НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра технічної кібернетики

«На правах рукопису»
УДК 004.43

«До захисту допущено»

Завідувач кафедри
_____ І.Р. Пархомей
(підпис)
“___”_____2018 р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності 126 «Інформаційні системи та технології»

на тему: Веб-орієнтована система для навчання студентів та керування навчальним процесом

Виконав: студент другого курсу, групи ІК-72мп
(шифр групи)

Сороченко Руслан Андрійович _____
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник доцент, к.т.н., доцент Крилов Є. В. _____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант _____ _____ _____
(назва розділу) (науковий ступінь, вчене звання, , прізвище, ініціали) (підпис)

Рецензент_____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2018 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра технічної кібернетики

Рівень вищої освіти – другий (магістерський)

Спеціальність 126 «Інформаційні системи та технології»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ І.Р. Пархомей
(підпис)

«___»_____2018 р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Сороченку Руслану Андрійовичу
(прізвище, ім'я, по батькові)

1. Тема дисертації «Веб-орієнтована система для навчання студентів та керування навчальним процесом»,
науковий керівник дисертації доцент, к.т.н., доцент Крилов Є. В.,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «07» листопада 2018 р. № 4112-с

2. Термін подання студентом дисертації _____

3. Об'єкт дослідження – *процес роботи веб-додатку для навчання студентів та керування навчальним процесом та час виконання запитів користувачів.*

4. Предмет дослідження – *методи та технології оптимізації роботи веб-додатку, які б забезпечили його належну швидкодію.*

5. Перелік завдань, які потрібно розробити – *аналіз існуючих рішень; вибір технологій для розробки системи та їх опис; розробка системи для навчання студентів та керування навчальним процесом; оптимізація роботи системи та забезпечення її належної швидкодії; тестування системи.*

6. Орієнтовний перелік ілюстративного матеріалу – шість плакатів

7. Орієнтовний перелік публікацій – дві публікації

8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

9. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	Аналіз предметної області та постановка задачі	03.09.2018 р.	
2	Вибір технологій для розробки системи	10.09.2018 р.	
3	Проектування бази даних	17.09.2018 р.	
5	Розробка основних модулів	28.09.2018 р.	
6	Оптимізація роботи системи	15.10.2018 р.	
7	Тестування розробленої системи	24.10.2018 р.	
8	Написання та оформлення пояснювальної записки до дипломної роботи	01.11.2018 р.	
9	Висновки	15.11.2018 р.	

Студент _____ Сороченко Р. А.
(підпис) (ініціали, прізвище)

Науковий керівник дисертації _____ Крилов Є. В.
(підпис) (ініціали, прізвище)

АНОТАЦІЯ

У роботі було розглянуто проблему у області оптимізації веб-додатків.

Було проаналізовано аналоги систем для навчання студентів та визначено їх основні переваги та недоліки. На основі отриманих результатів було визначено основні необхідні модулі, які повинна містити дана системи та способи її оптимізації та забезпечення належної швидкодії.

В результаті виконання дипломної дисертації було розроблено систему для навчання студентів та керування навчальним процесом, здійснено її оптимізацію та значно підвищено швидкодію за рахунок правильно продуманої структури бази даних, впровадження додаткової системи кешування даних, оптимізації запитів до бази даних, створення механізму виконання складних та трудомістких операцій у автоматичному режимі. Розгорнуто розроблену систему на сервері Amazon та створено базовий образ для подальшого її масштабування при збільшенні навантаження.

Ключові слова: оптимізація швидкодії роботи додатку, система для навчання, утиліта cron, Laravel, EC2, Amazon, PHP, MySQL, MVC.

Розмір пояснювальної записки – 93 аркуші, містить 30 ілюстрацій, 24 таблиці, 7 додатків.

ABSTRACT

Examines the problem of the optimization of web-applications.

Analogues of systems for student learning were analyzed and their main advantages and disadvantages were determined. On the basis of the obtained results, the basic necessary modules were defined.

As a result of the diploma thesis, a system for teaching students and managing the learning process was developed, its optimization was carried out and the speed was significantly increased due to a well-designed database structure, the introduction of an additional system of data caching, database query optimization, and the creation of a mechanism for the execution of complex and time-consuming operations. in automatic mode. The developed system is deployed on the Amazon server and the base image is created to further scale it with increasing load.

Keywords: optimization of the performance of web-applications, system for training students, utility cron, Laravel, EC2, Amazon, PHP, MySQL, MVC.

Explanatory note size – 93 pages, contain 30 illustrations, 24 tables, 7 applications.

ПОЯСНЮВАЛЬНА ЗАПИСКА

до магістерської дисертації

на тему: *Веб-орієнтована система для навчання студентів та керування навчальним процесом*

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ 8
ВСТУП 9
РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ І ПОСТАНОВКА ЗАДАЧІ 11
1.1. Об’єкт та предмет дослідження 11
1.2. Аналіз існуючих систем 11
1.3. Постановка задачі 17
Висновки до розділу 18
РОЗДІЛ 2. ВИБІР ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ СИСТЕМИ ДЛЯ НАВЧАННЯ СТУДЕНТІВ ТА КЕРУВАННЯ НАВЧАЛЬНИМ ПРОЦЕСОМ 19
2.1. HTML, CSS та фреймворк Bootstrap 19
2.2. JavaScript 22
2.3. PHP та фреймворк Laravel 23
2.4. MySQL 26
Висновки до розділу 27
РОЗДІЛ 3. РОЗРОБКА СИСТЕМИ ДЛЯ НАВЧАННЯ СТУДЕНТІВ ТА КЕРУВАННЯ НАВЧАЛЬНИМ ПРОЦЕСОМ 29
3.1. Архітектура системи 29
3.2. Структура бази даних 35
3.2.1. Структура таблиць та зв’язків між ними 37
3.2.2. Створення індексів для полів таблиць 40
3.3. Розробка моделей для роботи з базою даних 41
3.4. Розробка RESTfull API 44
3.5. Чітка валідація даних та механізм обробки помилок 47
3.6. Розробка логіки роботи основних модулів 49
3.7. Впровадження системи кешування даних Redis 53
3.8. Виконання складних задач за допомогою cron 54
3.9. Написання unit тестів до програми 57
3.10. Масштабування серверів за допомогою Amazon EC2 60
3.11. Аналіз отриманих результатів та швидкодії системи 62
Висновки до розділу 63

РОЗДІЛ 4. МАРКЕТИНГОВИЙ АНАЛІЗ СТАРТАП-ПРОЕКТУ 64
4.1. Опис ідеї проекту 64
4.2. Технологічний аудит ідеї проекту 66
4.3. Аналіз ринкових можливостей запуску стартап-проекту 67
4.4. Розроблення ринкової стратегії проекту 71
4.5. Розроблення маркетингової програми стартап-проекту 74
Висновки до розділу 75
ВИСНОВКИ 77
ПЕРЕЛІК ПОСИЛАНЬ 78
ДОДАТКИ 79

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

PHP – hypertext preprocessor

REST – representational state transfer

API – application programming interface

HTTP – hypertext transfer protocol

MVC – model-view-controller

JSON – JavaScript object notation

PSR – PHP standards recommendations

TDD – test-driven development

СУБД – система управління базою даних

ВСТУП

В наш час технології розвиваються швидкими темпами, а разом із ними і покращується рівень життя населення. Однієї із таких прогресивних технологій є розвиток мережі інтернет. Вона налічує велику кількість інформації та надає зручний доступ до неї.

