме створення сервісу Docker та мапплення локального шляху та шляху в контейнері – рис 1.10, задля правильної роботи xdebug.

Composer є важливим інструментом у проєкті, оскільки він автоматично завантажує та встановлює необхідні бібліотеки, забезпечуючи управління залежностями та контроль версій. Встановлення Composer гарантує сумісність компонентів і допомагає уникнути конфліктів версій. Крім того, Composer спрощує процес оновлення залежностей та налаштовує автозавантаження класів, що значно підвищує ефективність роботи над проєктом [9].

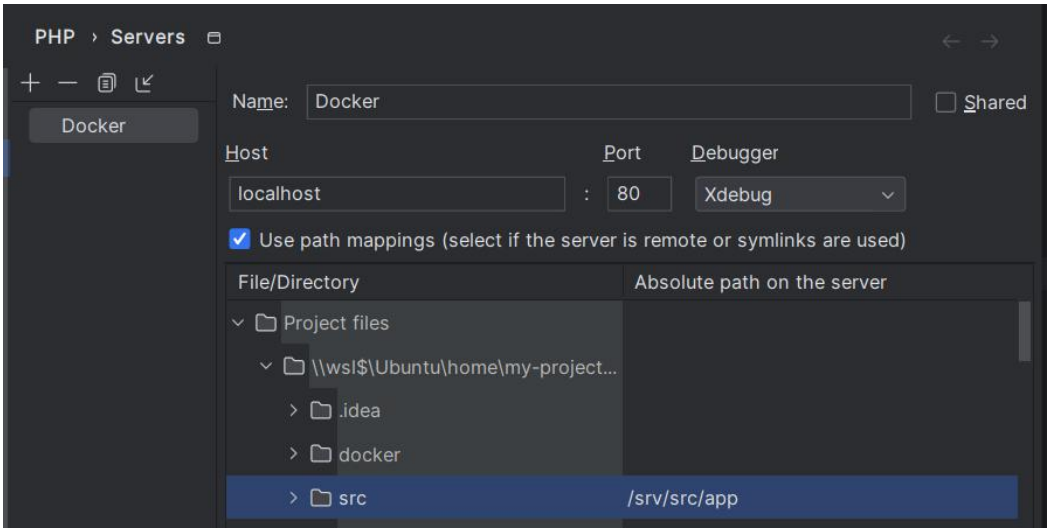


Рисунок 1.10. Налаштування Xdebug

Всі сервіси об'єднані в мережу my-network, яка використовується для забезпечення комунікації між контейнерами. В ній використовується мережевий драйвер bridge, який дозволяє контейнерам з'єднуватися в локальну мережу на рівні IP-адрес та забезпечує автоматичне призначення IP-адрес контейнерам а також дозволяє контейнерам спілкуватися між собою та зовнішніми мережевими ресурсами, а також забезпечує ізоляцію контейнерів у власній віртуальній мережі.

Після конфігурації Docker-збірки, контейнери можна підняти через командний рядок за допомогою наступної команди docker compose up --build -d, результат виконаний відображений на рис. 1.11.

Сам фреймворк знаходиться у директорії src задля можливості легко винести код, без необхідності відокремлення певних частин, в разі коли необхідно підняти проєкт без Docker.

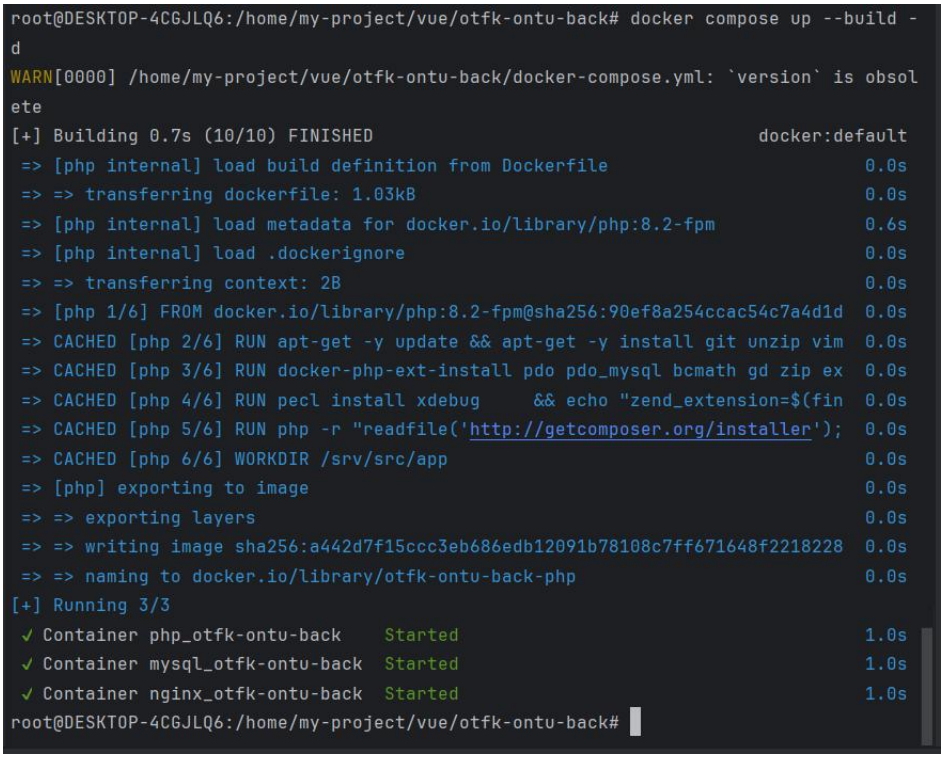


Рисунок 1.11. Результат підняття контейнерів

1.3.2 Створення міграцій

На основі всіх налаштувань середовища налаштовуємо змінні оточення безпосередньо для Laravel, а саме ті змінні оточення, що стосуються налаштування бази даних. Неведу їх нижче.

```
DB_CONNECTION=mysql
DB_HOST=db
DB_PORT=3306
DB_DATABASE=app
DB_USERNAME=app
DB_PASSWORD=secret
```

Для коректної роботи Eloquent потрібно вказати DB_CONNECTION для того, щоб вказати до якої бази даних буде підключений застосунок, в нашому випадку - MySQL. Також вказується хост та порт за якими доступна база даних, в нашому випадку хостом виступає внутрішнє ім’я Docker-контейнеру. Після чого вказується ім’я юзера, назва бази даних та пароль за якими потрібно приєднатися до бази даних.

По схемі бази даних, частини якої представлені на рис. 1.5. та рис. 1.6 будую міграції, повний список яких відображений на рис. 1.12.

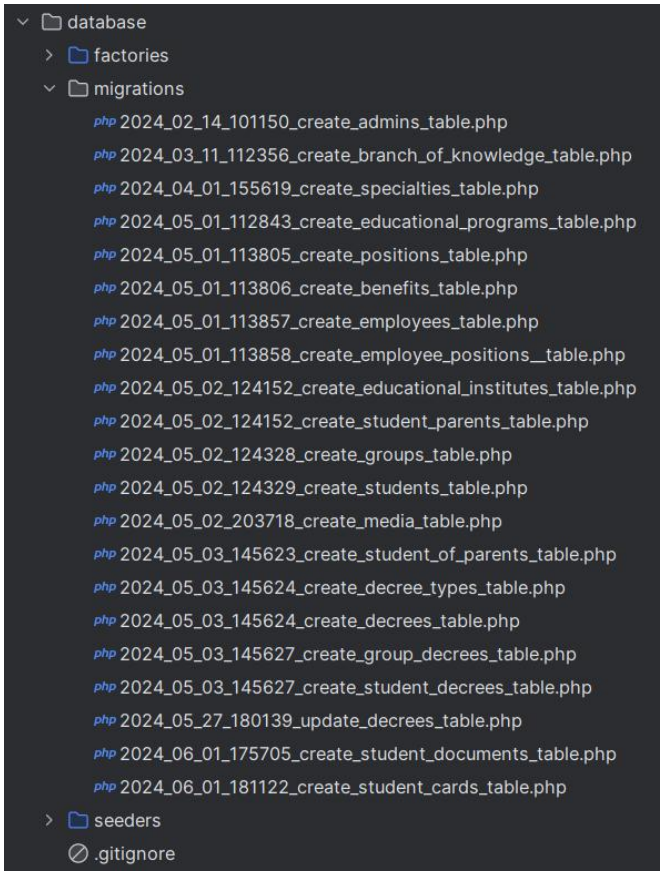


Рисунок 1.12. Створені міграції

Розглянемо типову міграцію Eloquent на прикладі міграції для моделі “Студент”, код наведу нижче.

```
<?php

use App\Models\Enums\EducationalDegrees;
use App\Models\Enums\EducationForms;
use App\Models\Enums\EnrollmentType;
use App\Models\Enums\FinancingForms;
use App\Models\Enums\Gender;
use App\Models\Enums\Languages;
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('students', function (Blueprint $table) {
            $table->id();
            $table->unsignedBigInteger('group_id');
            $table->unsignedBigInteger('educational_institute_id');
            $table->unsignedBigInteger('educational_program_id');
            $table->unsignedBigInteger('benefits_id')->nullable();
```

```
            $table->string('firstname')->index();
            $table->string('lastname')->index();
            $table->string('surname')->index();
            $table->string('phone')->unique();
            $table->string('email')->unique();
            $table->boolean('is_group_leader')->default(false);
            $table->boolean('is_academic_vacation')->default(false);
            $table->boolean('need_dormitory')->default(false);
            $table->date('entry_date');
            $table->integer('enrolled_in_the_course')->default(1);

$table->integer('financing_form')->default(FinancingForms::CONTRACT->value);
$table->integer('education_form')->default(EducationForms::DAILY->value);
$table->integer('enrollment_type')->default(EnrollmentType::TENDER->value);
$table->integer('educational_degrees')-
>default(EducationalDegrees::JUNIOR_PROFESSIONAL_BACHELOR->value);

            $table->date('birthday');
            $table->integer('gender');
            $table->string('citizenship');
            $table->string('place_of_residence');
            $table->date('graduation_date');
            $table->integer('language');
            $table->timestamps();

            $table->foreign('group_id')
                ->references('id')
                ->on('groups')
                ->onDelete('cascade')
                ->onUpdate('cascade');

            $table->foreign('educational_institute_id')
                ->references('id')
                ->on('educational_institutes')
                ->onDelete('restrict')
                ->onUpdate('cascade');

            $table->foreign('benefits_id')
                ->references('id')
                ->on('benefits')
                ->onDelete('set null')
                ->onUpdate('cascade');

            $table->foreign('educational_program_id')
                ->references('id')
                ->on('educational_programs')
                ->onDelete('restrict')
                ->onUpdate('cascade');
        });
    }
    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('students');
    }
};
```

З вищенаведеного коду бачимо, що міграція являє собою клас або файл, який повертає екземпляр класу. При цьому не дивлячись на те, як саме прописана міграція, вона повинна віднаслідуватись від базової Міграції задля того, щоб Eloquent міг з нею працювати.

В класі є два методи, метод up та down. Метод up виконується безпосередньо під час міграції, та використовується для опису схеми, для створення або оновлення таблиць. В методі up спочатку йде звернення до фасаду схема, який надає наступні методи: створення, видалення, оновлення та інші. З наведеного прикладу видно, що я використовую метод create для створення таблиці 'students', та передаю в нього анонімну функцію, яка приймає екземпляр класу Blueprint та описує, які поля та зв’язки мають бути в цій таблиці.

По схемі, яку я побудував раніше, будую поля для таблиці студенти, за допомогою об’єкту Blueprint будую всі необхідні поля, за допомогою $table->date додаю поля з типом дата, для зберігання дат в таблиці. Метод date приймає на вхід назву поля, яке буде додано до бази даних. Для створення текстових полів з типом CHAR, використовую $table->string, який приймає на вхід назву поля, та необов’язковий параметр довжини, від 1 до 255 символів. Для додавання булевих полей використовую $table->boolean, який приймає назву поля і так само для числових значень використовуючи метод integer.

Всі ці методи повертають об'єкти ColumnDefinition, тобто об’єкт, який передає інструкції для створення полів у таблиці. Відповідно вони мають свої методи, які можна викликати відразу на полі задля додаткових налаштувань, наприклад встановлення дефолтних значень, індексів, унікальності та інше.

Гарним прикладом використання цього є створення зовнішніх ключів для таблиці студента за допомогою методу foreign, який повертає об'єкт дефініції ключа – ForeignKeyDefinition. Метод приймає на вхід назву поля, яке є зовнішнім ключем. Далі, за допомогою методу references, вказую на поле у цільовій таблиці, на яке посилається зовнішній ключ. У цьому випадку зовнішній ключ group_id посилається на поле id у таблиці groups, та за допомогою onDelete('cascade') та onUpdate('cascade') вказую дії, які потрібно

виконувати при видалені та оновленні запису в батьківській таблиці, а саме на прикладі групи встановлюю каскадне видалення студентів та каскадне оновлення.

Метод down міграції містить в собі інструкції, які треба виконати для того, щоб відкотити зміни, які відбулись у базі даних назад. В наведеному вище коді, видно, що я використовую метод dropIfExists для того, щоб видалити таблицю студентів, якщо вона існує.

1.3.3 Створення моделей

На основі тої роботи, яку я виконав раніше, а саме, проєктування бази даних та створення міграцій, створюю моделі, які відповідають міграціям.

Модель Eloquent, хоч і є потужним інструментом для роботи з базою даних та представлення її в Моделях, працює в специфічний спосіб та порушує деякі принципи ООП, а саме інкапсуляцію та поліморфізм [10].

Моделі Eloquent порушують принцип інкапсуляції, надаючи доступ до всіх полів таблиці бази даних як до публічних властивостей моделі, при цьому значення береться через магічний метод гет. Це означає, що ви можете безпосередньо читати і змінювати будь-які атрибути моделі без використання спеціальних геттерів чи сеттерів. Такий підхід може призвести до непередбачуваних змін у внутрішньому стані об'єкта і ускладнити відстеження та контроль за цими змінами. Саме тому в коді передбачена взаємодія з моделлю саме через кастомні репозиторії та сервіси, які поліпшують ситуацію.

В Eloquent модель статично пов'язана з певною таблицею в базі даних, що обмежує її гнучкість, а саме позбавляє можливості легко вносити змінити поведінку моделі або її зв'язків із базою даних без внесення змін до базового класу.

Розглянемо основну структуру базової моделі “Студента” з описаними зв’язками з іншими моделями, приклад коду наведу нижче.

```
<?php

namespace App\Models;
use App\Models\Enums\DocumentType;
```

```
use App\Models\Enums\ParentStatuses;
use Illuminate\Database\Eloquent\Builder;
use Illuminate\Database\Eloquent\Collection;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\BelongsTo;
use Illuminate\Database\Eloquent\Relations\BelongsToMany;
use Illuminate\Database\Eloquent\Relations\HasMany;
use Illuminate\Database\Eloquent\Relations\HasOne;
use Illuminate\Database\Eloquent\Relations\MorphMany;
use Illuminate\Support\Carbon;

class Student extends Model
{
    use HasFactory;
    const IMAGE_TAG = 'student_photo';

    public function scopeSearch(Builder $query, mixed $value): Builder
    {
        if ($value) {
            $term = '%' . $value . '%';
            $query->where(function ($query) use ($term) {
$query->orWhere('phone', 'like', $term)
->orWhere('email', 'like', $term)
->orWhereRaw("CONCAT(firstname, ' ', lastname, ' ', surname)
 LIKE ?", [$term])->orWhere('place_of_residence', 'like', $term);
            });
        }
        return $query;
    }
    public function decrees(): BelongsToMany
    {
  return   $this->belongsToMany(Decree::class,   'student_decrees',   'student_id',
'decree_id')->withTimestamps();
    }
    public function group(): BelongsTo
    {
        return $this->belongsTo(Group::class);
    }
    public function parents(): BelongsToMany
    {
 return      $this->belongsToMany(StudentParent::class,      'student_of_parents',
'student_id', 'parent_id')->withPivot('status')->withTimestamps();
    }
    public function educationalInstitute(): BelongsTo
    {
 return $this->belongsTo(EducationalInstitute::class, 'educational_institute_id');
    }
    public function benefits(): BelongsTo
    {
        return $this->belongsTo(Benefits::class, 'benefits_id');
    }
    public function media(): MorphMany
    {
        return $this->morphMany(Media::class, 'model');
    }
    public function getPhotoByTag(): ?Model
    {
        return $this->media()->where('tag', self::IMAGE_TAG)->first();
    }
    public function getPhotoByName(): ?Model
```

```
        {
            $name = $this->getStudentPhotoName();
            return $this->media()->where('name', $name)->first();
        }
        public function studentCard(): HasOne
        {
            return $this->hasOne(StudentCard::class, 'student_id');
        }
        public function documents(): HasMany
        {
            return $this->hasMany(StudentDocument::class, 'student_id');
        }
        public function getStudentPhotoName(): string
        {
            return self::IMAGE_TAG . '_' . $this->id;
        }
        public function educationProgram(): BelongsTo
        {
        return $this->belongsTo(EducationalProgram::class, 'educational_program_id');
        }
    }
```

Через недоліки Eloquent, ми позбавлені гнучкості, тому замість опису властивостей безпосередньо в класі, я описую їх в док блоці моделі. Це робиться задля того, щоб IDE міг підказувати які саме модель має поля для полегшення розробки та не змінювати деякі методи базової моделі, що може призвести до помилок в роботі з моделями.

У представленій вище моделі “Студента” наведений метод для пошуку студентів, а саме метод який зберігає умови для пошуку за такими полями як електронна пошта, номер телефону, та ПІБ. Він викликається як статичний метод на об’єкті будувальника запитів без префіксу scope, після чого повертає об’єкт білдеру, що дозволяє додавати інші умови для більш складних запитів.

Нижче попереднього методу, знаходяться методи, які відповідають за встановлення зв’язків між моделями. Розглянемо деякі з них. Почнемо з методу group, код наведу нижче, який вказує на зв’язок з моделлю групи, а саме вказує на те, що студент належить одній групі, в той час, як група може мати безліч студентів. Тобто між вказуємо на те, що між моделями Група та студент присутній зв’язок один до багатьох.

```
    public function group(): BelongsTo
        {
            return $this->belongsTo(Group::class);
        }
```

Наступним специфічним прикладом встановлення зв’язку є зв’язок між студентом та його документами, код для методу наведу нижче. В ньому ми вказуємо що метод має повертати об’єкт HasMany, що вказує на те, що в цього студента є свій безліч документів, тут же я вказую на назву класу моделі, до якої має бути зв’язок та поле за яким цей зв’язок встановлено.

```
public function documents(): HasMany
    {
        return $this->hasMany(StudentDocument::class, 'student_id');
    }
```

Окрім зв’язків один до багатьох, в моделі ще прописані такі зв’язки як багато до багатьох, та один до одного. Для початку розглянемо зв’язок один до одного між студентом та студентським квитком, код методу наведу нижче. В методі ми вказуємо що він повертає об’єкт HasOne, що вказує на те, що в цього студента може бути лише один студентський квиток, сам метод вказує на те, що це зв’язок саме з моделлю студентського квитка, та поле за яким встановлено цей зв’язок.

```
public function studentCard(): HasOne
    {
        return $this->hasOne(StudentCard::class, 'student_id');
    }
```

Прикладом зв’язку багато до багатьох в моделі є зв’язок з моделлю наказів, зв’язок просаний в методі decrees, який вказує на те, що студент належить багатьом наказам, а в моделі наказу є метод, який вказує на те, що наказ належить багатьом студентам. Як і вище, в метод базової моделі передається назва класу моделі наказів, та інформація про поля та проміжкову таблицю, яка зв’язує обидві моделі.

```
public function decrees(): BelongsToMany
    {
            return $this->belongsToMany(Decree::class, 'student_decrees',
'student_id', 'decree_id')->withTimestamps();
    }
```

За прикладом моделі студент, можна визначити деякі позитивні сторони Eloquent, а саме простота та швидкість реалізації.

1.3.4 Створення контролерів та маршрутів

Створимо класи-контролери, що зв’яжуть класи моделей з представленнями – рис. 1.13.

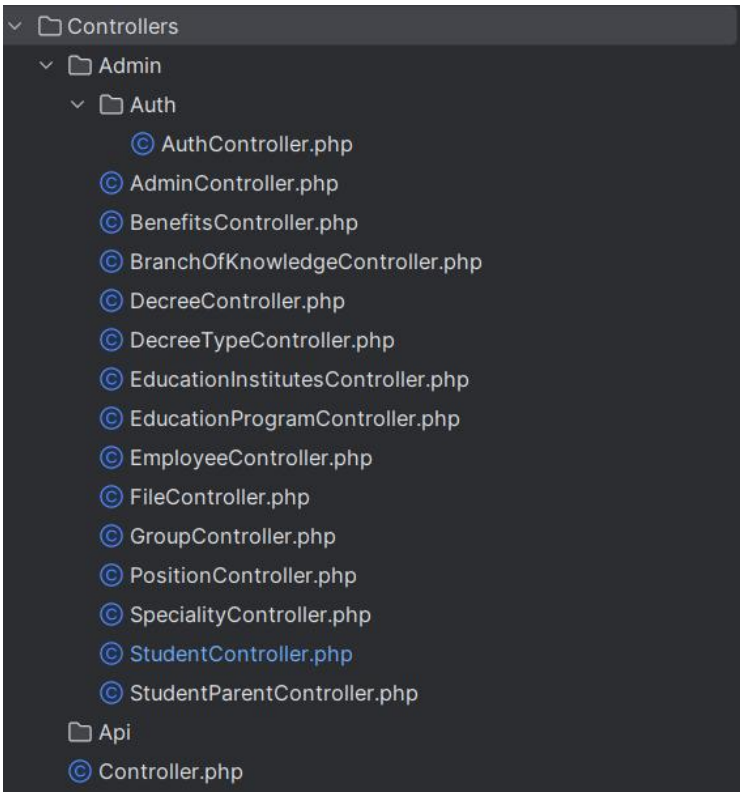


Рисунок 1.13. Створенні контролери

Розглянемо контроллер студента, код базової структури якого наведений нижче. Він буде мати методи, для втілення CRUD для студентів, а також метод для пошуку студентів через ajax запити описані на стороні фронту, кожний метод буде розглянуто окремо.

```
<?php

namespace App\Http\Controllers\Admin;

use App\Http\Controllers\Controller;
use App\Http\Requests\Student\StudentRequest;
use App\Models\Enums\Courses;
use App\Models\Enums\EducationalDegrees;
use App\Models\Enums\EducationForms;
use App\Models\Enums\EnrollmentType;
use App\Models\Enums\FinancingForms;
use App\Models\Enums\Gender;
use App\Models\Enums\Languages;
use App\Models\Student;
use App\Repositories\BenefitsRepository;
use App\Repositories\EducationalProgramRepository;
```

```
use App\Repositories\GroupRepository;
use App\Repositories\StudentRepository;
use App\Services\Files\FileService;
use App\Services\Model\StudentService;
use Illuminate\Contracts\View\Factory;
use Illuminate\Contracts\View\View;
use Illuminate\Foundation\Application;
use Illuminate\Http\JsonResponse;
use Illuminate\Http\RedirectResponse;
use Illuminate\Http\Request;
use Illuminate\Routing\Redirector;
use Throwable;

class StudentController extends Controller
{
    public function __construct(
        protected StudentService $service,
        protected StudentRepository $repository,
        protected GroupRepository $groupRepository,
        protected BenefitsRepository $benefitsRepository,
        protected EducationalProgramRepository $educationalProgramRepository,
        protected FileService $fileService
    )
    {
    }
            // інші методи, будуть розглянуті нижче
}
```

Вищенаведений клас контроллеру студентів, відображає базову структуру action-контроллера. А саме об'явлення класу контроллера, який наслідується від базового контроллера та магічний метод конструктора, в який я передаю всі залежності, які необхідні для роботи з ним. Контейнер Laravel сам розуміє, які екземпляри класів потрібно передати до конструктора, збирає їх та передає до контролеру, що є дуже ефективним для роботи з контролером.

Розглянемо метод store, код якого буде наведено нижче, який відповідає за створення студента.

```
public function store(StudentRequest $request): Redirector|RedirectResponse
    {
        try {
            $this->service->create(
                $request->getStudentValueObject(),
                $request->getPassportVO(),
                $request->getIdentificationVO(),
                $request->getEducationDocumentVO(),
                $request->getStudentCard(),
            );
$return    =    redirect('/admin/students')->with('success',    'Студента    успішно
створено');
        } catch (Throwable $e) {
$return   =   redirect('/admin/students')->with('error',   'Помилка   при   створенні
студента');
        }
```

```
        return $return;
    }
```

З наведеного коду видно, що метод очікує на вхід кастомний реквест StudentRequest, який наслідує FormRequest, для роботи з даними з форм. Завдяки контейнеру, всі методи контроллеру на вхід можуть приймати параметри, які з контексту збере контейнер та передасть у метод. В цьому випадку реквест студента використовується для валідації даних з форми, та створення іммутабельних об’єктів які будуть передаватись на сервісні слої, для створення студента, що гарантує незмінність даних.

Сам метод звертається до методу create сервісу, який переданий у конструкторі контролеру, та передає туди необхідні дані у вигляді імутабельних об’єктів, для зручної роботи з даними. Метод create в свою чергу звертається до відповідного методу в репозиторії, який в свою чергу вміє створювати та зберігати модель студента. В разі успішного створення студенту стається перенаправлення до списку студентів, з відповідним повідомленням. В разі виникнення виключень конструкція трай кетч перехоплює її, та повертає на сторінку зі списком студентів відповідне повідомлення, яке інформує про те, що студента не збережено через деякі проблеми.

Метод для оновлення даних студента подібний до методу створення, який був представлений вище. Єдина різниця між ними у тому, що замість методу create сервісу, метод для оновлення викликає метод update, та додатково передає в нього Модель студента, яку треба оновити.

Метод для видалення студента, код якого я наведу нижче, викликає у репозиторія метод deleteByModel, в який передає модель студента, після чого повертає повідомлення, з інформацією про статус видалення.

```
public function destroy(Student $student):

Application|Redirector|RedirectResponse|\Illuminate\Contracts\Foundation\Applicati
on
{
  try {
    $this->repository->deleteByModel($student);
   $return   =   redirect('/admin/students')->with('success',   'Студента   успішно
видалено');
  } catch (Throwable $e) {
```

```
        $return = redirect('/admin/students')->with('error', 'Помилка при видаленні студента');
        }
        return $return;
    }
```

Метод пошуку студента так само буде наведений нижче. Він приймає на вхід об’єкт базового реквеста, на відміну від методів створення та оновлення. Він викликає метод search сервісу, передаючи дані з реквесту, а саме значення, за яким потрібно шукати, та список ідентифікаторів, які треба виключити з пошуку.

```
    public function findPerson(Request $request): JsonResponse
    {
    $info = $this->service->search($request->input('search'), $request->input('selected_ids', []));
        return response()->json($info);
    }
```

Також розглянемо роути, код для яких наведено нижче, які пов’язани з даним контролером. Для всіх роутів, схему яких можна подивитись на рис. 1.14, системи додаю префікс адмін, для відображення, для кого є доступи до цих посилань, та доданий мідлвар 'auth.admin', який вказує системі перевіряти, чи авторизований користувач у системі перед наданням доступу.

```
    Route::prefix('admin')->name('admin.')->middleware('auth.admin')->group(function ()
{
    Route::resource('/students', StudentController::class)->names([
            'index' => 'students.list',
    ]);
    Route::get('/search-students', [StudentController::class, 'findPerson'])
        ->name('search-students');
    });
```

Контролери Laravel є дуже зручними, так як контейнер Laravel вміє добре збирати сервіси та саме контролери. Це дозволяє будувати різноманітні контролери для керування системою.

Окремий контроллер використовується для генерації файлів і має окремі методи, які відповідають за повернення, відповідних файлів, а саме особової картка, індивідуального плану, таблиці прийому групи. Код контроллеру наведу нижче:

```
    <?php
    namespace App\Http\Controllers\Admin;
```

```
use App\Http\Controllers\Controller;
use App\Models\Group;
use App\Models\Student;
use App\Repositories\GroupRepository;
use App\Services\Files\FileGenerator;
use App\Services\Files\HtmlFiles\StudentInfoService;
use App\Services\Files\XlsFiles\Generators\ReceptionGroupTableGenerator;
use Carbon\Carbon;
use Illuminate\Contracts\View\Factory;
use Illuminate\Contracts\View\View;
use Illuminate\Foundation\Application;
use Illuminate\Http\RedirectResponse;
use Illuminate\Http\Response;
use Illuminate\Routing\ResponseFactory;
use Symfony\Component\HttpFoundation\BinaryFileResponse;

class FileController extends Controller
{
    public function __construct(
        protected StudentInfoService           $studentInfoService,
        protected FileGenerator $fileGenerator,
        protected ReceptionGroupTableGenerator $xlsTablesService
    )
    {
    }
public              function             studentCard(Student             $student):
View|Application|Factory|\Illuminate\Contracts\Foundation\Application
    {
        $vo = $this->studentInfoService->getStudentIndividualPlanVO($student);
        return view('files.individual-plan', compact('vo'));
    }

public           function           studentPersonalCard(Student           $student):
View|Application|Factory|\Illuminate\Contracts\Foundation\Application
    {
        $vo = $this->studentInfoService->getStudentCardVO($student);
        return view('files.personal-card', compact('vo'));
    }
public            function            groupReceptionSample(Group            $group):
BinaryFileResponse|RedirectResponse
    {
        $vo = $this->fileGenerator->getGroupReceptionXlsx($group);
return          response()->download($vo->getPath(),          $vo->getFullName())-
>deleteFileAfterSend();
    }
}
```

Розглянемо метод groupReceptionSample, який генерує тимчасову ексель таблицю прийому групи через сервіс, який вказаний в залежностях контроллеру. При зверненні до файл генератору, в нього викликається метод getGroupReceptionXlsx, який приймає обов'язковим параметром модель групи, для якої буде побудована таблиця. Цей метод збирає всю необхідну для побудови таблиці інформації та передає її ще на рівень нижче а саме до класу ReceptionGroupTableGenerator та викликає в нього метод

generateReceptionSampleTables, який вміє генерувати потрібну таблицю за даними які в нього передані, код наведу нижче.

```
public function generateReceptionSampleTables(
        Group    $group,
        Employee $curator,
        array    $students,
        string   $filename,
        string   $format
): string
{
 $sheet = $this->spreadsheet->getActiveSheet();
 $this->spreadsheet->getDefaultStyle()->getFont()->setName('Times New Roman');
 $sheet->setCellValue('A1', 'Групи ' . $group->getCourseAndCode());
 $sheet->setCellValue('A2', Helper::currentAcademicYear() . ' навчальний рік');
 $sheet->setCellValue('A3', 'Кл.керівник - ' . $curator->getLastnameAndInitials());
 $sheet->mergeCells('A1:P1');
 $sheet->mergeCells('A2:P2');
 $sheet->mergeCells('A3:P3');
 $sheet->getStyle('A1:P3')->getAlignment()-
>setHorizontal(Alignment::HORIZONTAL_CENTER);
 $sheet->getStyle('A1:A3')->getFont()->setBold(true)->setSize(18);
 $sheet->getStyle('A2')->getFont()->setBold(true)->setSize(12);
 $sheet->getStyle('A3')->getFont()->setBold(true)->setSize(14);
 $this->getTableHeader($sheet);
 $this->addStudentDataRows($sheet, $students);
 $this->setColumnWidths($sheet);
 $this->setFixedRowHeight($sheet, 4);
 $writer = new Xlsx($this->spreadsheet);
 $tempFilePath = tempnam(sys_get_temp_dir(), $filename) . $format;
 $writer->save($tempFilePath);
 return $tempFilePath;
}
```

Він створює шапку для файлу та викликає методи для генерації шапки таблиці та заповнення рядків, передаючи туди масив з інформацією про студентів цієї групи. В результаті, виконання цих методів до контроллеру повертається шлях до файлу, який тимчасово збережений у директорії temp, та відправляється на завантаження до адміністратора.

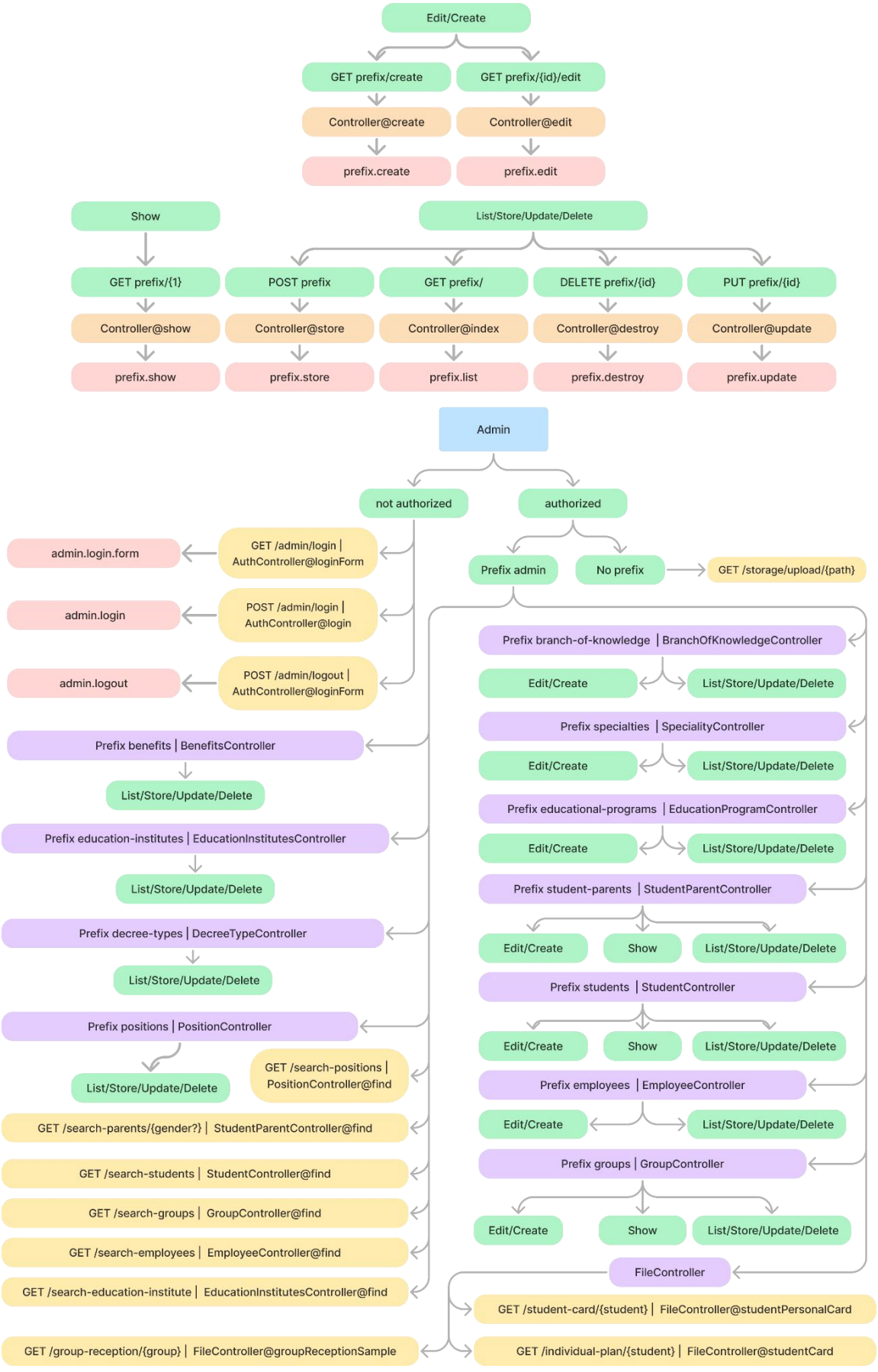


Рисунок 1.14. Схема роутів застосунку

1.3.5 Створення сервісу для генерації таблиць для груп

Окремою частиною роботи є саме генерація документів. Розглянемо сервіс генерації таблиць для груп, який отримує екземпляр класу, що вміє генерувати необхідну таблицю та запускає генерацію таблиці. Код сервісу та методу контролеру, де він викликається наведено нижче.

```
<?php
namespace App\Services\Files;
use App\Models\Group;
use App\Models\ValueObject\TemporaryFileValueObject;
  use App\Services\Files\XlsFiles\Interfaces\IGroupTableGenerator;
use Carbon\Carbon;
class GroupTableGenerator implements IGroupTableGenerator  {
       public function __construct(protected IGroupTableGenerator $tableGenerator){}
       public function generate(Group $group, string $filename, string $format =
       self::FORMAT_XLSX): string
       {
       return $this->tableGenerator->generate($group, $filename, $format);
       }
       public function setTableGenerator(IGroupTableGenerator $tableGenerator): void
       {
       $this->tableGenerator = $tableGenerator;
       }
       protected function getTemporaryFileVO(
              string $filename, string $format, string $filePath
       ): TemporaryFileValueObject
       {
       return new TemporaryFileValueObject($filename, $format , $filePath);
       }
}
public function groupReceptionSample(Group $group):
BinaryFileResponse|RedirectResponse
{
       try {
              $groupTableGenerator = $this->container-
              >get(GroupTableGenerator::class);
              $filename = 'reception_' . $group->getCourseAndCode() . ' ' .
              Carbon::now()->format('d-m-Y H-i-s');
              $format = FileTypes::TABLE_XLSX->value;
              $filepath = $groupTableGenerator->generate($group, $filename, $format);
              $return = response()->download($filepath, $filename . $format)-
              >deleteFileAfterSend();
       } catch (Throwable $e) {
              logger($e->getMessage());
              $return = redirect()->back()->with('error', 'Сталася помилка, спробуйте
              пізніше');
       }
       return $return;
}
```

По коду вище видно, що представлений сервіс використовує поведінковий шаблон проектування "Стратегія", щоб легко управляти генерацією та генерувати саме потрібний файл в потрібному місці. В реалізації всі стратегії

для генерації таблиць мають імплементувати інтерфейс IGroupTableGenerator. Це дозволяю створювати нові стратегії та гарантує, що в кожній з них буде відповідний метод, який буде генерувати таблицю та повертати шлях до неї.