Кожний користувач може ввести будь-який пошуковий запит та отримати відповідь на своє питання. Він отримає велику кількість результатів з різними відповідями і думками інших користувачів, якщо це, наприклад, якесь творче питання. Існує велика кількість веб-додатків, які дозволяють користувачу створити статтю чи відповідь на своє питання і розмістити її в мережі для інших користувачів. При цьому не потрібно знати програмування чи виконувати якісь інші складні операції, достатньо вміти користуватися всього лише текстовим редактором. Опублікована стаття буде відображатися іншим користувачам, коли вони будуть здійснювати пошук за ключовими словами, які обрані для неї.

Разом із розвитком інтернету покращується мережеве покриття та зростає швидкість передачі даних, яку пропонують провайдери. В наш час майже в кожному куточку світу можна отримати доступ до інтернету через мобільний телефон чи інший пристрій. Завдяки цьому користувачі мають постійний доступ до інформації та мають можливість переглядати відео на телефоні в хорошій якості. Важливою особливістю є те, що в ми не обмежені інформацією тільки країни, в якій ми живемо, ми легко можемо отримати доступ до даних інших країн.

Інтернет надає необмежені можливості користувачам в усіх сферах життя. Однієї із таких важливих напрямів є власне навчання студентів, адже на навчання ми витрачаємо велику кількість часу. Існує велика кількість, як іноземних, так і українських веб-додатків, які дозволяють обрати певний напрям навчання та вивчати його. Як правило, курс складається з відео та текстових матеріалів. Вони поділені на секції. Студент може передивитися відео, в якому викладач розповідає про певну тему та демонструє це наглядно на відео. Після перегляду відео доступі текстові матеріали, де знову ж таки розповідається про дану тему.

Щоб завершити певну секцію студент повинен пройти тести, які дозволяють перевірити якісь засвоєних знань. Якщо студент успішно пройшов всі секції та здав тести, він отримує сертифікат про проходження даного курсу із оцінкою. Якщо студенту не сподобався даний курс та він передумав його проходити, він може просто від нього відмовитися. Таким чином, користувачі можуть обирати теми, які їх цікавлять та отримувати відповідні знання від передових вузів світу. В результаті проходження курсу вони отримують сертифікати із назвою курсу.

Навчання в українських вузах має зовсім інший характер. Ми також можемо обрати спеціальність, за якою бажаємо навчатися. Для вступу до вузу на певну спеціальність потрібно добре здати екзамени та отримати якомога більшу оцінку за них. Після вступу, як правило, ми вивчаємо велику кількість інших предметів, які взагалі не пов’язані зі спеціальність. Рівень викладання матеріалу не завжди на високому рівні. Ми маємо велику кількість дисциплін, які мають сучасну назву, але чомусь використовується матеріал, якому вже 10 років. Більшість вивченого матеріалу ми не можемо застосувати, адже він або застарів, аби викладався досить на базовому рівні. Студенти поступово втрачають мотивацію до навчання і вивчають предмети тільки для того, що їх здати. Тому постає нагальна потреба в зміні системи навчання. На мою думку, це можна вирішити завдяки навчанню через інтернет та отриманню корисних знань для влаштування на роботу та подальшого розвитку. Такі системи повинні мати належну швидкодію та бути зручними у користуванні.

В даній магістерській дисертації поставлено за мету оптимізувати та підвищити швидкодію системи для навчання студентів та керування навчальним процесом.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ І ПОСТАНОВКА ЗАДАЧІ

1.1. Об’єкт та предмет дослідження

Дослідження проводиться в області розробки та оптимізації веб-додатків. Існує велика кількість додатків для навчання студентів, але більшість з них мають суттєві недоліки пов’язані із неналежною оптимізацією, перевантаженістю інформацією, незручністю в користуванні та іншими факторами. Постає потреба в розробці системи, яка б мала сучасний дизайн, була зручною в користуванні, забезпечувала можливість навчатися онлайн та мала належну швидкодію, що є одним із найголовніших параметрів системи. В системі для навчання студентів та керуванням навчальним процесом розміщується велика кількість відео та текстових матеріалів. Відповідно при завантаженні певної сторінки все це займає певний час і якщо цей час буде занадто великий, користуватися такою системи буде не зручно. Потрібно враховувати, що кількість студентів, які одночасно відвідали певну сторінку може бути великою і відповідно це буде спричиняти велику кількість запитів до серверу. Така система повинна бути належним чином протестована та оптимізована для зручного користування нею.

Об’єкт дослідження – процес роботи веб-додатку для навчання студентів та керування навчальним процесом та час виконання запитів користувачів.

Предмет дослідження – методи та технології оптимізації роботи веб-додатку, які б забезпечили його належну швидкодію.

1.2. Аналіз існуючих систем

В нас час існує велика кількість систем для навчання студентів, але більшість з них мають певні недоліки. Деякі системи досить довго завантажуються, що призводить до того, що ними незручно користуватися, оскільки доводиться чекати декілька хвилин, поки завантажиться певний матеріал чи відео. Певні системи мають застарілий дизайн, що в свою чергу відштовхує від користування такими системами. Інші, навпаки мають сучасний

дизайн, але відображають відразу занадто велику кількість інформації, що ускладнює пошук потрібного курсу чи напряму в таких системах.

Однієї із найвідоміших системи для навчання є ITVDN, зображена на рис. 1.1

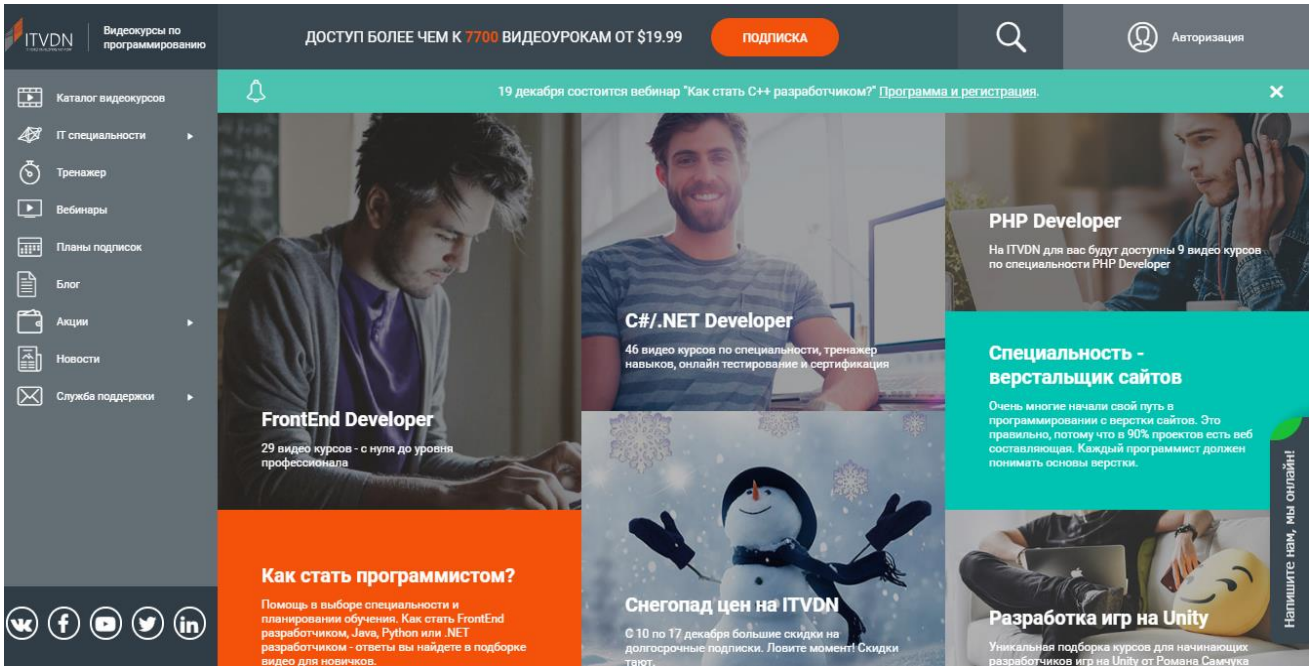

Рисунок 1.1. Система для навчання ITVDN

Можна помітити, що дана веб-орієнтована система має сучасний дизайн. У верхньому лівому краю знаходиться логотип. Після нього розміщується повідомлення з кількістю відео уроків, які розміщуються в системі. Користувачі можуть здійснити пошук, натиснувши на іконку пошуку та зареєструватися або ввійти в свій особистий кабінет. Зліва розміщується меню, яке дозволяє користувачам отримати інформацію про наявні курси, переглянути каталог відео уроків та багато інших можливостей. При натисненні на пункт “IT спеціальності”, користувачі отримують можливість переглянути найпопулярніші спеціальності або ж натиснути на кнопку “Всі спеціальності” та отримати список всіх існуючих спеціальностей. На головній сторінці у вигляді прямокутників розміщені найпопулярніші професії в наш час, що дозволяє користувачам відразу отримати інформацію про потрібну їм спеціальність. Система має велику кількість динамічних елементів і при наведенні на певний блок, він підсвічується або збільшується в розмірі, що є досить зручним, оскільки

можна відразу зрозуміти, на якому блоці ти знаходишся і відповідно при натисканні, яка інформація тобі буде надана. Головна сторінка має велику висоту та дозволяє користувачу перелистувати інформацію(рис. 1.2).

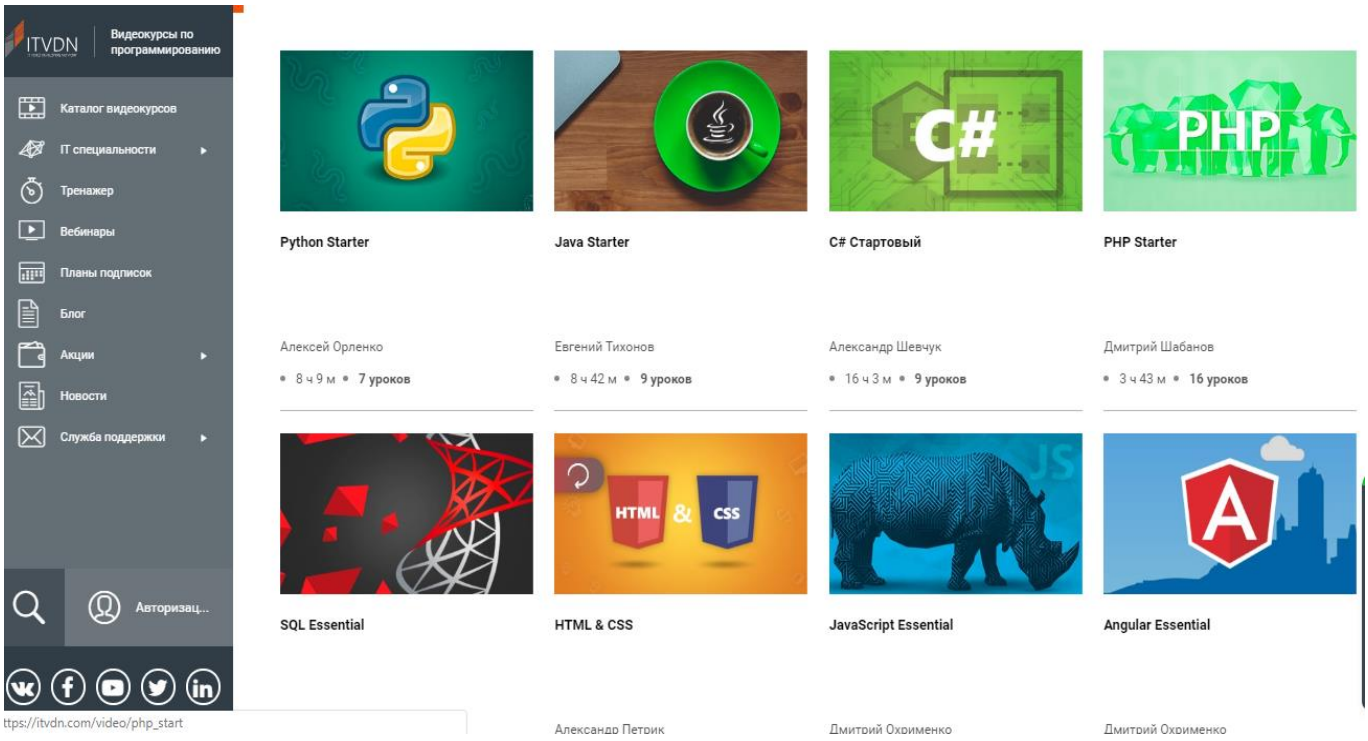

Рисунок 1.2. Інформація головної сторінки системи ITVDN

На мою думку, система має дещо перевантажений дизайн, оскільки користувач відразу отримує велику кількість інформації і тяжко зорієнтуватися, куди потрібно натискати для отримання того, що він хотів. Наприклад, на зображеннях можна помітити, що текст всіх елементів системи написаний російською мовою і якщо я хочу переключити мову системи на українську, то мені не зрозуміло, де я можу виконати цю дію. Доводиться передивлятися всі елементи системи, які відображаються для мене та шукати потрібну мені опцію. Це займає велику кількість часу та призводить до того, що користувачі перестають користуватися такою системою.

Після проходження реєстрації, користувач перенаправляється знову на головну сторінку, хоча, на мою думку, логічніше було б відобразити йому особистий кабінет. Особистий кабінет має, як на мене, не досить зручне розміщення елементів та є перевантажений інформацією(рис 1.3)