Для реалізації був використаний контейнер фреймворка задля того, щоб мати можливість отримувати сервіс та стратегію в будь-якому місці. Використання контейнеру дозволить мені, легко міняти стратегію та генерувати необхідну в цьому методі або при цій умові таблицю.

1.4 Мануальне тестування

1.4.1 Тестування авторизації

Для тестування роутів використовую Postman. Так як веб-застосунок монолітний, тип даних які відправляються на Back-End – це form-data, яка симулює відправку даних з форми. Спочатку тестую пост запит на URL http://localhost/admin/login, так як веб-застосунок піднятий локально, результати виконання запиту зображені на рис. 1.15.

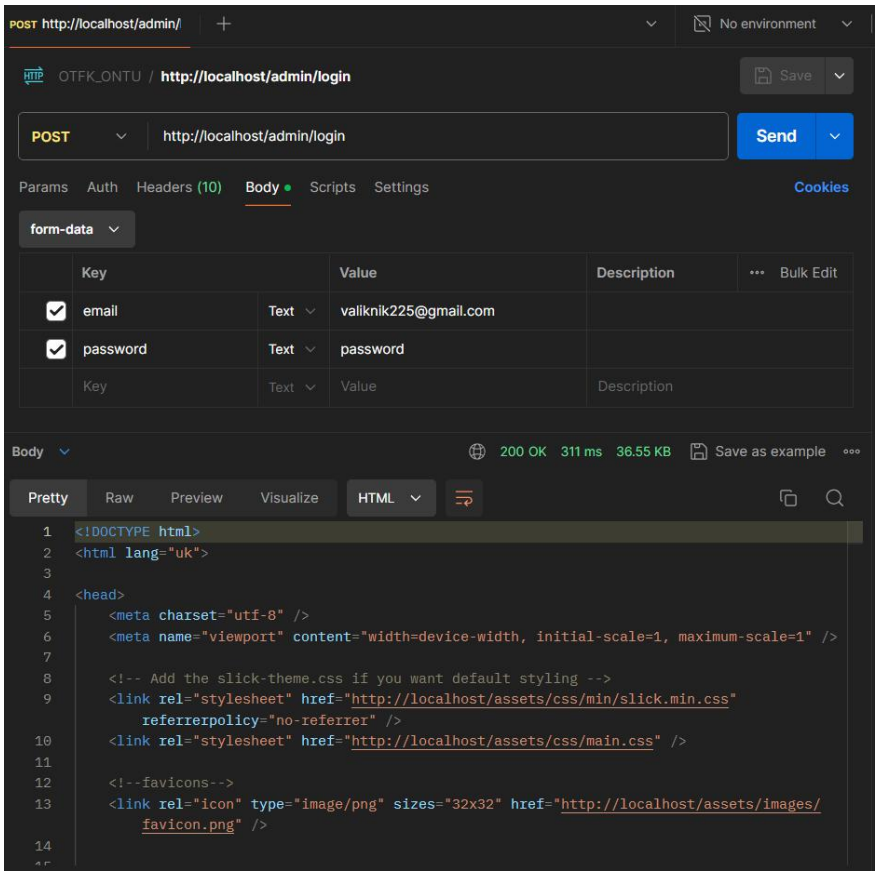


Рисунок 1.15. Результат підняття контейнерів

Так як веб-застосунок – монолітний, то в результаті я отримую html сторінки зі списком студентів та статус 200, що позначає, що застосунок відпрацював без помилок, так як метод контролеру, робить редирект на сторінку студентів, після успішної авторизації. В разі введення некоректних даних, користувача повертає на сторінку авторизації, та додається кастомне повідомлення до сесії про помилку з ключем error і значенням “Помилка авторизації”.

1.4.2 Тестування типових роутів для ресурсів

Протестуємо CRUD системи для студентів, вона вимагає авторизації від користувача перед запитом, тому для початку, подивимось на результат звернення без авторизації, який зображено на рис. 1.16.

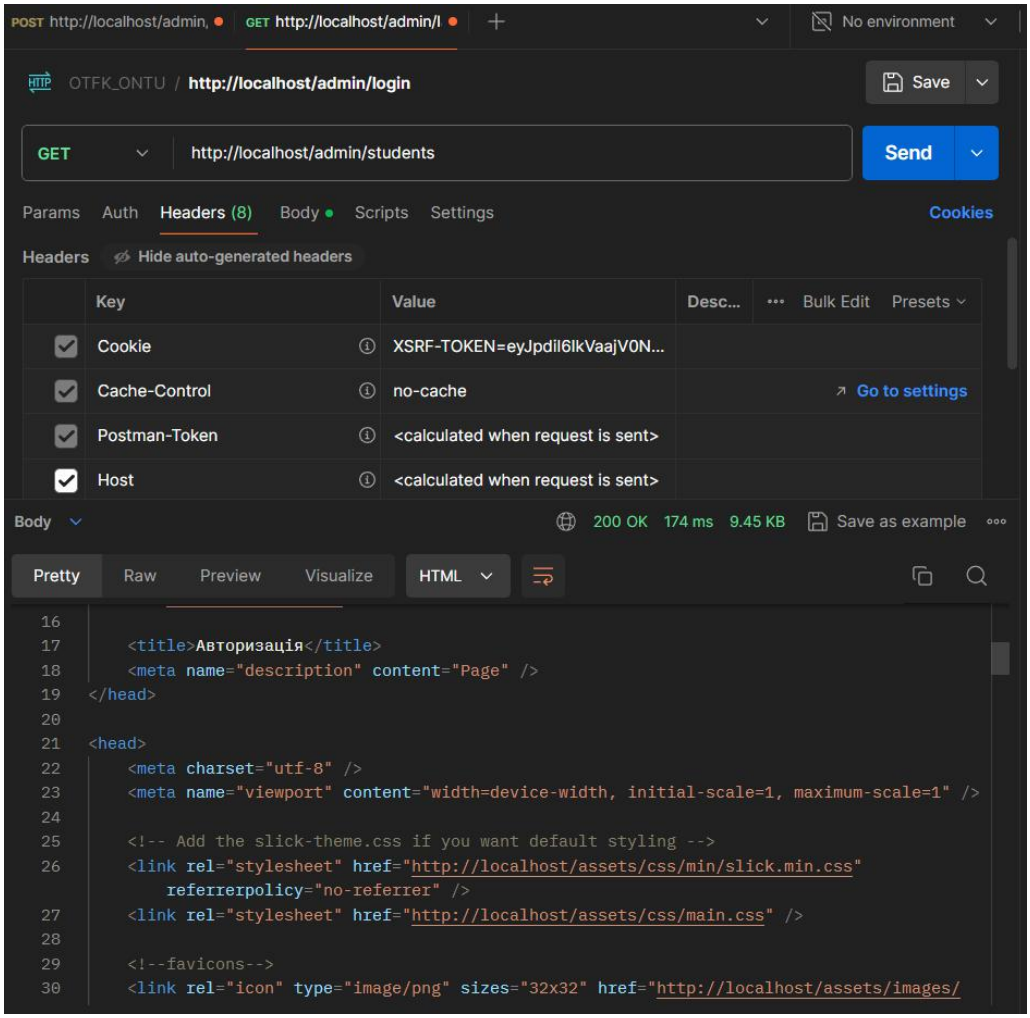


Рисунок 1.16. Результат надсилання запиту на отримання сторінки списку студентів при не авторизованому користувачеві

В результаті запиту отримали редирект на сторінку авторизації, мідлвар перевірки авторизації працює успішно.

Відправимо запит через авторизованого користувача на роут для перегляду певного студента, задля того, щоб перевірити, що авторизований користувач може отримати доступ до сторінки. Результат зображено на рис. 1.17.

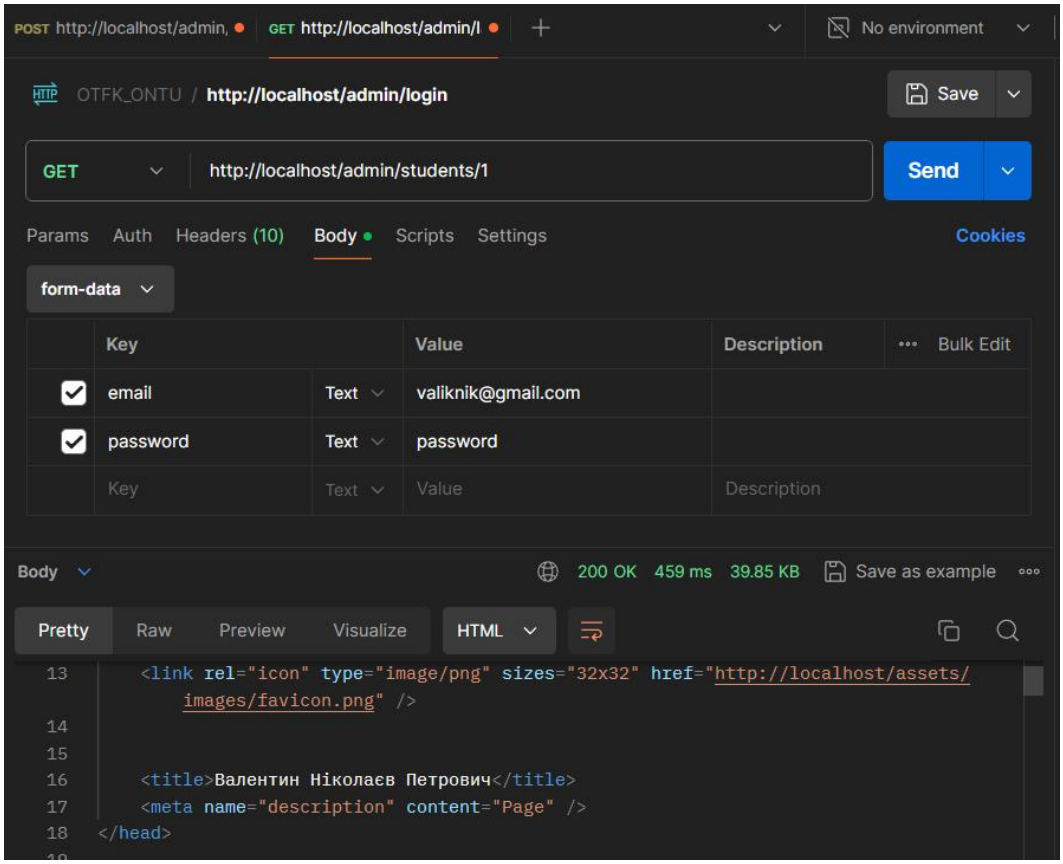


Рисунок 1.17. Результат надсилання запиту на отримання сторінки перегляду певного студента при авторизованому користувачеві

З рисунку видно, що в результаті ми отримуємо сторінку перегляду конкретного студента, яка має заголовок, який відповідає ПІБ студента.

Наступним роутом для тестування візьмемо роут для створення студента. Для здійснення запиту заповню form-data та встановлюю метод POST, для відправки даних на сервер. В результаті отримаю редирект на список студентів із повідомленням, про успішне створення студента – рис. 1.18.

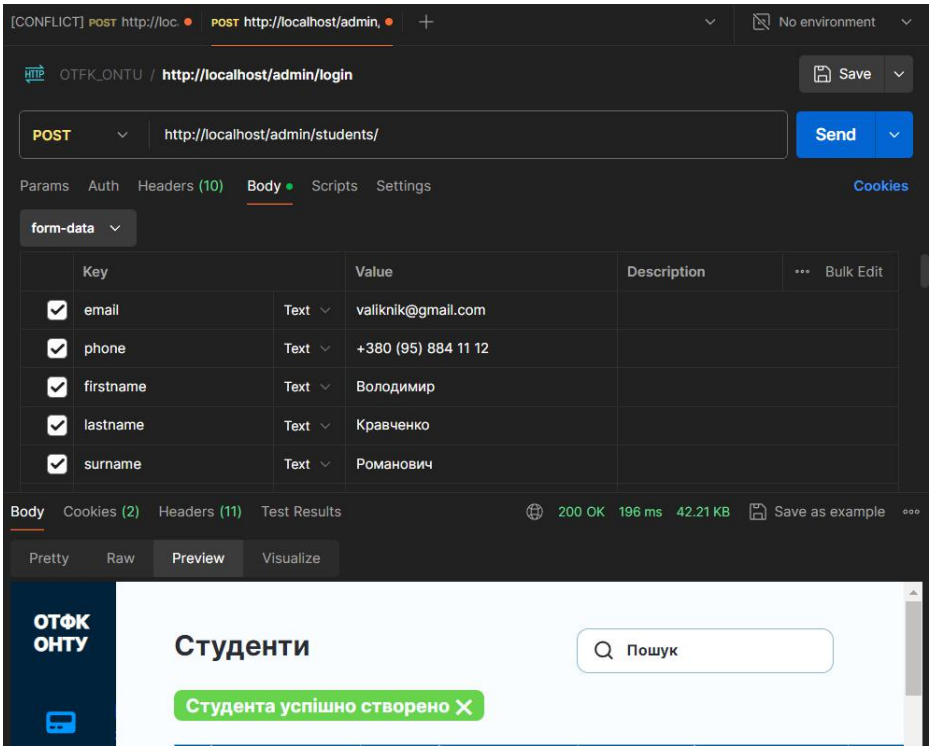


Рисунок 1.18. Результат надсилання запиту на створення студенту

Для здійснення запиту на оновлення студента міняю form-date та встановлюю метод PUT, для відправки даних на сервер. Отримаю редирект на список студентів із повідомленням, про оновлення студента - рис. 1.19.

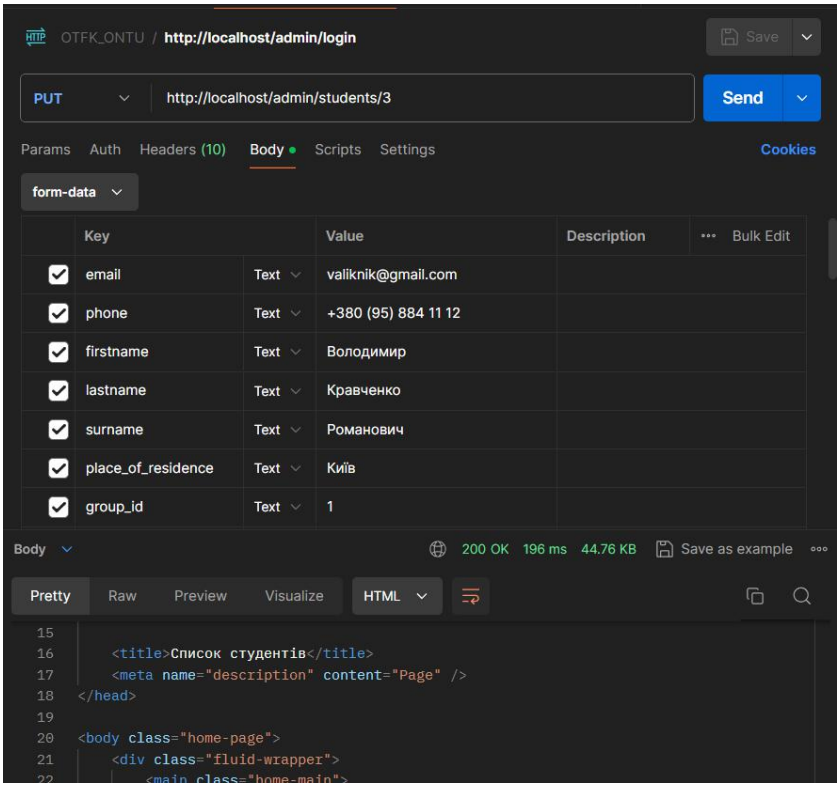


Рисунок 1.19. Результат надсилання запиту на оновлення студенту

Залишилось протестувати метод видалення студента, результат виконання якого буде зображено на рис. 1.20. Де буде видно повідомлення про успішне видалення, для відправки запити використовую метод DELETE, який вказує на те що ресурс потрібно видалити.