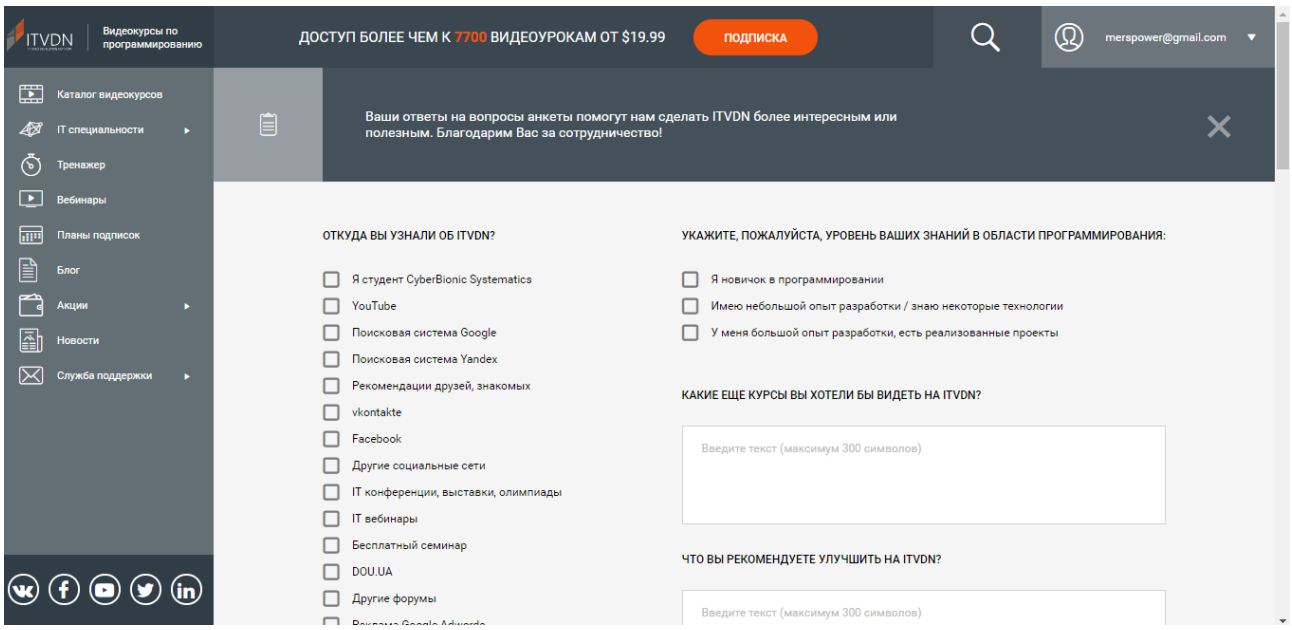

Рисунок 1.3. Особистий кабінет користувача системи ITVDN

Користувачу відразу відображається велика кількість питань та велика кількість інформації. Не зрозуміло, де я можу подивитися свої куплені відео уроки, курси, які я вже пройшов та іншу інформацію, яка, на мою думку, є досить важливою для користувачів та повинна відображатися в доцільному місці, де б це було інтуїтивно зрозуміло. Меню в особистому кабінеті та на головні сторінці залишається завжди незмінним, хоча там було доцільно відображати пункти для перегляну певної особистої інформації користувача після його авторизації. При пролистуванні сторінки особистого кабінету, основна інформація користувача знаходиться аж майже внизу сторінки. При цьому ми отримуємо подвійне меню, зображене на рис 1.4.

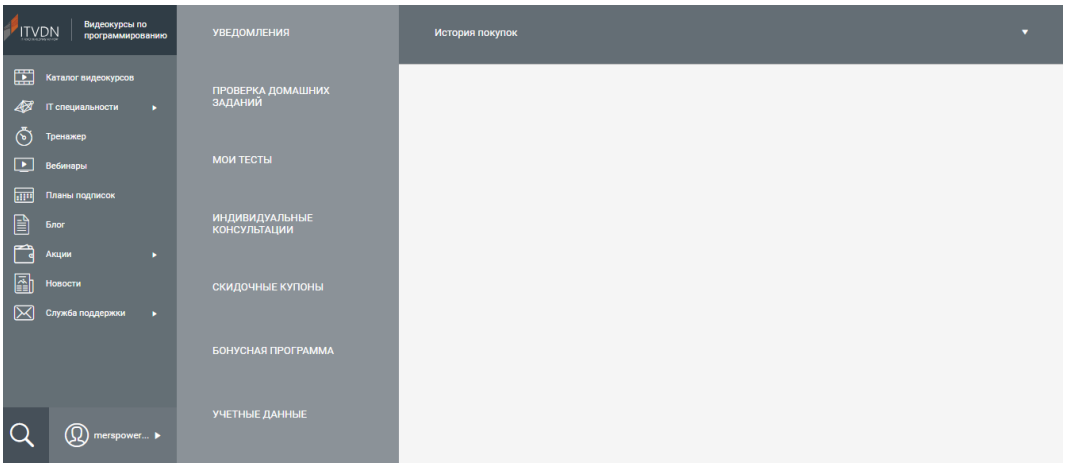

Рисунок 1.4. Меню із особистою інформацією користувача

Відображення користувачу великої кількості інформації призводить до сповільнення роботи системи(рис 1.5)

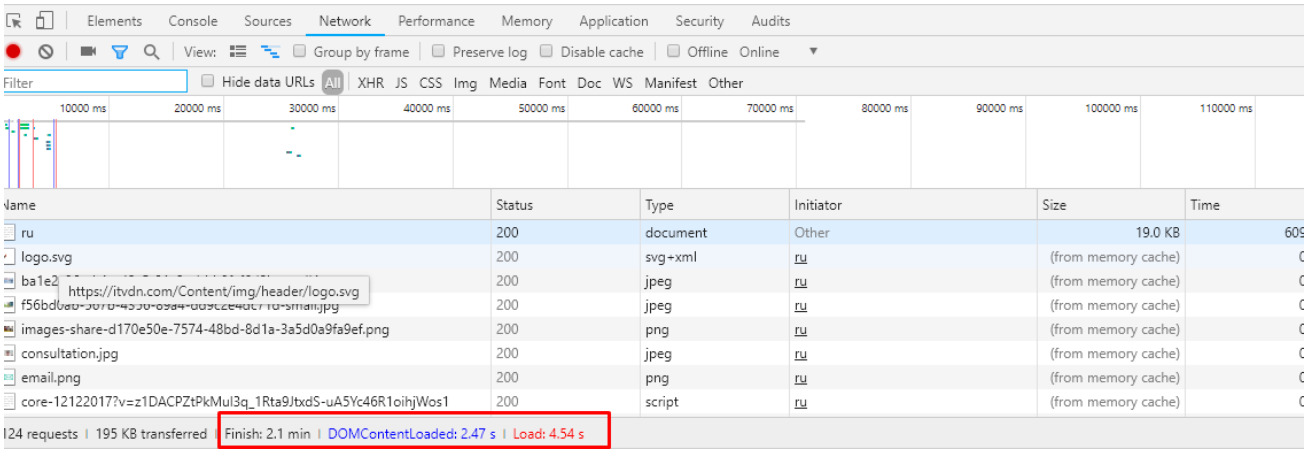

Рисунок 1.5. Час завантаження головної сторінки ITVDN

Можна помітити, що головна сторінка завантажується 5 секунд. На головній сторінці відображається лише текстовий матеріал без відео уроків та інших елементів, які потребують здійснення ще більших зусиль для відображення їх користувачу. Загальний час завантаження всього контенту сторінки становить 2 хвилини, що також є не найкращим показником. При завантаженні системи та відкритті консолі розробника, можна помітити, що відображаються деякі помилки пов'язані із завантаженням javascript скриптів для взаємодії з користувачем. Будь-які помилки в системі є причиною довгого завантаження сторінки та є її великим недоліком.

Все це не є досить незручним у користуванні та призводить до того, що для пошуку потрібної інформації користувачу доводиться просто переходити по різним сторінкам та перелистувати їх, що займає досить багато часу. Занадто велике нагромадження інформації на кожній сторінці даної системи є її значним недоліком. Процес реєстрації та авторизації налаштовано незручним способом, зазвичай після авторизації користувач переходить в свій особистий кабінет, де бажає переглянути куплені курси та іншу інформацію, в даній системі користувач знову перенаправляється на головну сторінку системи. Що для особистого кабінету, що для головної та інших сторінок, ми маємо одне і теж меню зліва сторінки, хоча його доцільність використання в особистому кабінеті незрозуміла, особливо при відображенні ще одного меню для перегляду

особистої інформації користувача. Всі ці недоліки системи призводять до того, що нею незручно користуватися та потрібно витратити досить багато часу для того, щоб запам’ятати, де можна знайти ту чи іншу інформацію.

Однією із систем для навчання є Main Academy(рис. 1.6)

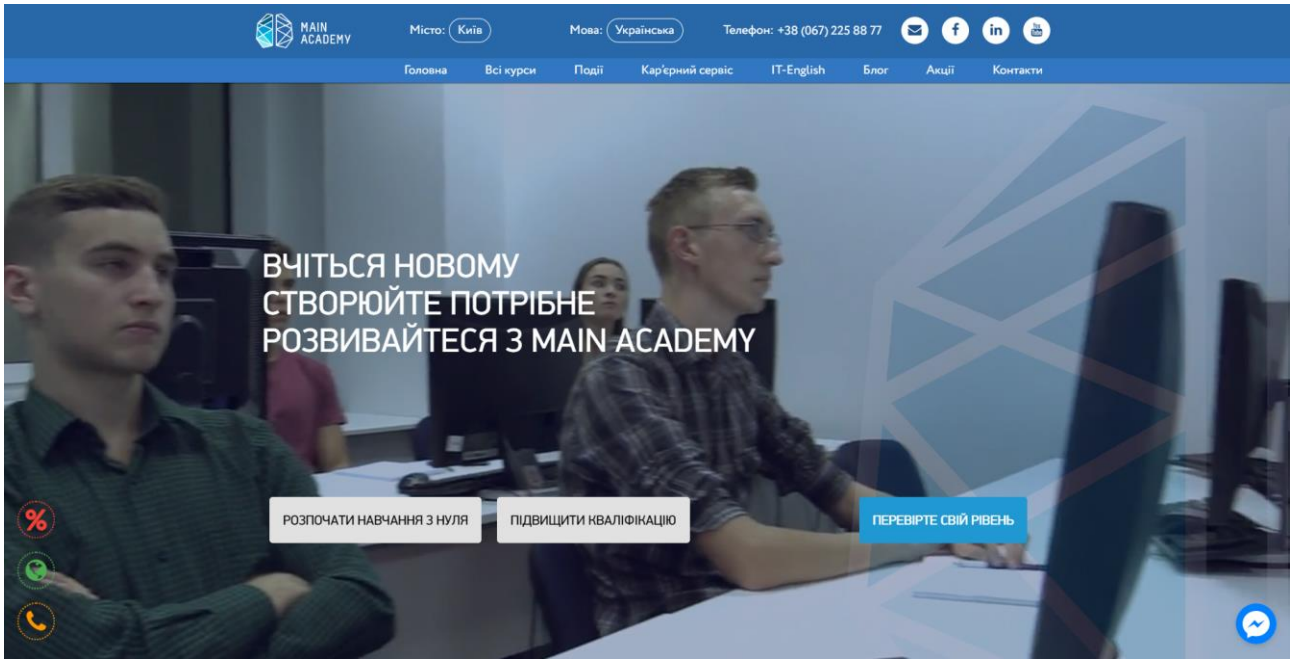

Рисунок 1.6. Головна сторінка системи для навчання Main Academy

На головній сторінці даної системи ми можемо помітити логотип, місто, в якому ми бажаємо переглянути доступні курси, можемо обрати основну мову системи, на якій буде відображатися вся інформація. Це є значною перевагою даної системи, оскільки важлива інформація розміщується на потрібних місцях, дизайн побудований таким чином, що вона добре виділяється серед інших елементів та розташовується в зручному місці. Користувач може відразу обрати потрібне собі місто та мову системи.

Меню системи розміщується під основними налаштуваннями, але при натисненні на певний пункт відбувається пролистування сторінки та відображення відповідних даних.

При пролистуванні головної сторінки ми знову бачимо занадто велику кількість інформації, що знову ж таки впливає на роботу системи та збільшує час завантаження головної сторінки. Відкривши консоль розробника, можемо помітити, що час завантаження сторінки є також досить довгим(рис 1.7)

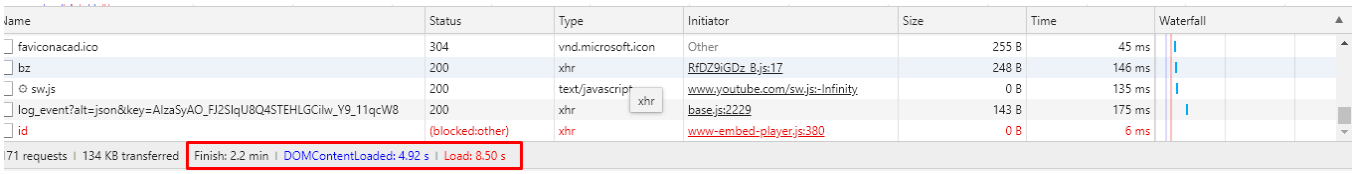

Рисунок 1.7. Час завантаження сторінки системи для навчання Main Academy

Відразу під меню даної системи розміщується відео, яка загалом не несе жодної важливої інформації користувачу. Справа та зліва розміщуються різні іконки системи, кольори яких зливаються із загальним дизайном системи і це є досить незручним.

В даній системі відсутня можливість зареєструватися та авторизуватися. Користувач не може зайти у свій особистий кабінет, переглянути певний курс чи іншу інформацію про себе. Відсутня можливість перегляду відео та текстового матеріалу для того чи іншого курсу.

Дана система має досить обмежений функціонал та не дозволяє користувачу повноцінно навчатися, використовуючи відео та текстовий матеріал. Основна її ціль – це надати можливість користувачам записатися на певний курс чи напрям, але при цьому все навчання буде здійснюватися у навчальному класі із вчителем.

1.3. Постановка задачі

Метою даної магістерської дисертації є оптимізація та підвищення швидкодії системи для навчання студентів та керування навчальним процесом.

Для досягнення даної мети необхідно вирішити наступні задачі:

– проаналізувати аналоги, визначити їх основні переваги та недоліки, а також причини, які сповільнюють їх роботу.

– на основі проведеного аналізу визначити загальну концепцію системи та її основні модулі;

– створити структуру бази даних із врахуванням найчастіше виконуваних запитів користувачів;

– створити моделі для роботи з базою даних та оптимізувати запити до неї;

– написати RESTful API та логіку роботи основних модулів системи;

– написати правила для валідації всіх даних, які ми отримуємо в системі;

– написати тести для автоматичного тестування системи;

– впровадити систему кешування Redis для підвищення швидкодії системи;

– розробити механізм виконання складних задач за допомогою утиліти cron у фоновому режимі;

– забезпечити масштабування системи;

Висновки до розділу

В даному розділі було розглянуто область, в якій проводиться дослідження. Проведено аналіз існуючих системи для навчання студентів та керування навчальним процесу. Однією із таких системи є ITVDN. Дана система має сучасний дизайн, але кожна її сторінка є перевантаженою різною інформацією. Для знаходження потрібного пункту чи розділу потрібно витратити значну кількість часу, адже головна та важлива інформація не виділена належним чином. В даній системі неправильно побудована логіка авторизації користувачів. Після авторизації або реєстрації користувача перенаправляє на головну сторінку, хоча, якщо користувач хотів ввійти в систему, то напевно йому потрібний особистий кабінет, а не головна сторінка, яка завжди відображається всім користувачам. Особистий кабінет теж перевантажений зайвою інформацією. Якщо користувач зайшов у свій кабінет, то він хоче переглянути свою особисту інформацію або дані, про пройдені чи поточні курси. Першим, що він бачить – це велика кількість запитань, на які йому пропонують дати відповідь. Інша проаналізована система для навчання студентів має досить обмежений функціонал та взагалі не дозволяє користувачам навчатися онлайн. Враховуючи всі проаналізована існуючі системи було визначено мету, предмет та об’єкт дослідження. Визначено основні задачі, які потрібно виконати для досягнення даної мети.