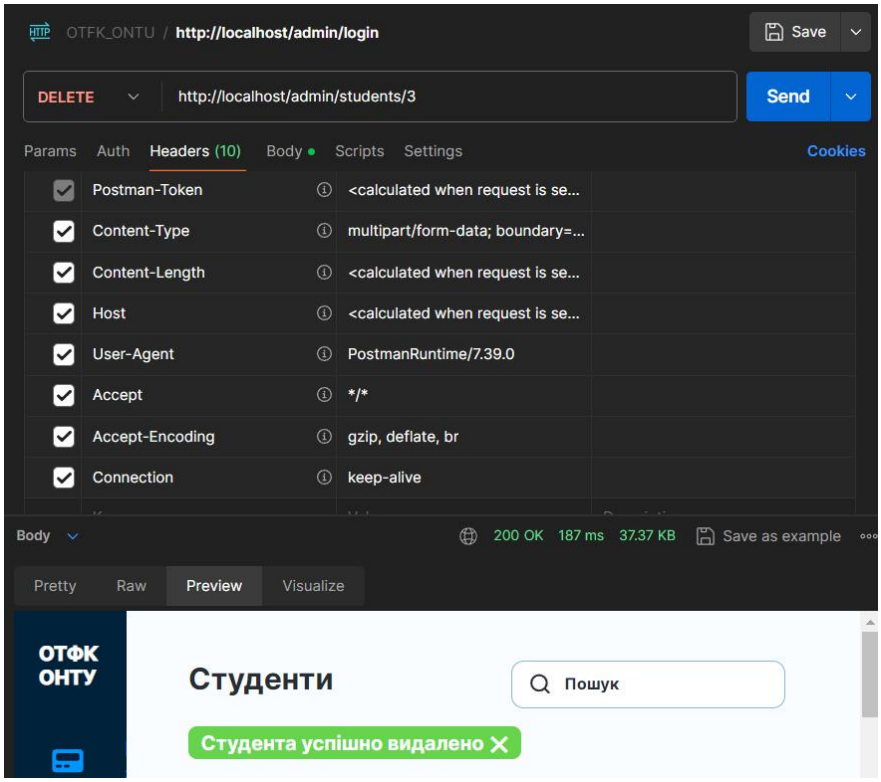


Рисунок 1.20. Результат надсилання запиту на видалення студенту

1.4.3 Тестування роутів для генерації файлів

Протестуємо генерацію таблиці прийому групи, для цього надсилаю GET запит на http://localhost/admin/group-reception/1, де 1 – це ідентифікатор групи, для якої потрібно згенерувати цей файл. Результат на рис. 1.21.

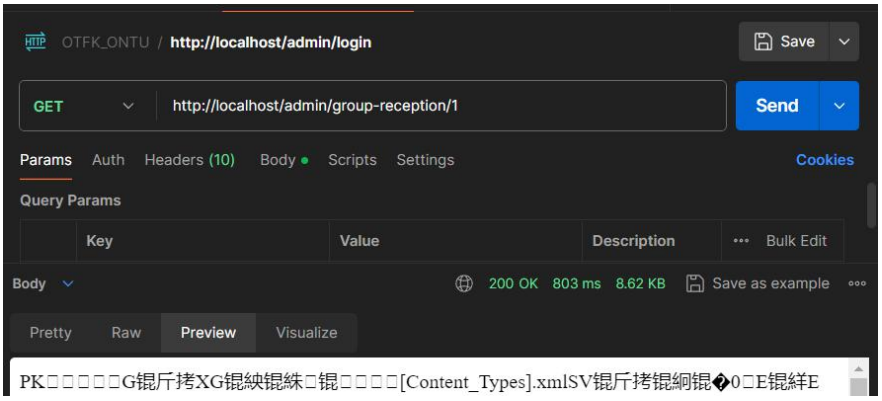


Рисунок 1.21 Результат генерації файлу запиту на видалення студенту

2 ЕКОНОМІЧНИЙ РОЗДІЛ

В дипломному проєкті створювалася Back-End частина автоматизованої веб-системи обліку руху контингенту здобувачів відділення комп'ютерних систем. Цей сайт розроблений на замовлення і не буде приносити дохід, так як не є комерційним. Веб-система буде використовуватися завідувачем відділення та довіреними особами для ефективного та зручного управління обліком здобувачів. При оцінці ефективності створюваної веб-системи слід враховувати, що в залежності від досягнутого результату можна визначити такі види ефективності сайту: економічна, функціональна та соціальна ефективність. Оскільки вона не є комерційним і не приноситиме доходу, розраховуємо лише витрати на її розробку та функціональну і соціальну ефективності веб-системи.

Розрахунок економічної ефективності розробки створеного web-сайту.

Загальні витрати ($В_з$) на створення сайту складаються з витрати на розробку сайту, на впровадження сайту та на експлуатацію сайту, і розраховується за формулою:

$$В_з = В_р + В_в + В_е,$$

Витрати на розробку сайту ($В_р$) розраховуємо з урахуванням оплати праці виконавця - Back-End розробника та єдиного соціального внеску.

Для визначення трудомісткості розробки сайту ($В_р$) складено план-графік по розробці Back-End частини веб-системи і тривалості виконання робіт. Розподіл робіт по етапах і видах виконавців наведено в таблиці 2.1.

Таблиця 2.1. План-графік по розробці веб-системи

№	Назва етапу	Час виконання (годин)	Посада виконавця
1	Ініціалізація проєкту веб-системи	5 години	Back-End розробник
2	Створення структури та зв’язків баз даних	14 годин	Back-End розробник

Продовження таблиці 2.1

№	Назва етапу	Час виконання (годин)	Посада виконавця
3	Написання CRUD системи для веб-системи	90 годин	Back-End розробник
4	Генерація документів	20 годин	Back-End розробник
	ВСЬОГО:	129 годин	

Витрати на заробітну плату приведені в таблиці 2.2.

Таблиця 2.2. Витрати на заробітну плату

№	Персонал	Етапи розробки	Кількість робочих годин (або днів)	Погодинна ставка (або денна ставка), грн.	Заробітна плата, грн.
1	Back-End розробник	Ініціалізація проєкту веб-системи	5 годин	140грн./год.	700 грн.
2	Back-End розробник	Створення структури та зв’язків баз даних	14 годин	140грн./год.	1 960 грн.
3	Back-End розробник	Написання CRUD системи для веб-системи	90 годин	140грн./год.	12 600 грн.
4	Back-End розробник	Генерація документів	20 годин	140грн./год.	2 800 грн.
ВСЬОГО:					$В_{зп}$ = 18 060 грн.

Треба зазначити, що розробка Back-End частини автоматизованої веб-системи обліку руху контингенту здобувачів відділення комп'ютерних систем є лише частиною розробки для цілої веб-системи і є комплексною роботою. Тому для розрахунку повної суми витрат на заробітну плату я додам суму заробітної плати за роботу Frontend-End розробника, яка включає в себе аналіз предметної

області (8 годин), аналіз конкурентів та цільової аудиторії (10 годин), створення дизайну (22 години) та розробку Front-End частини (40 годин). Погодинна оплата праці Front-End розробника становить 120грн./год., з чого можемо розрахувати вартість роботи розробки Front-End частини – 13 200 грн.

Тоді загальна сума на заробітну плату розробникам буде становити:

$$В_{зп} = 18\ 060 + 13\ 200 = 31\ 260 \text{ грн.}$$

Розмір єдиного соціального внеску складає 22% від заробітної плати, розраховується за наступною формулою:

$$В_{єсв} = В_{зп} \text{ х } 0{,}22 = 31\ 260 \text{ х } 0{,}22 = 6\ 877 \text{ грн.}$$

Загальні витрати ($В_р$) на розробку веб-сайту розраховуються як сума витрат на заробітну плату праці персоналу ($В_{зп}$) та єдиного соціального внеску ($В_{єсв}$):

$$В_р = В_{зп} + В_{єсв} = 31\ 260 + 6\ 877 = 38\ 137 \text{ грн.}$$

Витрати на впровадження сайту ($В_в$) складаються з двох складових :

1. Вартість на реєстрацію доменного імені .com зазвичай становить 756 грн. ($В_{в1}$) дані взяті з сайту imena.ua;
2. Витрати на реєстрацію в пошукових системах ($В_{в2}$), наприклад Google через Google Search Console є безкоштовною

$$В_в = В_{в1} + В_{в2} = 756 + 0 = 756 \text{ грн.}$$

Витрати на експлуатацію сайту ($В_е$) включають вартість робіт з підтримки сайту в робочому стані і вартість послуг по продовженню доменного імені на 1 рік. Я розраховую вартість хостингу на 1 рік, в суму якого також входить вартість налаштування параметрів серверу хостингу, забезпечення щомісячного захисту веб-системи та створення резервних копій системи. Додатково враховуються витрати на технічне обслуговування, що включають оновлення програмного забезпечення та моніторинг безпеки. У таблиці 2.3 визначаються постійні витрати як сума витрат на впровадження та експлуатацію сайту протягом одного року, що дозволяє отримати повну картину необхідних ресурсів для стабільного функціонування веб-платформи.

Таблиця 2.3. Постійні витрати

№	Стаття витрат	Вартість за рік, грн.
1	Хостинг на 1 рік	8 707 грн.
Всього:		$В_е$ = 8 707 грн.

Нижче наведено рис. 2.1 з даними сайту сайту хмарної платформи DigitalOcean, де показано суму витрат на хостинг на 1 місяць (18 доларів). З чого робимо висновок, що сума вартості хостингу на 1 рік становитиме 8 707 грн.

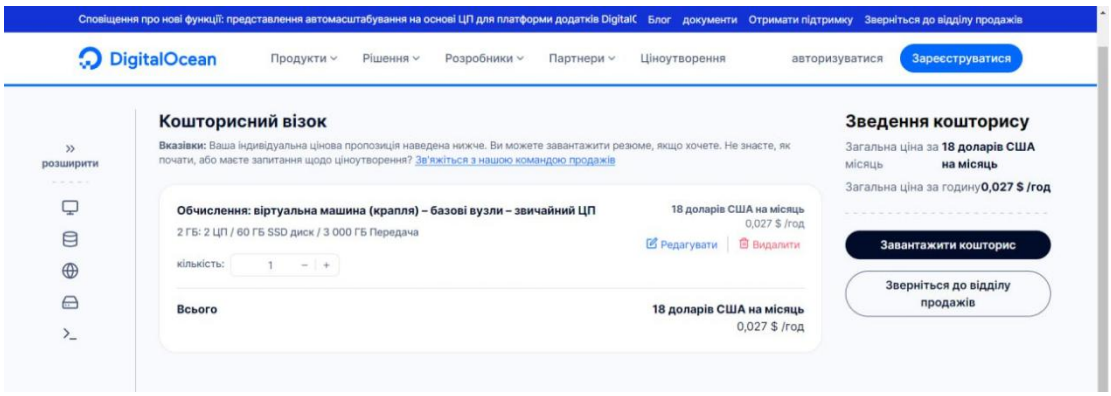


Рисунок 2.1. Вартість хостингу з даних сайту хмарної платформи DigitalOcean

Загальні витрати ($В_з$) на розробку, впровадження та експлуатацію веб-сайту розраховуються за наступною формулою:

$$В_з = В_р + (В_в + Ве) = 31\ 260 + (756 + 8\ 707) = 40\ 723 \text{ грн.}$$

Функціональна ефективність

Функціональна ефективність автоматизованої веб-системи обліку руху контингенту здобувачів відділення комп'ютерних систем забезпечує повноту, точність і доступність інформації. Система дозволяє зберігати, оновлювати та отримувати необхідну інформацію про студентів, групи, батьків, спеціальності, освітні програми, накази, типи наказів, співробітників, посади, навчальні заклади та пільги в будь-який час. Це гарантує, що завідувач відділення та довірені особи завжди будуть мати доступ до актуальних даних.

Система значно спрощує технологічні процеси. Завідувач відділення може легко додати нового студента, вводячи його дані в базу даних. Інші дані, такі як група, галузь знань або навчальний заклад, автоматично підтягнуться з попередньо заповнених таблиць. Це значно зменшить час і зусилля, необхідні

для введення даних вручну. Всі дані зберігаються в централізованій базі, що дозволяє уникнути дублювання інформації та помилок, пов'язаних з ручним введенням. Автоматизоване управління та обробка даних також дозволяють швидко генерувати звіти та документи.

Автоматизована система значно підвищує продуктивність праці, оскільки багато рутинних операцій виконуються автоматично. Веб-система дозволяє легко відстежувати зміни та оновлення, що сприяє більш ефективному управлінню контингентом здобувачів. Таким чином, функціональна ефективність автоматизованої веб-системи обліку руху контингенту здобувачів полягає в забезпеченні високого рівня організації даних, їх точності та доступності, а також у значному спрощенні та прискоренні процесів управління та обробки інформації.

Соціальна ефективність

Веб-система обліку студентів допомагає формувати позитивний імідж відділення комп'ютерних систем як сучасного та інноваційного освітнього закладу. Автоматизація процесів, пов'язаних з обліком та управлінням контингентом здобувачів, демонструє відкритість до новітніх технологій та підвищує ефективність роботи, що є привабливим фактором для потенційних студентів та їхніх батьків. Це також допомагає зміцнити довіру до адміністрації та її здатності забезпечити високий рівень організації навчального процесу.

Система дозволяє завідувачу відділення та довіреним особам ефективно управляти даними про студентів, групи, спеціальності та інші аспекти освітнього процесу, що покращує організацію та управління. Це сприяє поліпшенню комунікації між студентами, їх батьками та адміністрацією, що підвищує загальний рівень обізнаності та задоволеності всіх учасників освітнього процесу.

Важливо зазначити, що доступ до даних у системі мають лише завідувач відділення та довірені особи, що забезпечує конфіденційність та безпеку інформації. Це підвищує рівень довіри з боку студентів та їхніх батьків. Тому у результаті впровадження веб-системи, яка розроблялася у дипломному проєкті,

маємо покращення комунікації, підвищення рівня задоволеності учасників освітнього процесу та зміцнення позитивного іміджу освітнього заклад.

3 РОЗДІЛ ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

В даному розділі розглядаються питання охорони праці, пов'язані з роботою за персональним комп'ютером. Відповідно до проєкту, при розробці Back-End частини, яка була розроблена під час виконання дипломного проєкту, більшість часу була проведена за комп’ютером на персональному робочому місці вдома. Для забезпечення зручних та безпечних умов, потрібно ретельно дотримуватися всіх норм та правил охорони праці та пожежної безпеки, розмістивши робоче місце згідно з вимогами ергономіки та забезпечивши необхідні умови мікроклімату.

3.1 Негативні фактори під час роботи за комп’ютером

Під час роботи за комп'ютером, на людину можуть діяти фізичні та психофізіологічні небезпечні та шкідливі фактори. При дослідженні небезпечних факторів, було виявлено, що 60% користувачів стикаються з захворюваннями органів зору, 20% – з хворобами серцево-судинної системи, 10% – із захворюваннями шлунково-кишкового тракту, а 5% – із шкірними захворюваннями. У наслідках розповсюджених проблем також є призведення до неправильностей у вигині тіла, що може викликати біль у спині та шиї через тривалий час у сидячому положенні.

Постійна робота за монітором може спричинити напругу та втомленість очей, що може призвести до синдрому комп'ютерного зору та інших проблем із зором, таких як погіршення гостроти зору. Психологічні навантаження впливають на організм людини не менше інших. Вони можуть бути викликані високим тиском, дедлайнами та необхідністю постійного концентрування. Це все може призвести до стресу та погіршення психічного здоров'я.

3.2 Гігієнічні норми

Охорона праці є важливою складовою, яка забезпечує безпеку та здоров'я працівників на будь-якому робочому місці. Працюючи на особистому робочому

місці за комп’ютером вдома, дуже важливо дотримуватись правил охорони праці, оскільки працівник сам повинен подбати про власне здоров’я та безпеку.

В проєкті розглянуті та застосовані гігієнічні норми охорони праці, вимоги до приміщення, робочого місця, використання ергономічного обладнання, освітлення, мікроклімату, шуму, електробезпеку та пожежну безпеку.

3.2.1 Вимоги до приміщення

Згідно з нормативними вимогами, кімната для роботи з комп'ютером повинна мати достатню площу для забезпечення комфортного розміщення необхідних меблів та обладнання. За рекомендаціями охорони праці, площа робочого місця для роботи з комп'ютером повинна складати не менше 6 кв. метрів.

3.2.2 Вимоги до робочого місця

Робочий стіл повинен стояти в добре освітленому місці, так, щоб на екрані не було відблисків. Його параметри мають відповідати сучасним вимогам ергономіки і забезпечувати оптимальне розміщення на робочій поверхні використовуваного обладнання і документів, що показано на рис. 3.1.

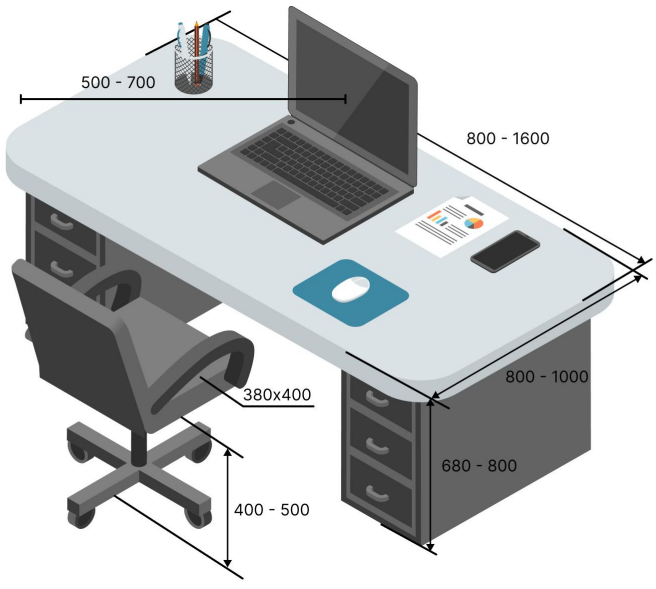


Рисунок 3.1. Вимоги ергономіки робочого столу у міліметрах

Висота столу має бути у межах 680-800 мм, його глибина від 800 до 1000 мм, а глибина – від 800 до 1000 мм. Інші розміри не рекомендуються до

використання, так як будуть незручними для виконання певних робочих завдань. Робочий стіл повинен мати простір для ніг заввишки не менше ніж 600 мм, завширшки не менше ніж 500 мм, завглибшки (на рівні колін) не менше ніж 450 мм, на рівні простягнутої ноги – ніж 650 мм.

Робочий стілець має бути регульованим за висотою, підйомно-поворотним, з кутом і нахилу сидіння та спинки, передній край - заокругленим. Регулювання за кожним із параметрів має здійснюватися незалежно, легко і надійно фіксуватися. Висота поверхні сидіння має регулюватися в межах від 400 до 500 мм, а ширина і глибина становити не менше ніж 400 мм. Кут нахилу сидіння – до 15° вперед і до 5° назад.

Висота спинки стільця має становити (300±20) мм, ширина – не менше ніж 380 мм, радіус кривизни горизонтальної площини – 400 мм. Відстань від спинки до переднього краю сидіння має регулюватися в межах від 260 до 400 мм.

Для зниження напруження м'язів кистей рук слід використовувати стаціонарні або змінні підлокітники завдовжки не менше ніж 250 мм, завширшки від 50 до 70 мм, які можуть регулюватися за висотою над сидінням у межах від 230 до 260 мм і відстанню між підлокітниками в межах від 350 до 500 мм.

Поверхня сидіння і спинки стільця має бути напівм'якою з нековзним, повітронепроникним покриттям, яка легко чиститься і не електризується.

Підставка для ніг є необов’язковим елементом, якщо при положенні сидячи ноги рівно стоять на підлозі, в інших випадках - необхідна. Робоче місце має бути обладнане підставкою для ніг завширшки не менше ніж 300 мм, завглибшки не менше ніж 400 мм, що регулюється за висотою в межах до 150 мм і за кутом нахилу опорної поверхні підставки до 20°. Підставка повинна мати рифлену поверхню і бортик по передньому краю заввишки 10 мм.

Екран комп'ютеру повинен розташовуватися на оптимальній відстані від очей на відстані від 500 до 700 мм. Клавіатуру слід розташовувати на поверхні столу на відстані 100-300мм від зовнішнього краю до працюючого.

3.2.3 Вимоги до світлення

Природне освітлення має здійснюватися через світлові прорізи, орієнтовані переважно на північ чи північний схід, і забезпечувати коефіцієнт природної освітленості (КПО) не нижче, ніж 1,5%.

Штучне освітлення в приміщені із робочим місцем, обладнаним комп'ютером, має здійснюватися системою загального рівномірного освітлення. Значення освітленості на поверхні робочого столу в зоні розміщення документів має становити 300-500 лк, якщо це неможливо забезпечити системою загального освітлення, допускається використовувати місцеве освітлення. Як джерела світла для штучного освітлення мають застосовуватись переважно люмінесцентні лампи типу ЛБ.

3.2.4 Вимоги до шуму і вібрацій

Працюючи вдома за комп'ютером, важливо звести рівень шуму до мінімуму, щоб уникнути відволікання та стресу для організму. Відповідно до вимог, було досліджено, що допустимий рівень шуму на робочих місцях, де використовується комп'ютер, не повинен перевищувати 50-65 дБ. Це дозволяє підтримувати комфортні умови для зосередження на роботі та знижує ризик втоми та головного болю. Для досягнення цих цілей можна використовувати звукоізоляційні матеріали, можуть бути встановлені склопакети або можна використовувати навушники з шумозаглушенням.

3.2.5 Вимоги до мікроклімату

Дотримання вимог до мікроклімату на робочому місці є важливою складовою при роботі за комп'ютером, так як вони забезпечують комфортні умови, запобігають втомі та підвищують продуктивність праці. Рекомендується провітрювати кімнату, у якій проходить робота за комп'ютером, кожні 1-2 години, щоб забезпечити притік свіжого повітря, знизити рівень вуглекислого газу, уникнути перегріву техніки та підтримувати комфортний мікроклімат.

На робочих місцях мають забезпечуватись нормовані значення параметрів мікроклімату, такі як температура повітря – 22–25°C, відносна вологість повітря

— 40–60%, швидкість руху повітря — не більше 0,1 м/с. Для підтримки допустимих значень мікроклімату необхідно передбачати установки або прилади зволоження та кондиціювання повітря.

3.2.6 Електробезпека

Електробезпека на домашніх робочих місцях важлива для запобігання надзвичайних ситуацій. Основні причини ураження людини електричним струмом на робочому місці:

- дотик до корпусу, периферії комп'ютера, які можуть бути під напругою у результаті ушкодження ізоляції;
- ненормоване використання електричних приладів;
- відсутність знань правил електробезпеки.

Необхідно використовувати лише справні прилади та кабелі, тому що пошкоджені дроти чи розетки можуть спричинити коротке замикання або пожежу. Важливо правильно організувати підключення всіх приладів, використовувати заземлені розетки і не допускати перевантаження електромережі.

Щоб зменшити ризик нещасних випадків, після завершення роботи комп’ютери та інші пристрої слід відключити від мережі. При появі ознак несправності або проблем, наприклад таких як запах гару або іскріння, необхідно негайно відключити пристрій від мережі та звернутися до фахівців. Дотримання цих правил допоможе забезпечити безпеку та комфорт на робочому місці.

3.3 Пожежна безпека

Працюючи вдома за комп'ютером, важливо дотримуватись кількох правил, щоб уникнути пожежі. Пожежі на робочому місці з комп'ютером можуть виникнути через перегрівання електронних пристроїв, коротке замикання в електромережі або несправність електронних компонентів.

Важливо регулярно перевіряти стан електричних розеток та кабелів. Не можна перегрівати розетки, вони повинні бути доступними для охолодження.

Для безпечної експлуатації необхідно уникати перевантаження електричних мереж, тому не слід одночасно підключати до розеток надто багато пристроїв. Крім того, треба регулярно очищати вентиляційні отвори комп'ютера від пилу та бруду, тому що через накопичення пилу існує ризик перегріву системи або займання. Також важливо мати вогнегасник у доступному місці та вміти ним користуватися у разі потреби. Краще використовувати вуглекислотний вогнегасник, так як він не залишає слідів, не проводить електрики і не пошкоджує електроприлади. Використання інших типів вогнегасників може призвести до пошкодження електронних пристроїв через контакт із рідиною або порошком.

Дотримання цих правил допоможе забезпечити безпеку при роботі за комп'ютером вдома та уникнути пожеж.

Загальна мета дотримання всіх ергономічних норм та організації охорони праці, яка розглядалася вище, полягає в створенні комфортного та безпечного робочого середовища, яке буде сприяти здоров'ю та запобіганню можливих хвороб. Тому щоб запобігти можливих хвороб під час роботи за комп'ютером, важливо дотримуватися кількох правил, щоб запобігти негативним наслідкам для здоров'я. Необхідно роботи перерву для відпочинку тривалістю 10-15 хвилин через кожну годину роботи за комп’ютером. У цей час рекомендується робити гімнастику для очей, перемикання уваги з рутинної роботи дозволяє розслабити нервові закінчення, навколоочні м’язи. Також можна зробити загальні фізичні вправи, щоб підтримувати кровообіг та зменшити напругу у м'язах. Можна налаштувати екран комп’ютеру, наприклад встановити фільтри синього світла або спеціальні режими екрану для нічної пори, які також допоможуть зменшити напругу очей.

Треба слідкувати за освітленням робочого місця так, щоб було достатньо світла, але не зайвого сяйва. Використовуйте природне світло та за потреби додаткові джерела освітлення. Регулярно провітрювання кімнати запобігає перегріву та збереженню норм мікроклімату.

Дотримання правильної позиції тіла за робочим столом, контролювання довгого сидіння у неправильному положенні та фіксування правильного положення спини та шиї, допоможе запобігти фізичним травмам.

ВИСНОВКИ

В дипломному проєкті створено Back-End частину автоматизованої веб-системи обліку руху контингенту здобувачів відділення комп'ютерних систем.

Для керування доступами була розроблена система авторизації за електронною поштою та паролем та відповідно для виходу з системи. Всі роути, що відповідають за безпосереднє керування обліком студентів проходять попередню перевірку на авторизацію адміністратора у системі, задля обмеження доступів неавторизованим користувачам. Під цю перевірку також було доданий роут, який відповідає за отримання файлів з локального сховища, що гарантує що неавторизований користувач не може отримати доступ до фотографій студентів та інших файлів, які будуть там зберігатись, що забезпечує деяку безпеку інформації.

При створенні Back-End частини веб-системи для відділення комп’ютерних частин, було задіяно Docker для забезпечення портативності та масштабованості застосунку; в якості серверної мови програмування - PHP 8.2, а саме був використаний фреймворк Laravel 10; як веб-сервер - Nginx, який швидко опрацьовує запити та передає їх далі на обробку, а для зберігання інформації був обраний MySQL 8.0.

В пояснювальній записці розглянуті всі питання, які передбачені технічним завданням дипломного проєктування. Проведено аналіз предметної області, детально описано огляд існуючих аналогів та технології, які використовувалися при створені проєкту. Описане технічне завдання та показано основні етапи розробки Front-End частини веб-застосунку. У частині тестування показано роботу тестування основної ефективності роутів через Postman. Проведено розрахунок економічної створення та впровадження веб-системи, розглянуті питання охорони праці та наведений перелік використаних джерел.

ПЕРЕЛІК ВИКОРИСТАНИХ ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

1. Офіційна документація PHP. [Веб-сайт]. URL: https://www.php.net/manual/en/intro-whatis.php.
2. Офіційна документація Nginx. [Веб-сайт]. URL: https://nginx.org/en/index.html.
3. MySQL 8.0 Release Notes. MySQL. URL: https://dev.mysql.com/doc/relnotes/mysql/8.0/en/.
4. Офіційна документація Docker. [Веб-сайт]. URL: https://docs.docker.com/.
5. Офіційна документація laravel. Laravel. [Веб-сайт]. URL: https://laravel.com/docs/10.x.
6. CRUD-ПЕРЕВІРКА БАЗИ ДАНИХ. qatestlab training center. URL: https://training.qatestlab.com/blog/technical-articles/crud-database-validation/#:~:text=CRUD%20–%20це%20абревіатура,%20яка%20стосується,)%20та%20видалення%20(delete).
7. MVC. brander. URL: https://brander.ua/technologies/mvc.
8. С. І. Доценко. Організація та системи керування базами даних: Навчальний посібник. – «Український державний університетзалізничного транспорту», 2023.
9. О. С. Бунке. Серверні WEB-технології: Навчальний посібник. – «КПІ ім. Ігоря Сікорського», 2023.
10. Eloquent: Getting Started. laravel. URL: https://laravel.com/docs/8.x/eloquent.