РОЗДІЛ 2. ВИБІР ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ СИСТЕМИ ДЛЯ НАВЧАННЯ СТУДЕНТІВ ТА КЕРУВАННЯ НАВЧАЛЬНИМ ПРОЦЕСОМ

Оскільки ми плануємо розробляти та здійснювати оптимізацію веб-орієнтовану систему для навчання студентів та керування навчальним процесом, доцільним буде використання для клієнтської частини мови розмітки HTML, каскадних таблиць стилів CSS, фреймворку Bootstrap та мови програмування JavaScript. Дані технології завжди використовуються для клієнтської частини, оскільки вони не мають аналогів. Фреймворк Bootstrap має аналоги, але він є досить поширеним в наш час, має добре описану та повноцінну документацію, велику кількість публікацій, є досить простим та зручним у використанні. Для розробки бекенд частини будемо використовувати мову програмування PHP, оскільки вона є найбільш популярною при розробці веб-додатків, має повноцінну документацію, велику кількість готових рішень та фреймворків, які мають добре продуману структуру, забезпечують стандартизацію при розробці веб-додатків, надають можливість досить просто додавати новий функціонал та окремі компоненти, а також забезпечують належну безпеку веб-додатку. За основну для розробки даної системи будемо використовувати фреймворк Laravel. Він є одним із найпопулярніших фреймворків в наш час, завдяки своїй простоті у використанні та добре продуманій структурі. В якості бази даних будемо використовувати MySQL.

Розглянемо основні особливості та переваги обраних технологій.

2.1. HTML, CSS та фреймворк Bootstrap

Розпочнемо розгляд технологій з мови HTML.

HTML – це мова розмітки веб-сторінок у мережі Інтернет. Абревіатура HTML розшифровується як Hyper Text Markup Language. HTML не є мовою програмування, оскільки за допомогою неї не можна написати програму, яка б виконувала певний алгоритм або здійснювала розрахунки.

В наш час всі веб-сторінки створюються за допомогою HTML. Дана мова розмітки дозволяє створювати структуровані документи, в яких розміщується

текст з виділенням заголовків та абзаців, таблиці, цитати, нумеровані та марковані списки. В даному документі можна розташовувати гіперпосилання на інші веб-сторінки, створювати інтерактивні форми, розміщувати зображення та вставляти відео для перегляду. Документ даного типу передається за допомогою протоколу HTTP або HTTPS. Потім він інтерпритується браузером та відображається для користувача у звичайному вигляді.

Мова розмітки HTML складається с чотирьох основних компонентів:

- елементів;
- символьних мнемонік;
- декларації типу документу.

Декларація типу документу прописується спочатку документу та визначає тип документ та версію HTML, яка буде використовуватися для його розмітки[1].

Елемент є головним компонентом мови HTML. Він починається з початкового тегу, що має вигляд <*element-name*> та закінчується кінцевим тегом, який має схожий вигляд </*element-name*> (на початку елемента додається скісна риска). Між початковим та кінцевим тегом розміщується певна інформація. Атрибути прописуються в початковому тезі та задають певні характеристики для тексту. Існують елементи, які мають тільки початковий тег. За допомогою цих елементів, як правило, здійснюється форматування тексту, наприклад, перенесення речення на новий рядок.

Символьні мнемоніки – це набір певний сполучень символів, які дозволяють вставляти в HTML документ специфічні позначки, наприклад, символи «”» та «&». Вони досить часто використовуються при створенні документів[3].

В наш час використовується версія HTML 5.0. Кожний документ наповнюється відповідно до того, як він повинен бути відображений користувачу, але при цьому всі документи мають одну базову структуру (рис. 2.1).

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>

</body>
</html>
```

Рисунок 2.1. Базова структура HTML документа

Каскадні таблиці стилів або CSS – це мова стилів, яка визначає відображення сторінок, що написані мовами розмітки даних. За допомогою даної мови можна задати певний шрифт та колір для сторінки або певного блоку, встановити потрібний відступ для блоку та задати його ширину і висоту, встановлювати позиціонування для певних блоків, задавати фонове зображення для сторінки або ж блоку. Каскадні таблиці стилів мають великий набір правил для виконання різних дій над сторінкою[4].

Дана мова стилів має нескладний синтаксис. Кожне правило складається з трьох основних елементів: селектора, властивості та значення властивості. Селектор – це HTML тег або клас (атрибут тега class), для якого ми хочемо здійснити певні дії, наприклад, вирівняти його по правому краю. Під властивістю ми розуміємо назву дії, яку ми хочемо виконати над блоком. Каскадні таблиці мають власний набір дій, кожна з який має своє призначення. І, як зрозуміло з назви, значення властивості – це кількісне або одне із зарезервованих значень властивості, яку ми хочемо застосувати до обраного блоку.

CSS може також містити коментарі. Синтаксис коментарів подібний до синтаксису, що використовується в багатьох мовах програмування (наприклад C, PHP). Слід звернути увагу на те, щоб в коментарях CSS-файлів не використовувалися кирилічні літери, оскільки деякі браузери можуть некоректно обробляти CSS-файли з українськими або російськими літерами.

В якості прикладу, розглянемо наступне правило (рис. 2.2).

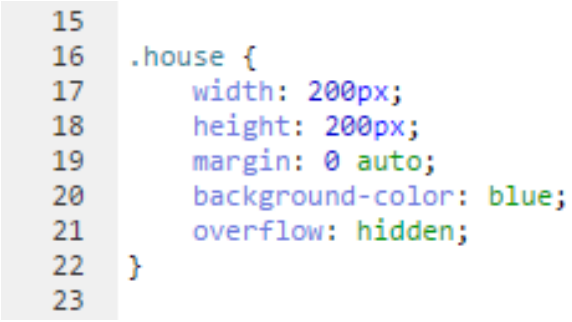

Рисунок 2.2. Задання певного правила для блоку

На наведеному рисунку в якості селектора в нас виступає клас *.house*, який заданий певному тегу. Властивості *width*, яка задає ширину блоку, задане значення *200px*. Аналогічно прописуються і інші властивості.

Bootstrap – це один із найбільш популярних в наш час HTML, CSS та JavaScript фреймворків, що дозволяє значно спростити розробку та пришвидшити час, який необхідний для створення веб-додатку. На офіційному сайті є досить добре описана документація з прикладами створення різних елементів. При скачуванні користувач отримує звичайні та мінімізовані файли CSS та JavaScript. Використовуючи заздалегідь відомі класи та інші атрибути для HTML тегів, можна легко створити адаптивний сайт.

При розробці адаптивних веб-додатків ідеологія Bootstrap полягає в тому, що веб-сторінка ділить на 12 стовпців, ширина яких залежить від розміру сторінки. Далі ми кожному тегу, задаємо кількість стовпців, які він буде займати в залежності від розміру екрану. Таким чином, задаючи певні класи тегам ми отримуємо повністю адаптивний сайт[7].

2.2. JavaScript

JavaScript – це об’єктно-орієнтована мова програмування, яка є реалізацією мови ECMAScript. Вона широко використовується для створення динамічних елементів на сайті, зміни певних елементів та структури веб-сторінки, асинхронного обміну даними з сервером та виконання інших дій в браузері.

Структурно JavaScript можна представити у вигляді об’єднання трьох частин: ядра(ECMAScript), об’єктної моделі браузера та об’єктної моделі документа. Як зрозуміло з назви, ядро становить основу даної мови

програмування. В ньому визначені основні типи даних, ключові та зарезервовані слова, конструкції, оператори, об’єкти та інші основні компоненти даної мови. Об’єктна модель браузера є проміжним шаром між ядром та об’єктною моделлю документа. Вона дозволяє здійснювати вплив на браузер, а саме: здійснювати зациклювання коду, виконувати затримку у виконанні коду, наприклад, для очікування певних дій від користувача, здійснювати переадресацію користувача на вказану url адресу, отримувати інформацію про розмір вікна браузера та багато іншого. Об’єктна модель документа є складовою частиною JavaScript та дозволяє виконувати маніпуляцію над HTML сторінкою або сторінкою, що створена за допомогою мови розмітки. Вона дозволяє змінювати структуру певного блоку, видаляти його вміст, копіювати та вставляти в інше місце. Завдяки такій складовій структурі дана мова програмування дозволяє виконувати велику кількість задач у браузері.

JavaScript має синтаксис схожий на синтаксис мов програмування C та Java. В даній мові назва змінної може починатися з букви, символа долара, підкреслення; для оформлення однорядкових коментарів використовуються //, а для багаторядкових /**/. До її особливостей відноситься динамічна типізація, яка дає змогу оголошувати змінні з одним типом даних, а потім записувати в них дані іншого типу, автоматичне управління пам’яттю та прототипне наслідування[5].

2.3. PHP та фреймворк Laravel

PHP – це скриптова мова програмування, що в нас час широко використовується для розробки веб-сайтів. Близько 80% сайтів написані за допомогою цієї мови. Її головною особливістю є те, що виконується лише потрібний в даний момент файл. Тобто, якщо розробник допустить помилку в якось іншому файлі, то сайт продовжить успішно працювати, поки не буде виконаний запит до файлу, в якому допущена помилка. Ця особливість є досить важливою і відрізняє дану мову від таких мов програмування, як Java та C#, в яких для роботи веб-сайта потрібно зібратися всі бібліотеки та файли, їх

скомпілювати і тільки після цього все буде працювати. PHP на відміну від наведених інших мов має велике розмаїття фреймворків та систем керування контентом, що дозволяє розробникам вибирати різні рішення для виконання певних задач.

Синтаксис PHP подібний синтаксису C. Для виконання простої програми достатньо відкрити тег даної мови програмування. PHP підтримував чотири види тегів, але з виходом нової версії залишилася підтримка лише двох видів. Розпочати програму можна за допомогою тегів $<?php?>$ або коротких тегів <??>, але використовувати останні не рекомендується, так як вони не підтримується багатьма хостингами. Інструкції в даній мові програмування закінчуються символом ;. Назви змінних починаються з символу $. Велике значення в php приділяється асоціативним масивам. За допомогою них здійснюється передача великої кількості даних. Більшість фреймворків використовують їх для задання певних параметрів та початкових налаштувань різних бібліотек, що до них входять. Масиви в даній мові програмування є динамічними, тобто початково не потрібно вказувати його розмір. Саме в нових версіях даної мови велике значення приділялося їх оптимізації для підвищення швидкодії роботи даної мови.

PHP є динамічно типізованою мовою. В ній підтримується вісім типів даних. З них чотири простих типи - $integer, double, boollean$ та $string$, два складних $array$ та $object$ та два спеціальних типи - $resource$ та $null$. В останні версії PHP з'явилось правило, за допомогою якого можна включити строгу типізацію даних у файлі або всьому проекті[2].

Дана мова є досить простою в освоєнні, оскільки для того, щоб написати свою першу програму достатньо сервера та інтерпритатора PHP. Після цього створюється файл із розширенням .php, в ньому відкриваються описані вище теги та пишеться програма. Ось простий приклад програми, яка здійснює вивід в HTML розмітку певного тексту(рис. 2.3).