ДОДАТОК А. Програмний код основної логіки веб-застосунку

```
// 2024_05_02_124329_create_students_table.php

<?php
use App\Models\Enums\EducationalDegrees;
use App\Models\Enums\EducationForms;
use App\Models\Enums\EnrollmentType;
use App\Models\Enums\FinancingForms;
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;
return new class extends Migration
{
    public function up(): void
    {
        Schema::create('students', function (Blueprint $table) {
            $table->id();
            $table->unsignedBigInteger('group_id');
            $table->unsignedBigInteger('educational_institute_id');
            $table->unsignedBigInteger('educational_program_id');
            $table->unsignedBigInteger('benefits_id')->nullable();
            $table->string('firstname')->index();
            $table->string('lastname')->index();
            $table->string('surname')->index();
            $table->string('phone')->unique();
            $table->string('email')->unique();
            $table->boolean('is_group_leader')->default(false);
            $table->boolean('is_academic_vacation')->default(false);
            $table->boolean('need_dormitory')->default(false);
            $table->date('entry_date');
            $table->integer('enrolled_in_the_course')->default(1);
            $table->integer('financing_form')->default(FinancingForms::CONTRACT->value);
            $table->integer('education_form')->default(EducationForms::DAILY->value);
            $table->integer('enrollment_type')->default(EnrollmentType::TENDER->value);
            $table->integer('educational_degrees')-
>default(EducationalDegrees::JUNIOR_PROFESSIONAL_BACHELOR->value);
            $table->date('birthday');
            $table->integer('gender');
            $table->string('citizenship');
            $table->string('place_of_residence');
            $table->date('graduation_date');
            $table->integer('language');
            $table->timestamps();
            $table->foreign('group_id')
                ->references('id')
                ->on('groups')
                ->onDelete('cascade')
                ->onUpdate('cascade');

            $table->foreign('educational_institute_id')
                ->references('id')
                ->on('educational_institutes')
                ->onDelete('restrict')
                ->onUpdate('cascade');
            $table->foreign('benefits_id')
                ->references('id')
```

```
                ->on('benefits')
                ->onDelete('set null')
                ->onUpdate('cascade');
            $table->foreign('educational_program_id')
                ->references('id')
                ->on('educational_programs')
                ->onDelete('restrict')
                ->onUpdate('cascade');
        });
    }
    public function down(): void
    {
        Schema::dropIfExists('students');
    }
};

// 2024_05_02_124328_create_groups_table.php

<?php
use App\Models\Enums\EducationalDegrees;
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;
return new class extends Migration
{
    public function up(): void
    {
        Schema::create('groups', function (Blueprint $table) {
            $table->id();
            $table->unsignedBigInteger('curator_id')->unique();
            $table->string('name');
            $table->string('code')->unique();
            $table->date('started_date');
            $table->date('finished_date');
            $table->timestamps();
            $table->foreign('curator_id')
                ->references('id')
                ->on('employees')
                ->onDelete('restrict')
                ->onUpdate('cascade');
        });
    }
    public function down(): void
    {
        Schema::dropIfExists('groups');
    }
};

// 2024_05_02_203718_create_media_table.php

<?php
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;
return new class extends Migration
{
    public function up(): void
    {
        Schema::create('media', function (Blueprint $table) {
            $table->id();
            $table->morphs('model');
            $table->string('disk')->index();
```

```
            $table->string('storage_path')->unique();
            $table->string('name')->nullable();
            $table->string('type')->nullable();
            $table->text('file_description')->nullable();
            $table->string('collection')->nullable()->index();
            $table->string('tag')->nullable()->index();
            $table->string('status')->default('private');
            $table->string('file_name');
            $table->string('mime_type');
            $table->unsignedBigInteger('size');
            $table->timestamps();
        });
    }
    public function down(): void
    {
        Schema::dropIfExists('media');
    }
};

// IGroupTableGenerator.php

<?php
namespace App\Services\Files\Interfaces;
use App\Models\Group;
interface IGroupTableGenerator
{
    const FORMAT_XLS = 'xls';
    const FORMAT_XLSX = 'xlsx';
    public function generate(Group $group, string $filename, string $format =
self::FORMAT_XLSX): string;
}

// GroupTableReceptionGroupGenerator.php

<?php
namespace App\Services\Files\XlsFiles\Generators;
use App\Models\Group;
use App\Models\ValueObject\ReceptionInfoValueObject;
use App\Services\Files\HtmlFiles\StudentInfoService;
use App\Services\Files\Interfaces\IGroupTableGenerator;
use App\Services\Helper;
use PhpOffice\PhpSpreadsheet\Spreadsheet;
use PhpOffice\PhpSpreadsheet\Style\Alignment;
use PhpOffice\PhpSpreadsheet\Style\Border;
use PhpOffice\PhpSpreadsheet\Worksheet\Worksheet;
use PhpOffice\PhpSpreadsheet\Writer\Xlsx;
class GroupTableReceptionGroupGenerator implements IGroupTableGenerator
{
    const COLUMN_WIDTHS = [
        'A' => 6,
        'B' => 22,
        'C' => 15,
        'D' => 11,
        'E' => 22,
        'F' => 18,
        'G' => 18,
        'H' => 13,
        'I' => 13,
        'J' => 15,
        'K' => 40,
        'L' => 27,
        'M' => 18,
```

```
        'N' => 15,
        'O' => 3,
        'P' => 8,
    ];
    protected Worksheet $sheet;
    public function __construct(
        protected Spreadsheet $spreadsheet,
        protected StudentInfoService $studentInfoService,
    )
    {
    }
    public function generate(Group $group, string $filename, string $format =
self::FORMAT_XLSX): string
    {
        $this->getBasicStyleSheet();
        $students = $this->getInfo($group);
        $this->generateHead($group);
        $this->getTableHeader();
        $this->addStudentDataRows($students);
        $this->setColumnWidths();
        $this->setFixedRowHeight(4);
        return $this->saveTempFile($filename, $format);
    }
    protected function getInfo(Group $group): array
    {
        $students = $group->getSortedStudents();
        return $this->studentInfoService->getArrReceptionInfoFromStudents($students);
    }
    protected function saveTempFile(string $filename, string $format): string
    {
        $writer = new Xlsx($this->spreadsheet);
        $tempFilePath = tempnam(sys_get_temp_dir(), $filename) . $format;
        $writer->save($tempFilePath);
        return $tempFilePath;
    }
    protected function getBasicStyleSheet(): void
    {
        $this->sheet = $this->spreadsheet->getActiveSheet();
        $this->spreadsheet->getDefaultStyle()->getFont()->setName('Times New Roman');
    }
    protected function generateHead(Group $group): void
    {
        $this->sheet->setCellValue('A1', 'Групи ' . $group->getCourseAndCode());
        $this->sheet->setCellValue('A2', Helper::currentAcademicYear() . ' навчальний
рік');
        $this->sheet->setCellValue('A3', 'Кл.керівник - ' . $group->curator?-
>getLastnameAndInitials());
        $this->sheet->mergeCells('A1:P1');
        $this->sheet->mergeCells('A2:P2');
        $this->sheet->mergeCells('A3:P3');

        $this->sheet->getStyle('A1:P3')->getAlignment()-
>setHorizontal(Alignment::HORIZONTAL_CENTER);
        $this->sheet->getStyle('A1:A3')->getFont()->setBold(true)->setSize(18);
        $this->sheet->getStyle('A2')->getFont()->setBold(true)->setSize(12);
        $this->sheet->getStyle('A3')->getFont()->setBold(true)->setSize(14);
    }
    protected function getTableHeader(): void
    {
        $this->sheet->setCellValue('A4', '№');
        $this->sheet->setCellValue('B4', 'П.І.Б.');
        $this->sheet->setCellValue('C4', 'Дата і місце народження');
```

```
        $this->sheet->setCellValue('D4', 'Основа');
        $this->sheet->setCellValue('E4', 'Серія та номер документа про освіту');
        $this->sheet->setCellValue('F4', 'Рік закінчення, назва навчального закладу');
        $this->sheet->setCellValue('G4', 'Номер та серія паспорта');
        $this->sheet->setCellValue('H4', 'Ким виданий');
        $this->sheet->setCellValue('I4', 'Дата видачі');
        $this->sheet->setCellValue('J4', 'ІНН');
        $this->sheet->setCellValue('K4', 'Адреса');
        $this->sheet->setCellValue('L4', 'Телефон батьків');
        $this->sheet->setCellValue('M4', 'Телефон студента');
        $this->sheet->setCellValue('N4', 'Пільги');
        $this->sheet->setCellV
alue('O4', 'Мова');
        $this->sheet->setCellValue('P4', 'Гуртожиток');
        $columns = ['D', 'E', 'F', 'O'];
        foreach ($columns as $column) {
            $this->sheet->getStyle($column . '4')->getAlignment()->setTextRotation(90);
        }
        $this->applyRowStyles(4, true);
    }
    protected function addStudentDataRows(array $students): void
    {
        $rowIndex = 5;
        foreach ($students as $m => $student) {
            $this->sheet->setCellValue('A' . $rowIndex, $m + 1);
            $this->sheet->setCellValue('B' . $rowIndex, $student->getFullName());
            $this->sheet->setCellValue('C' . $rowIndex, $student->getBirthday()-
>format('d/m/Y'));
            $this->sheet->setCellValue('D' . $rowIndex, $student->getFinancingForm());
            $this->sheet->setCellValue('E' . $rowIndex, $student-
>getEducationDocumentNumber());
            $this->sheet->setCellValue('F' . $rowIndex, $student-
>getEducationalInstitute()->name);
            $this->sheet->setCellValue('G' . $rowIndex, $student->getDocumentNumber());
            $this->sheet->setCellValue('H' . $rowIndex, $student->getIssuedBy());
            $this->sheet->setCellValue('I' . $rowIndex, $student-
>getDocumentIssueDate()->format('d.m.Y'));
            $this->sheet->setCellValue('J' . $rowIndex, $student-
>getIdentificationCodeNumber());
            $this->sheet->setCellValue('K' . $rowIndex, $student->getPlaceOfResidence());
            $this->sheet->setCellValue('L' . $rowIndex, $student->getParentString());
            $this->sheet->setCellValue('M' . $rowIndex, $student->getPhone());
            $this->sheet->setCellValue('N' . $rowIndex, $student->getBenefits()?->name);
            $this->sheet->setCellValue('O' . $rowIndex, $student-
>getFirstLiteralLanguage());
            $this->sheet->setCellValue('P' . $rowIndex, $student-
>isNeedDormitoryForString());
            $this->applyRowStyles($rowIndex);
            $rowIndex++;
        }
    }
    protected function applyRowStyles(int $rowIndex, bool $setBold = false): void
    {
        $columns = range('A', 'P');
        foreach ($columns as $column) {
            $cell = $column . $rowIndex;
            $this->sheet->getStyle($cell)->getAlignment()-
>setHorizontal(Alignment::HORIZONTAL_CENTER);
            $this->sheet->getStyle($cell)->getAlignment()-
>setVertical(Alignment::VERTICAL_CENTER);
            $this->sheet->getStyle($cell)->getAlignment()->setWrapText(true);
            $this->sheet->getStyle($cell)->getFont()->setSize(14);
```

```
            $this->sheet->getStyle($cell)->getBorders()->getAllBorders()-
>setBorderStyle(Border::BORDER_THIN);
        }
        if ($setBold) {
            $this->sheet->getStyle('A' . $rowIndex . ':P' . $rowIndex)->getFont()-
>setBold(true);
        }
    }
    protected function autoSizeColumns(): void
    {
        $columns = range('A', 'P');
        foreach ($columns as $column) {
            $this->sheet->getColumnDimension($column)->setAutoSize(true);
        }
    }
   protected function setColumnWidths(): void
    {
        foreach (self::COLUMN_WIDTHS as $column => $width) {
            $this->sheet->getColumnDimension($column)->setWidth($width);
        }
    }
    protected function setFixedRowHeight(int $startRow, float $height = 90): void
    {
        $highestRow = $this->sheet->getHighestRow();
        for ($row = $startRow; $row <= $highestRow; $row++) {
            $this->sheet->getRowDimension($row)->setRowHeight($height);
        }
    }
}

// GroupTableGenerator

<?php
namespace App\Services\Files;
use App\Models\Group;
use App\Models\ValueObject\TemporaryFileValueObject;
use App\Services\Files\Interfaces\IGroupTableGenerator;
class GroupTableGenerator implements IGroupTableGenerator
{
    public function __construct(
        protected IGroupTableGenerator $tableGenerator
    )
    {
    }
    public function generate(Group $group, string $filename, string $format =
self::FORMAT_XLSX): string
    {
        return $this->tableGenerator->generate($group, $filename, $format);
    }
    public function setTableGenerator(IGroupTableGenerator $tableGenerator): void
    {
        $this->tableGenerator = $tableGenerator;
    }
    protected function getTemporaryFileVO(string $filename, string $format, string
$filePath): TemporaryFileValueObject
    {
        return new TemporaryFileValueObject($filename, $format , $filePath);
    }
}

// Student.php
```

```php
<?php
namespace App\Models;
use App\Models\Enums\DocumentType;
use App\Models\Enums\ParentStatuses;
use Illuminate\Database\Eloquent\Builder;
use Illuminate\Database\Eloquent\Collection;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\BelongsTo;
use Illuminate\Database\Eloquent\Relations\BelongsToMany;
use Illuminate\Database\Eloquent\Relations\HasMany;
use Illuminate\Database\Eloquent\Relations\HasOne;
use Illuminate\Database\Eloquent\Relations\MorphMany;
use Illuminate\Support\Carbon;
class Student extends Model
{
    const IMAGE_TAG = 'student_photo';
    use HasFactory;
    protected $guarded = [];
    public function scopeSearch(Builder $query, mixed $value): Builder
    {
        if ($value) {
            $term = '%' . $value . '%';
            $query->where(function ($query) use ($term) {
                $query
                    ->orWhere('phone', 'like', $term)
                    ->orWhere('email', 'like', $term)
                    ->orWhereRaw("CONCAT(firstname, ' ', lastname, ' ', surname) LIKE ?",
[$term])
                    ->orWhere('place_of_residence', 'like', $term);
            });
        }
        return $query;
    }
    public function decrees(): BelongsToMany
    {
        return $this->belongsToMany(Decree::class, 'student_decrees', 'student_id',
'decree_id')
            ->withTimestamps();
    }
    public function latestDecree(): BelongsToMany
    {
        return $this->belongsToMany(Decree::class, 'student_decrees', 'student_id',
'decree_id')
            ->latest('student_decrees.created_at')
            ->limit(1);
    }
    public function group(): BelongsTo
    {
        return $this->belongsTo(Group::class);
    }
    public function parents(): BelongsToMany
    {
        return $this->belongsToMany(StudentParent::class, 'student_of_parents',
'student_id', 'parent_id')
            ->withPivot('status')
            ->withTimestamps();
    }
    public function mother(): ?StudentParent
    {
        return $this->parents()->wherePivot('status', ParentStatuses::MATHER->value)-
>first();
```

```
    }
    public function father(): ?StudentParent
    {
        return $this->parents()->wherePivot('status', ParentStatuses::FATHER->value)-
>first();
    }
    public function guardian(): ?StudentParent
    {
        return $this->parents()->wherePivot('status', ParentStatuses::GUARDIAN->value)-
>first();
    }

    public function media(): MorphMany
    {
        return $this->morphMany(Media::class, 'model');
    }
    public function getPhotoByTag(): ?Model
    {
        return $this->media()->where('tag', self::IMAGE_TAG)->first();
    }
    public function getPhotoByName(): ?Model
    {
        $name = $this->getStudentPhotoName();
        return $this->media()->where('name', $name)->first();
    }
    public function benefits(): BelongsTo
    {
        return $this->belongsTo(Benefits::class, 'benefits_id');
    }
    public function educationalInstitute(): BelongsTo
    {
        return $this->belongsTo(EducationalInstitute::class, 'educational_institute_id');
    }
    public function getEducationalInstitute(): EducationalInstitute
    {
        return $this->educationalInstitute()->first();
    }
    public function studentCard(): HasOne
    {
        return $this->hasOne(StudentCard::class, 'student_id');
    }
    public function getStudentCard(): ?StudentCard
    {
        return $this->studentCard()->first();
    }
    public function documents(): HasMany
    {
        return $this->hasMany(StudentDocument::class, 'student_id');
    }
    public function getDocumentsByType(int $type): Collection|array
    {
        return $this->documents()->where('document_type', $type)->get();
    }
    public function getPassportDocument(): ?StudentDocument
    {
        return $this->documents()->where('document_type', DocumentType::PASSPORT)-
>first();
    }
    public function getIdentificationCodeDocument(): ?StudentDocument
    {
        return $this->documents()->where('document_type',
DocumentType::IDENTIFICATION_CODE)->first();
```

```
    }
    public function getEducationDocuments(): ?StudentDocument
    {
        return $this->documents()->where('document_type',
DocumentType::EDUCATION_DOCUMENTS)->first();
    }
    public function getStudentPhotoName(): string
    {
        return self::IMAGE_TAG . '_' . $this->id;
    }
    public function getLastnameAndInitials(): string
    {
        return $this->lastname . ' '
            . mb_strtoupper(mb_substr($this->firstname, 0, 1)) . '.'
            . mb_strtoupper(mb_substr($this->surname, 0, 1)) . '.';
    }
    public function educationProgram(): BelongsTo
    {
        return $this->belongsTo(EducationalProgram::class, 'educational_program_id');
    }
    public function getEducationProgram(): ?EducationalProgram
    {
        return $this->educationProgram()->first();
    }
}

// FileController.php

<?php
namespace App\Http\Controllers\Admin;
use App\Http\Controllers\Controller;
use App\Models\Group;
use App\Models\Student;
use App\Services\Files\Enums\FileTypes;
use App\Services\Files\GroupTableGenerator;
use App\Services\Files\HtmlFiles\StudentInfoService;
use Carbon\Carbon;
use Exception;
use Illuminate\Container\Container;
use Illuminate\Contracts\View\Factory;
use Illuminate\Contracts\View\View;
use Illuminate\Foundation\Application;
use Illuminate\Http\RedirectResponse;
use Symfony\Component\HttpFoundation\BinaryFileResponse;
use Throwable;
class FileController extends Controller
{
    public function __construct(
        protected StudentInfoService $studentInfoService,
        protected Container          $container
    )
    {
    }
    public function studentCard(Student $student):
View|Application|Factory|\Illuminate\Contracts\Foundation\Application
    {
        $vo = $this->studentInfoService->getStudentIndividualPlanVO($student);
        return view('files.individual-plan', compact('vo'));
    }

    public function studentPersonalCard(Student $student):
View|Application|Factory|\Illuminate\Contracts\Foundation\Application
```

```
    {
        $vo = $this->studentInfoService->getStudentCardVO($student);
        return view('files.personal-card', compact('vo'));
    }
    public function groupReceptionSample(Group $group):
BinaryFileResponse|RedirectResponse
    {
        try {
            $groupTableGenerator = $this->container->get(GroupTableGenerator::class);
            $filename = 'reception_' . $group->getCourseAndCode() . ' ' . Carbon::now()-
>format('d-m-Y H-i-s');
            $format = FileTypes::TABLE_XLSX->value;
            $filepath = $groupTableGenerator->generate($group, $filename, $format);
            $return = response()->download($filepath, $filename . $format)-
>deleteFileAfterSend();
        } catch (Throwable $e) {
            logger($e->getMessage());
            $return = redirect()->back()->with('error', 'Сталася помилка, спробуйте
пізніше');
        }
        return $return;
    }
}
```

ДОДАТОК Б. Слайди мультимедійної презентації

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

РП.07.16.000.ДП

ДИПЛОМНИЙ ПРОЄКТ

На тему:

Розробка Back-End частини автоматизованої веб-системи обліку руху контингенту здобувачів відділення комп'ютерних систем

Ніколаєва Валентина Петровича
Спеціальність: 121 - «Комп'ютерна інженерія»
Освітньо-професійна програма: «Інженерія програмного забезпечення»

СХЕМА ВЗАЄМОЗВ'ЯЗКІВ МІЖ БАЗОВИМИ СУТНОСТЯМИ СЕРВІСУ

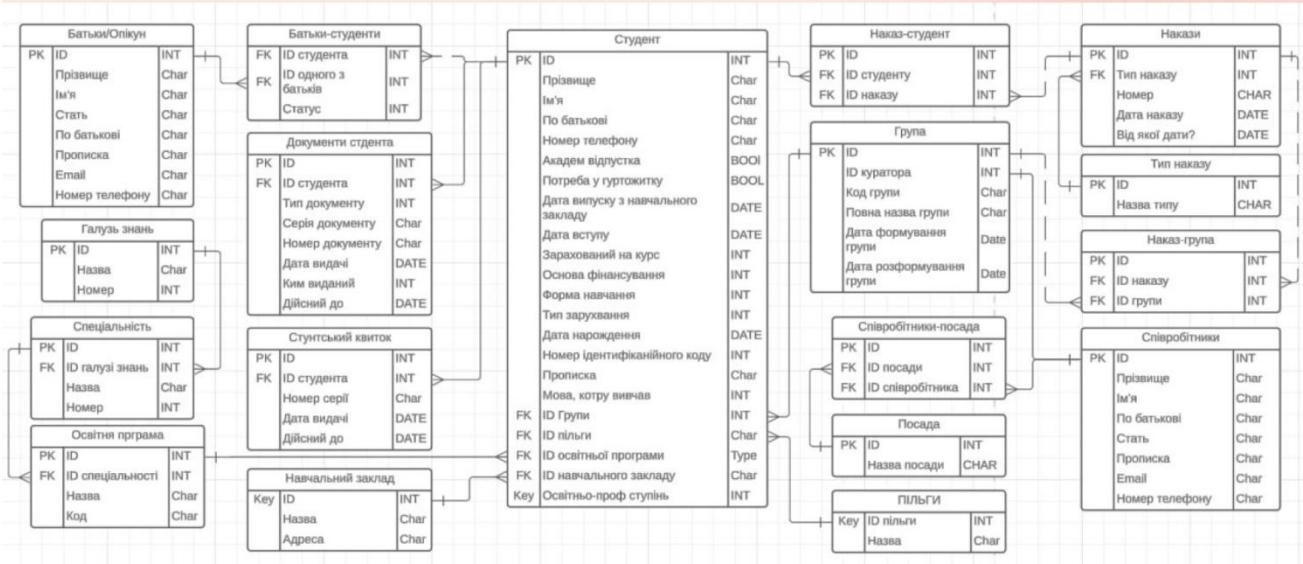


СХЕМА БАЗОВИХ ТАБЛИЦЬ ВЕБ-ЗАСТОСУНКУ

Файли

PK	ID	INT
	Тип моделі / таблиця	Char
	ID запису / моделі	INT
	Диск	Char
	Шлях	Char
	Назва	Char
	Реальна назва файлу	Char
	Опис	Char
	Колекція	Char
	Тег	Char
	Тип	Char
	MIME-Тип	Char
	Розмір файлу	Char

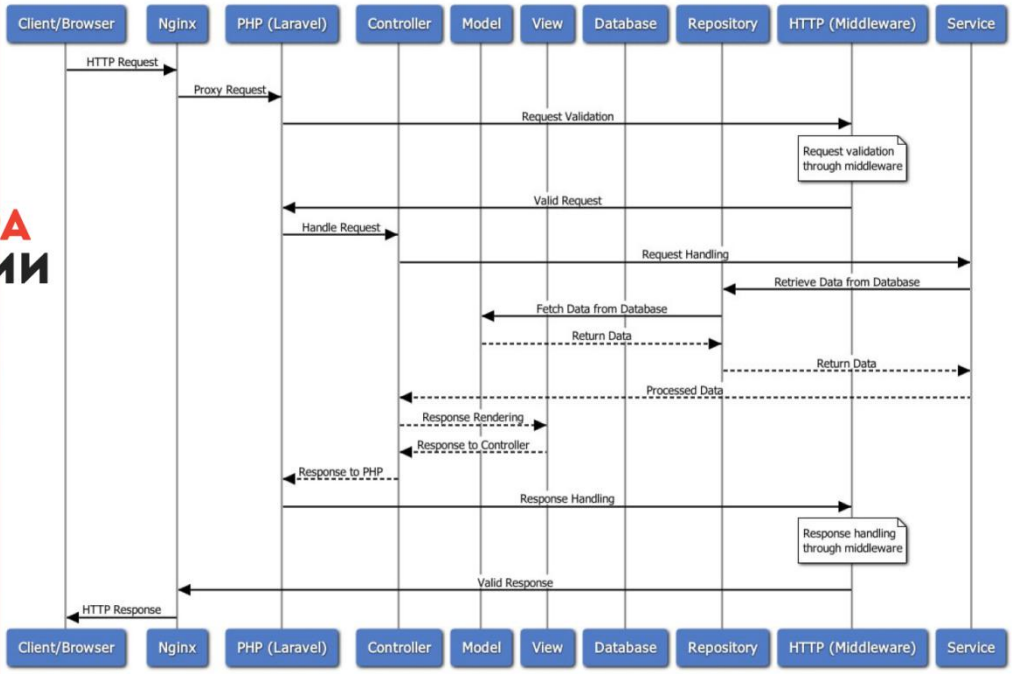
Адміністратор

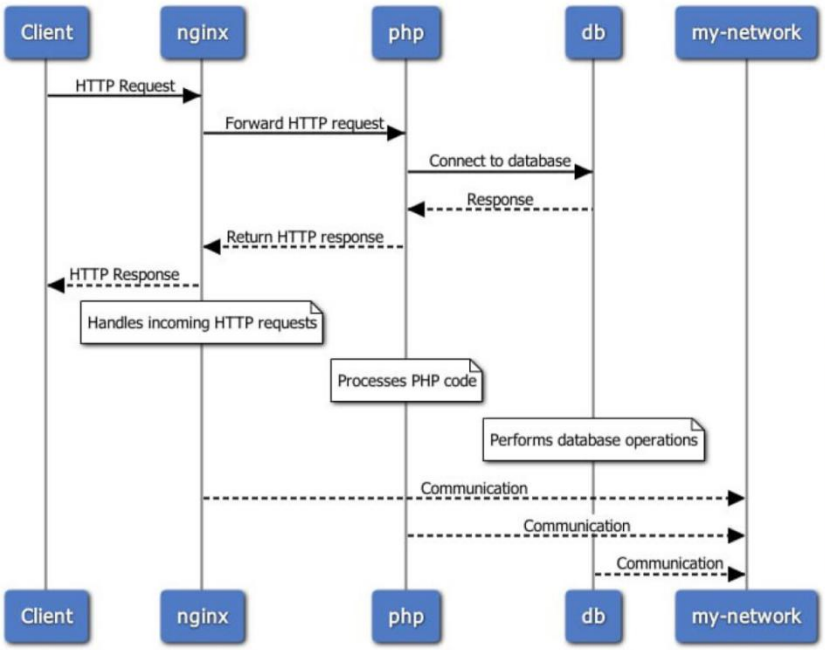
PK	ID	INT
	Прізвище	Char
	Ім'я	Char
	Пошта	Char
	Телефон	Char
	Пароль	Char

Міграції

PK	ID	INT
	Назва міграції	Char
	Номер партії	Char

АРХІТЕКТУРА ВЕБ-СИСТЕМИ

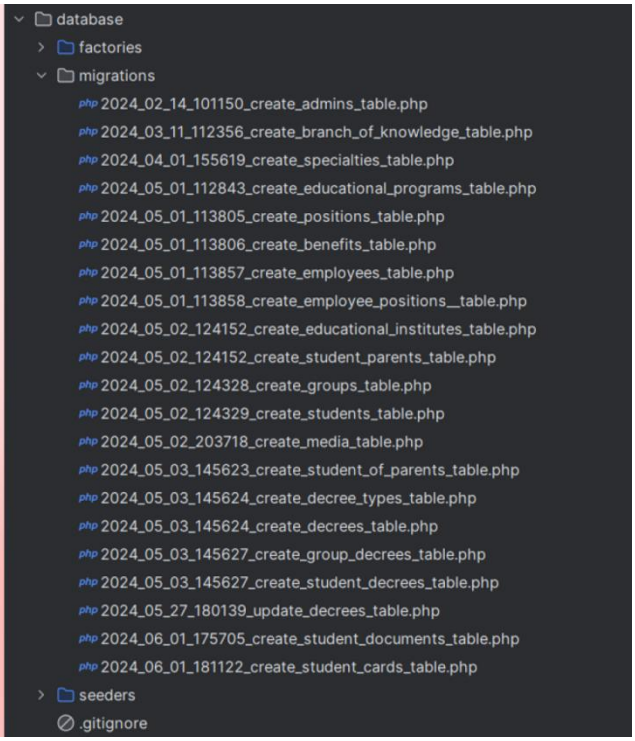




DOCKER-СХЕМА ВЕБ-ЗАСТОСУНКУ

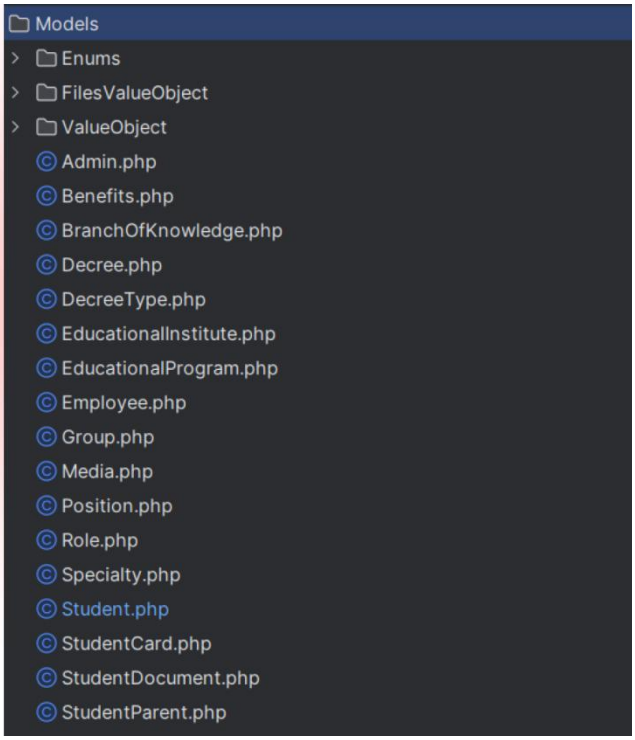
СТВОРЕНІ МІГРАЦІЇ

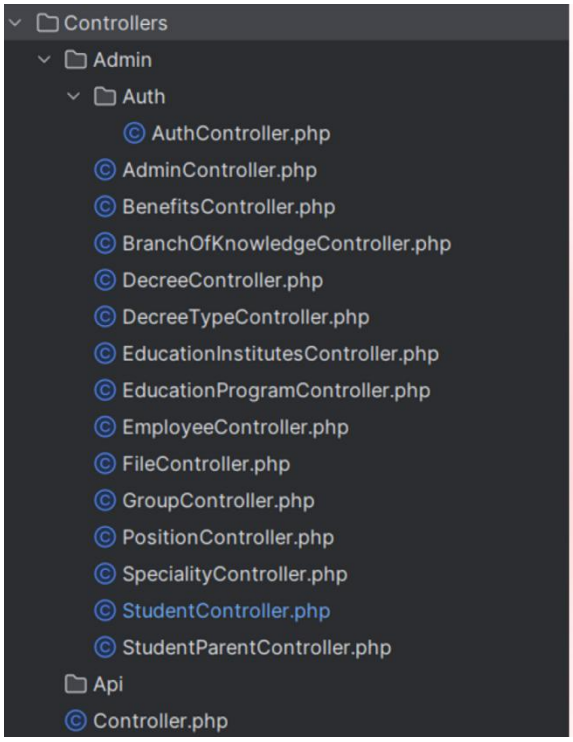
```
public function up(): void
{
    Schema::create('specialties', function (Blueprint $table) {
        $table->id();
        $table->unsignedBigInteger('branch_knowledge_id');
        $table->string('name')->unique();
        $table->integer('number')->unique();
        $table->timestamps();

        $table->foreign('branch_knowledge_id')
            ->references('id')
            ->on('branch_of_knowledge')
            ->onDelete('cascade')
            ->onUpdate('cascade');
    });
}

public function down(): void
{
    Schema::dropIfExists('specialties');
}
```



СТВОРЕНІ МОДЕЛІ

```
class Specialty extends Model
{
  use HasFactory;
  protected $table = 'specialties';

  public function scopeSearch(Builder $query, mixed $value): Builder
  {
    if ($value) {
      $term = '%' . $value . '%';
      $query->where(function ($query) use ($term) {
        $query
          ->orWhere('number', 'like', $term)
          ->orWhere('name', 'like', $term)
          ->orWhere('id', 'like', $term);
      });
    }

    return $query;
  }

  public function branchOfKnowledge(): BelongsTo
  {
    return $this->belongsTo(BranchOfKnowledge::class,
'branch_knowledge_id');
  }
  public function educationalPrograms(): HasMany
  {
    return $this->hasMany(EducationalProgram::class);
  }
}
```





СТВОРЕНІ КОНТРОЛЕРИ

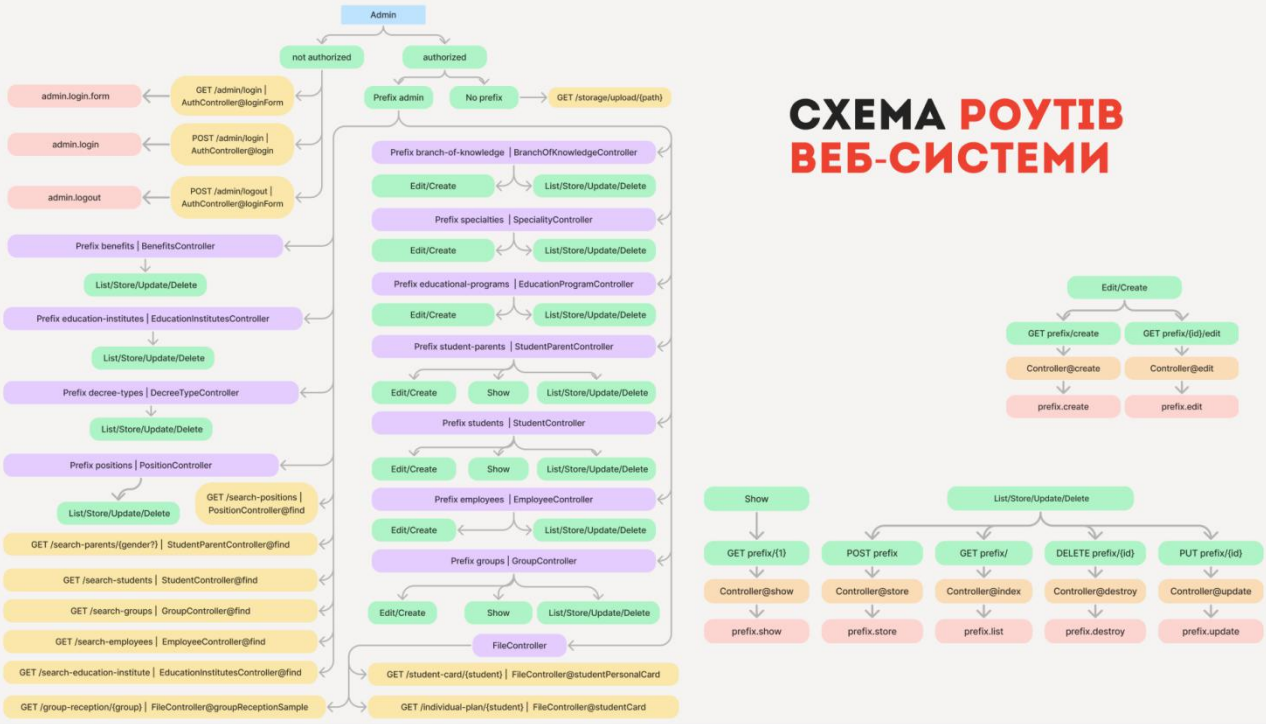
```
class SpecialityController extends Controller
{
  public function __construct(
    protected SpecialtyRepository $repository,
    protected BranchOfKnowledgeRepository $branchOfKnowledgeRepository
  ) {
  }

public function index(): View|Application|Factory|
\Illuminate\Contracts\Foundation\Application
{
  $specialties = Specialty::search(request('search'))->latest('id', 'desc')->paginate(10);
  return view('admin.specialties.list', compact('specialties'));
}

public function create(): View|Application|Factory|
\Illuminate\Contracts\Foundation\Application
{
  $branches = $this->branchOfKnowledgeRepository->getAll();
  return view('admin.specialties.create', compact('branches'));
}
public function store(SpecialtyRequest $request): RedirectResponse
{
  try {
    $this->repository->create($request->getNumber(), $request->getName(),
$request->getBranchOfKnowledge());
    $return = redirect()->route('admin.specialties.list')->with('success', 'Спеціальність
успішно створено');
  } catch (Throwable $e) {
    $return = redirect()->route('admin.specialties.list')->with('error', 'Системний збій,
зверніться до розробника');
  }
  return $return;
}
```

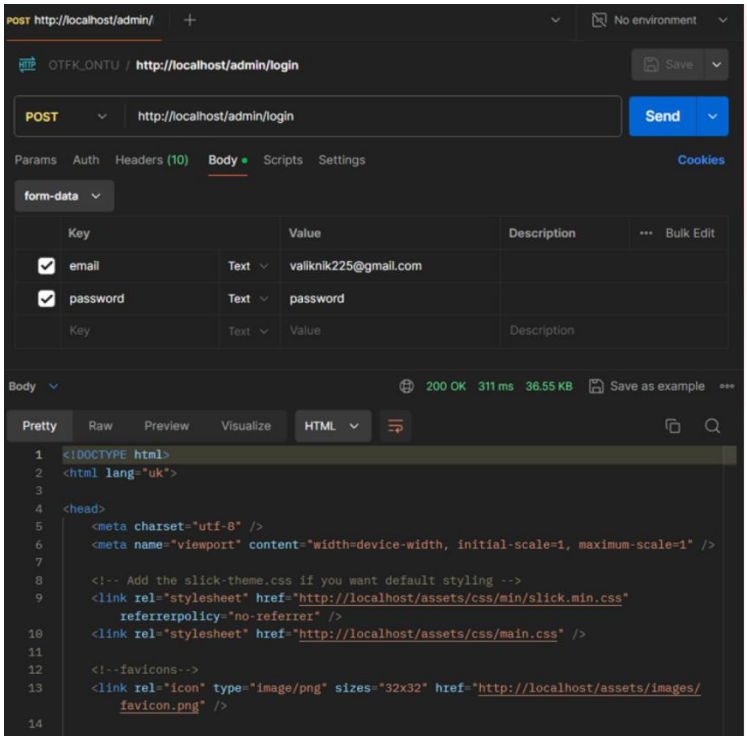
СТРАТЕГІЯ ГЕНЕРАЦІЇ ТАБЛИЦЬ ДЛЯ ГРУПИ

```php
<?php

namespace App\Services\Files;

use App\Models\Group;
use App\Models\ValueObject\TemporaryFileValueObject;
use App\Services\Files\XlsFiles\Interfaces\IGroupTableGenerator;
use Carbon\Carbon;

class GroupTableGenerator implements IGroupTableGenerator
{
  public function __construct(
    protected IGroupTableGenerator $tableGenerator
  )
  {
  }

  public function generate(Group $group, string $filename, string $format =
self::FORMAT_XLSX): string
  {
    return $this->tableGenerator->generate($group, $filename, $format);
  }

  public function setTableGenerator(IGroupTableGenerator $tableGenerator): void
  {
    $this->tableGenerator = $tableGenerator;
  }

  protected function getTemporaryFileVO(string $filename, string $format, string
$filePath): TemporaryFileValueObject
  {
    return new TemporaryFileValueObject($filename, $format , $filePath);
  }
}
```

```php
public function groupReceptionSample(Group $group): BinaryFileResponse|
RedirectResponse
{
  try {
    $groupTableGenerator = $this->container->get(GroupTableGenerator::class);

    $filename = 'reception_' . $group->getCourseAndCode() . ' ' .
Carbon::now()->format('d-m-Y H-i-s');
    $format = FileTypes::TABLE_XLSX->value;
    $filepath = $groupTableGenerator->generate($group, $filename, $format);

    $return = response()->download($filepath, $filename .
$format)->deleteFileAfterSend();
  } catch (Throwable $e) {
    logger($e->getMessage());
    $return = redirect()->back()->with('error', 'Сталася помилка, спробуйте пізніше');
  }
  return $return;
}
```

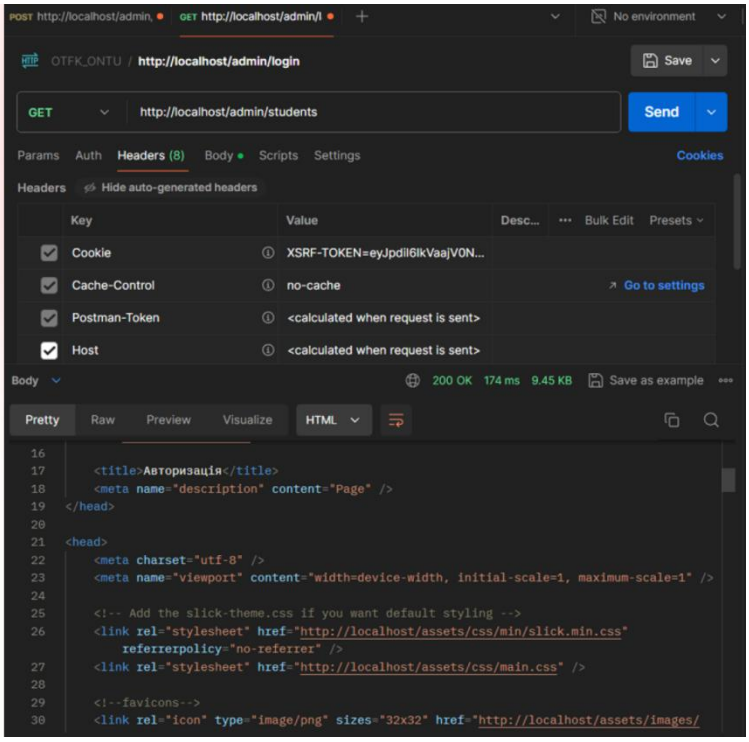
СХЕМА РОУТІВ ВЕБ-СИСТЕМИ

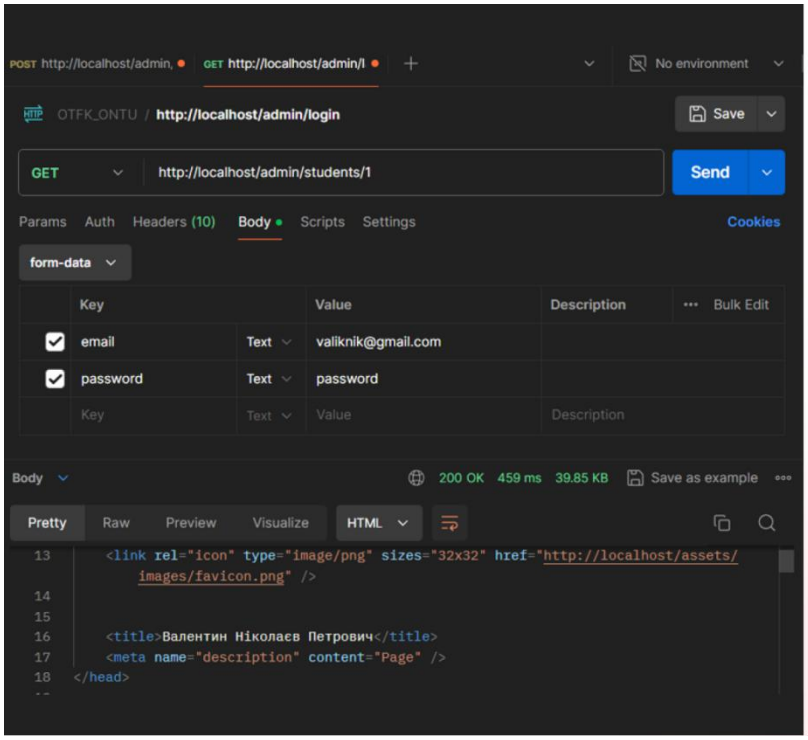




РЕЗУЛЬТАТ НАДСИЛАННЯ ЗАПИТУ НА АВТОРИЗАЦІЮ

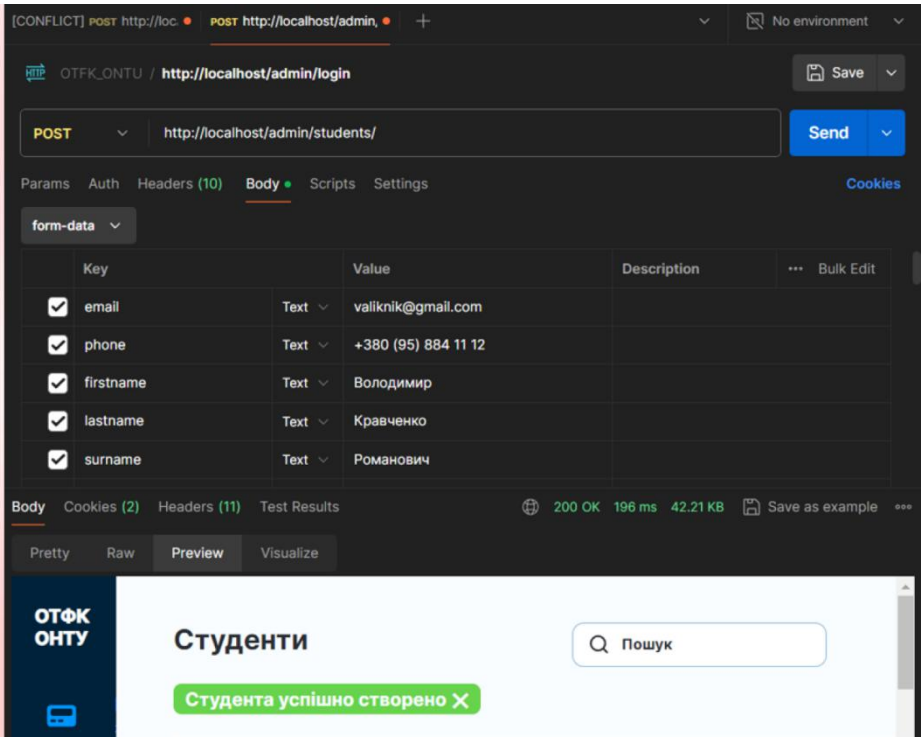
РЕЗУЛЬТАТ НАДСИЛАННЯ ЗАПИТУ НА ОТРИМАННЯ СТОРІНКИ СПИСКУ СТУДЕНТІВ ПРИ НЕ АВТОРИЗОВАНОМУ КОРИСТУВАЧЕВІ