```
1
2     <!DOCTYPE html>
3     <html lang="en">
4     <head>
5         <meta charset="UTF-8">
6         <title>Title</title>
7     </head>
8     <body>
9     <?php  echo "Привет мир!"; ?>
10    </body>
11    </html>
```

Рисунок 2.3. Приклад реалізації виводу інформації на PHP

Зрозуміло, що це лише простий приклад реалізації виведення інформації за допомогою PHP. Насправді системи управління контентом та фреймворки мають складну архітектуру і розробляти новий функціонал, використовуючи їх, не так просто.

Для розробки системи для навчання студентів та керування навчальним процесом використовувався фреймворк Laravel. В основу даного фреймворку покладений принцип MVC, який полягає в поділі додатку на три основні частини: модель, контролер та відображення.

В моделях описується структура таблиць в базі даних, методи для отримання даних з бази даних та встановлення зв’язків між різними таблицями. При виконання запиту до бази даних ми отримуємо об’єкт або масив об’єктів, у властивостях яких зберігаються значення даних. Обмін даними через об’єкти здійснюється за допомогою ORM Eloqument. Саме вона є проміжним прошарком між базою даних та програмним кодом і здійснює обмін даними у зручному для нас вигляді. Контролер відповідає за логіку роботи програми при певному запиті користувача. В ньому розміщуються виклики методів в заданому порядку та здійснюється обмін даними між користувачем та нашою програмою. Користувачу із контролера повертається або масив даних, або ж певна частина HTML верстки із новими даними. Для відображення HTML верстки та даних, які динамічно підставляються у необхідні блоки, в даному фреймворці використовується шаблонізатор blade, який має деякі елементи синтаксису, подібного до php.

Фреймворк Laravel побудований на модульній основі, яка дозволяє скачувати та використовувати в ньому різні бібліотеки. Завантаження бібліотек та самого фреймворку здійснюється за допомогою менеджера залежностей, в якому для скачування необхідного компонента достатньо прописати його назву та версію і виконати команду оновлення[10].

2.4. MySQL

MySQL – це система управління реляційними базами даних. Вона має відкритий програмний код та використовує мову SQL(Structured Query Language) для виконання запитів та інших дій. Важливою перевагою даної системи є те, що вона працює практично на всіх платформах, включаючи Linux, Unix та Windows. Найчастіше вона використовується у веб-додатках, завдяки своїй швидкій роботі.

MySQL базується на моделі клієнт-сервер. Ядром цієї системи є сервер MySQL, який обробляє всі інструкції бази даних. Він доступний як окрема програма для використання в мережевому середовищі клієнт-сервер, так і як бібліотека, яку можна вбудувати в окремі конкретні додатки. Дана система була розроблена для швидкої обробки великих баз даних, тому вона і набула значної популярності у веб-додатках, оскільки вони забезпечує їх належну швидкодію.

MySQL дозволяє створювати два типи таблиць InnoDB та MarinaDB. Основна відмінність між цими типами таблиць полягає в тому, що InnoDB підтримує зовнішні ключі між таблицями, а MyISAM – ні. Зовнішні ключі допомагають забезпечити цілісність бази даних та не дозволяють записати дані, якщо вони порушують певний зв’язок. Перед кожним записом даних відбувається перевірка всіх обмежень та зв’язків і, якщо нічого не порушується, відбувається додавання даних. В MyISAM відсутні зовнішні зв’язки, тому вона використовується у досить складних та навантажених додатках, для яких є важливою висока швидкодія. Відсутність зв’язків спричиняє пришвидшення виконання запитів до бази даних, оскільки в такому випадку відсутні перевірки можливих зв’язків та обмежень перед записом та оновленням даних. Це

призводить до значного пришвидшення роботи з базою даних, але недоліком є те, що в такому випадку, у базі даних можуть зберігатися некоректні дані або може відбутися дублювання даних, тому доцільність застосування такого типу таблиць є досить спірним рішенням.

Система управління базами даних MySQL надає змогу зберігати та отримувати доступ до даних із використанням різних драйверів, які виконують операції над даними. Дана система дозволяє виконувати додаткове копіювання даних та розподіляти їх по різним таблицям. Така необхідність часто виникає в досить складних та навантажених системах, які оброблюють велику кількість даних. В таких системах виникають проблеми, пов’язані з тим, що певна таблиця починає містити занадто велику кількість даних і виникає необхідність вилучити певні дані з цієї таблиці та розподілити по декількох таблицях для пришвидшення пошуку. MySQL має певні інструменти та команди, які дозволяють виконати операції з розподілу даних.