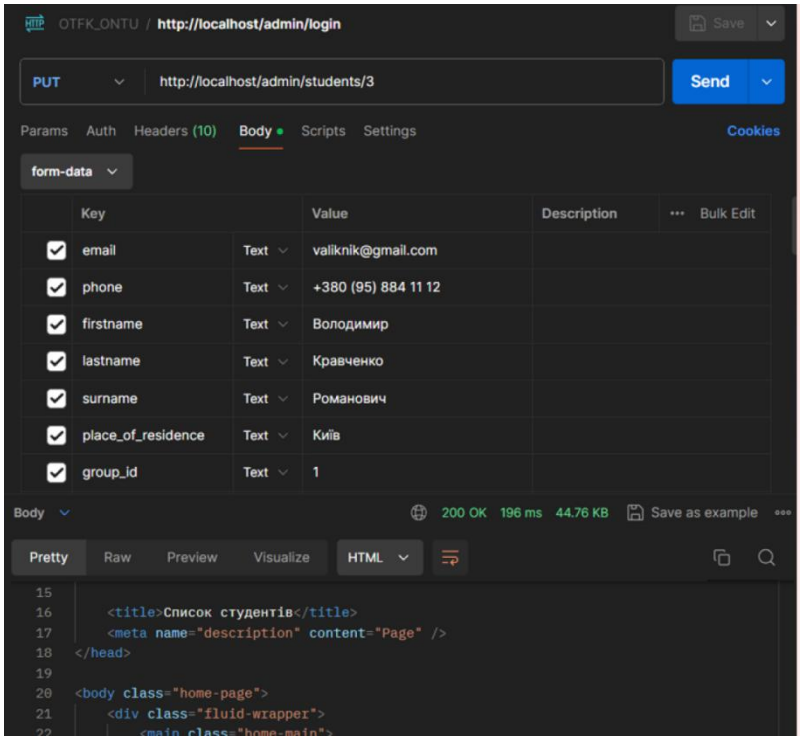




РЕЗУЛЬТАТ НАДСИЛАННЯ ЗАПИТУ НА ОТРИМАННЯ СТОРІНКИ ПЕРЕГЛЯДУ ПЕВНОГО СТУДЕНТА ПРИ АВТОРИЗОВАНОМУ КОРИСТУВАЧЕВІ

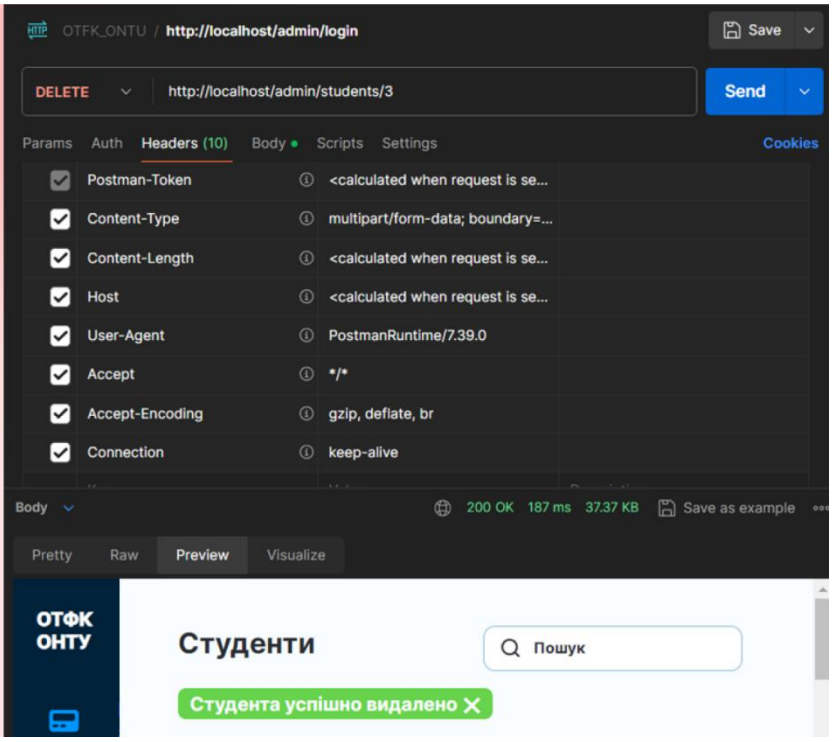
РЕЗУЛЬТАТ НАДСИЛАННЯ ЗАПИТУ НА СТВОРЕННЯ СТУДЕНТУ



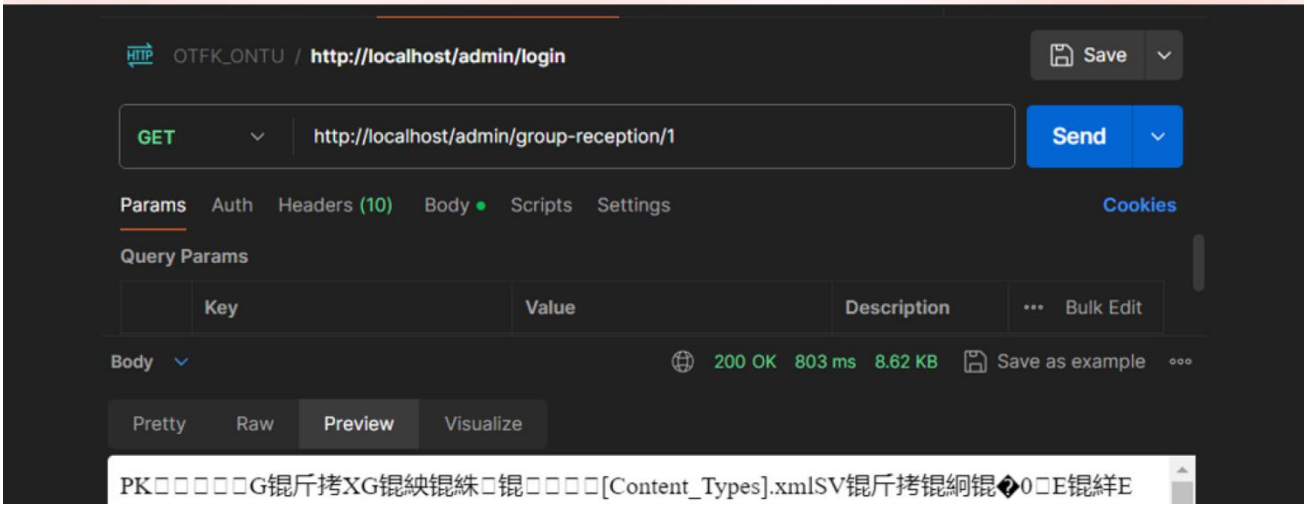


РЕЗУЛЬТАТ НАДСИЛАННЯ ЗАПИТУ НА ОНОВЛЕННЯ СТУДЕНТУ

РЕЗУЛЬТАТ НАДСИЛАННЯ ЗАПИТУ НА ВИДАЛЕННЯ СТУДЕНТУ



РЕЗУЛЬТАТ ГЕНЕРАЦІЇ ФАЙЛУ ЗАПИТУ НА ВИДАЛЕННЯ СТУДЕНТУ



В роботі було проведено економічний розрахунок:

Загальні витрати (Вз) на створення веб-системи розраховуються за формулою:

Де **Вр** – витрати на розробку сайту;
Вв – витрати на впровадження сайту;
Ве – витрати на експлуатацію сайту;

Вз = Вр+ Вв+ Ве

Вз = Вр+ (Вв+ Ве) = 40 723 грн.

Автоматизована веб-система обліку руху контингенту здобувачів не буде приносити дохід, оскільки це не комерційний проєкт. Однак вона принесе велику користь у контексті роботи відділення комп'ютерних систем:

1. Централізоване та захищене зберігання даних
2. Ефективне управління інформацією
3. Оптимізація технологічних процесів
4. Покращення комунікації
5. Підвищення продуктивності
6. Формування позитивного іміджу

В розділі охорони праці були розглянуті:

Охорона праці спрямована на забезпечення здоров'я та безпеки працівників.

При роботі за комп'ютером основними проблемами є:

- порушення зору
- болі в спині, шиї та плечах
- синдром зап'ястного каналу

Для запобігання цим проблемам важливо дотримуватися ергономічних рекомендацій, регулярно робити перерви та займатися фізичною активністю.

Гігієнічні вимоги до робочого місця:
належна організацію приміщення
освітлення
мікроклімату
рівня шуму

Робоче місце повинно бути просторим, з ергономічними меблями та правильно розташованим монітором.

- температура повітря – 22-25°C
- відносна вологість 40-60%
- рівень шуму – до 65 дБА.

Для забезпечення пожежної безпеки важливо уникати перевантаження електромережі, регулярно оглядати вилки та дроти, забезпечувати вентиляцію комп'ютера та очищати його від пилу. При виникненні пожежі слід використовувати вуглекислотний вогнегасник типу ВВК.

ДЯКУЮ ЗА УВАГУ!

Чи є якісь питання у шановної комісій?

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

ВІДГУК

керівника на дипломний проект здобувача (здобувачки) освіти

відділення комп'ютерних систем

Ніколаєва Валентина Петровича

(прізвище, ім`я та по батькові)

Спеціальність: *121 «Інженерія програмного забезпечення»*

Освітня програма: *«Розробка програмного забезпечення»*

Тема дипломного проекту: *Розробка Back-End частини автоматизованої веб-системи обліку руху контингенту здобувачів відділення комп'ютерних систем*

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ

а) обсяг і якість виконання проекту (графічного матеріалу і розрахунково-пояснювальної записки) *Дипломний проект виконано відповідно технічному завданню. Пояснювальна записка до дипломного проекту містить 92 сторінки. У пояснювальній записці описано етапи розробки Back-End частини автоматизованої веб-системи обліку руху контингенту здобувачів відділення комп'ютерних систем засобами Laravel та MySQL. Графічна частина складається з окремих слайдів, оформлених у вигляді презентації, передбачених технічним завданням. Якість виконання пояснювальної записки та слайдів відмінна.*

б) самостійність роботи над проектом: *Протягом виконання дипломного проекту здобувач освіти Ніколаєв Валентин поступово та послідовно виконував всі етапи, проявляв ініціативу в створенні загальної концепції та реалізації роботи. Всі роботи здобувач освіти виконував самостійно, з оглядом на рекомендації керівника.*

в) теоретична підготовка випускника (випускниці): *Здобувач освіти Ніколаєв Валентин під час роботи над дипломним проектом вивчив достатньо багато літературних та інтернет-джерел за даною тематикою.*

Вважаю, що теоретична підготовка дипломника достатня і він готовий до захисту проекту.

г) вміння розв'язувати виробничі та конструкторські питання *Під час виконання дипломного проекту здобувач освіти Ніколаєв Валентин показав вміння організовано працювати над поставленим завданням, застосовувати знання у галузі програмування та математики, розробляти, встановлювати та налаштовувати спеціалізоване програмне забезпечення, оформлювати слайди та складати презентації, користуючись сучасними комп'ютерними програмними засобами, такими як MS VS Code, Laravel, MySQL, Microsoft PowerPoint, Microsoft Visio та ін.*

Оцінка розрахункової частини *Відмінно*

Оцінка графічної частини *Відмінно*

Загальна оцінка *Відмінно*

Прізвище, ім'я, по батькові керівника дипломного проекту *Жадан Артур Сергійович*

Місце роботи і посада керівника дипломного проекту *ВСП «Одеський технічний фаховий `коледж ОНТУ», викладач спецдисциплін циклової комісії комп'ютерної техніки та програмної інженерії*

Підпис

«10» червня 2024 р.

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

РЕЦЕНЗІЯ

на дипломний проект здобувача (здобувачки) освіти

відділення комп'ютерних систем

Ніколаєва Валентина Петровича

(прізвище, ім`я та по батькові)

Спеціальність *121 «Інженерія програмного забезпечення»*

Освітня програма *«Розробка програмного забезпечення»*

Керівник дипломного проекту (роботи) *Жадан Артур Сергійович*

(прізвище, ім`я та по батькові)

Тема дипломного проекту (роботи) *Розробка Back-End частини автоматизованої веб-системи обліку руху контингенту здобувачів відділення комп'ютерних систем*

Обсяг розрахунково-пояснювальної записки *92* сторінок

Обсяг графічної (презентаційної) частини *20* аркушів (слайдів)

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ (РОБОТИ)

а) заключення про ступінь відповідності виконаного дипломного проекту завданню

Представлений на рецензію дипломний проект відповідає затвердженій темі та виконаний відповідно технічному завданню. Дипломний проект присвячений проблемі обліку руху контингенту здобувачів на відділені та складається з пояснювальної записки, додатку з програмним кодом та мультимедійної презентації, що містить приклади роботи програми.

б) характеристика виконання кожного розділу дипломного проекту

Пояснювальна записка складається з основного розділу (аналізу предметної області, проектування застосунку, реалізації застосунку, тестування застосунку), економічного розділу, розділу охорони праці та додатків. Перелічені розділи поетапно охоплюють розробку, виконані докладно та обґрунтовано. Розділ охорони праці містить загальну інформацію та вимоги до техніки безпеки оператора КТ. Економічний розділ проекту містить розрахунок витрат на НДР та реалізацію проекту.

в) оцінка якості виконання пояснювальної записки та графічної частини дипломного проекту

Графічна частина складається з 20 слайдів мультимедійної презентації, виконаної у програмному продукті MS PowerPoint, які містять ілюстративні схеми, скріншоти роботи програмного застосунку, передбачені технічним завданням. Пояснювальна записка виконана акуратно та у відповідності до норм. Якість виконання графічної частини проекту та пояснювальної записки відмінна, розробку виконано у повному обсязі.

г) перелік позитивних якостей дипломного проекту *Реалізовано Back-End частину автоматизованої веб-системи, що дозволяє генерувати заповнені бланки документів про студентів на основі інформації, що зберігається у базі даних.*

Під час проектування БД використовувались зв'язки багато до багатьох. Під час розробки веб-системи використовувались концепції ООП – патерни проектування та принципи SOLID.

д) основні недоліки дипломного проекту

Back-End частина автоматизованої веб-системі містить реляційну таблицю "Студент", що виглядає перевантаженою.

Незважаючи на тему дипломної роботи, для цілісності сприйняття матеріалу пояснювальної записки рекомендовано додати розділ про Front-End частину.

Оцінка розрахункової частини *Відмінно*

Оцінка графічної частини *Відмінно*

Загальна оцінка *Відмінно*

Прізвище, ім'я, по батькові рецензента *Царьов Роман Юрійович*

Місце роботи і посада рецензента *Державний університет інтелектуальних технологій і зв'язку, ст. викладач, зав. кафедри комп'ютерної інженерії та інформаційних систем*

Підпис

14 червня 2024 р.

UNICHECK
by Turnitin

Ім'я користувача:
Катерина Григоріївна Краснокутська

ID перевірки:
1016330007

Дата перевірки:
07.06.2024 00:04:23 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
07.06.2024 07:41:26 EEST

ID користувача:
100011688

Назва документа: **4РП-07_Ніколаєв_В**

Кількість сторінок: **53** Кількість слів: **9392** Кількість символів: **70811** Розмір файлу: **2.98 MB** ID файлу: **1016129417**

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

4.35%
Схожість

Найбільша схожість: 1.11% з Інтернет-джерелом (https://idoc.pub/documents/php-microservices-143gyyor1onj)

4.35% Джерела з Інтернету | **399** ..... Сторінка 55

Не знайдено джерел з Бібліотеки

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Підозріле форматування | **14 сторінок**

ДОЗВІЛ
НА РОЗМІЩЕННЯ
ВИПУСКНОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
(ДИПЛОМНОГО ПРОЕКТУ)
В ЕЛЕКТРОННОМУ РЕПОЗИТАРІЇ ВСП «ОТФК ОНТУ»

Ми, що нижче підписалися,

Ніколаєв Валентин Петрович,
здобувач освіти гр. 4РП-07, та

Жадан Артур Сергійович,
керівник дипломного проекту,

не заперечуємо щодо розміщення електронного варіанту пояснювальної записки до дипломного проекту фахового молодшого бакалавра на тему:

«Розробка Back-End частини автоматизованої веб-системи обліку руху контингенту здобувачів відділення комп'ютерних систем» (автор роботи – Ніколаєв В.П., керівник роботи – Жадан А.С.)

виконаного у ВСП «Одеський технічний фаховий коледж Одеського національного технологічного університету» в 2024 році, у повному обсязі в електронному репозитарії ВСП «ОТФК ОНТУ» для вільного доступу через мережу Інтернет.

Несемо відповідальність за ідентичність електронного та друкованого варіантів випускної кваліфікаційної роботи і даємо згоду на обробку персональних даних.

Виконавець _____ / Ніколаєв В.П. /

Керівник _____ / Жадан А.С. /

«10» червня 2024 р.