Важливою особливістю MySQL є те, що вона не потребує від користувачів вивчення додаткових мов для виконання запитів. Для користування цією системою та виконання запитів до бази даних, вони можуть використовувати стандартні команди SQL.

Висновки до розділу

В даному розділі були розглянуті технології, що використовувались для розробки системи для навчання студентів та керування навчальним процесом. Так, HTML, CSS та Bootstrap будемо використовувати для створення клієнтської частини. HTML – мова розмітки, основними компонентами якої є елементи, символьні мнемоніки та декларації типу документа. CSS – це мова стилів, яка складається з правил. Кожне правило містить три основні компоненти: селектор, властивість та її значення. Фреймворк Bootstrap включає в себе набір стилів та коду на мові JavaScript, який дозволяє надавати елементам HTML сторінки динамічності, а також обмінюватися з сервером даними без перезавантаження сторінки. PHP використовується для написання скриптів обміну даними між

базою даних та клієнтською частину. Фреймворк Laravel має добре продуману структуру та набір класів, що спрощуються розробку та забезпечують створення якісної системи. Він побудований на модульній основі, яка дозволяє скачувати та використовувати в ньому різні бібліотеки. Завантаження бібліотек та самого фреймворку здійснюється за допомогою менеджера залежностей, який є досить простий у використанні. В якості бази даних будемо використовувати MySQL. Вона являє собою систему управління реляційними базами даних. Має відкритий програмний код та використовує мову SQL(Structured Query Language) для виконання запитів та інших дій. Важливою перевагою даної системи є те, що вона працює практично на всіх платформах, включаючи Linux, Unix та Windows.

РОЗДІЛ 3. РОЗРОБКА СИСТЕМИ ДЛЯ НАВЧАННЯ СТУДЕНТІВ ТА КЕРУВАННЯ НАВЧАЛЬНИМ ПРОЦЕСОМ

3.1. Архітектура системи

Система для навчання студентів та керування навчальним процесом складається з 8 модулів , які забезпечують її роботу(рис 3.1).

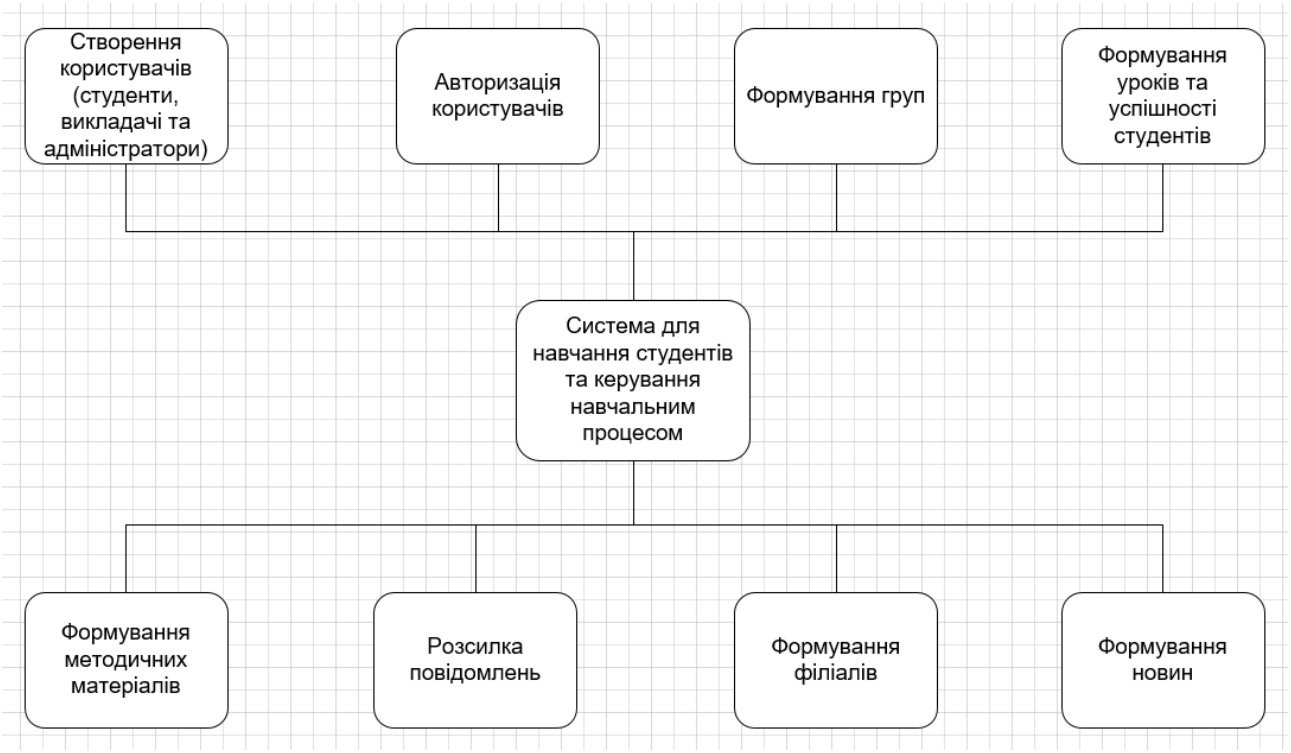

Рисунок 3.1. Архітектура системи для навчання студентів

Розглянемо модуль авторизації та реєстрації користувачів. Даний модуль відповідно забезпечує додавання нових користувачів у систему, відновлення почтової скриньки користувачів та їх авторизацію в системі. Для входу в систему потрібно ввести пароль та почтову скриньку користувача. Є можливість обрати пункт “Запам’ятати мене”. В цьому випадку система, крім надання токену та створення сесії для поточного користувача, ще й здійснить формування спеціальних даних в файлах браузера, які будуть залишати користувача авторизованим в системі навіть після того, як він закрив браузер та знову його відкрив. Якщо користувач ввів правильну почтову скриньку та пароль, його буде автоматично перенаправлено на сторінку новин системи. В разі введення некоректної почтової скриньки або паролю, користувач отримає повідомлення

про помилку та прохання перевірити введені дані ще раз. Також користувачу надається можливість змінити свій пароль. Оскільки паролі завжди зберігаються в базі даних в закодованому вигляді з використанням алгоритмів, які не дозволяють його розкодувати або ж ця операція є досить складною та трудомісткою, тому його відновити досить складно. Для зміни паролю користувачу необхідно ввести свою почтову адресу. Якщо така адреса існує в системі, відбувається відправлення повідомлення на дану адресу із посиланням на введення нового паролю. Після переходу за посиланням, користувачу необхідно ввести новий пароль та ще раз ввести в інше поле новий пароль для того, щоб переконатися, що він справді ввів коректний пароль і всі символи співпадають. Введені паролі порівнюються системою і в тому разі, коли вони однакові, відбувається зміна паролю для користувача, його кодування та збереження до бази даних в закодованому вигляді. Реєстрація користувачів відбувається адміністратором системи. Користувачі не можу самі зареєструватися в системі.

Після того, як користувач ввійшов в систему, першим, що він бачить – це новини системи, за які відповідає відповідно модуль новин. На головній сторінці системи відображаються чотири новини, які розміщені у вигляді прямокутних плиток по дві в рядоку. В першій плитці, як правило, розташовується привітання іменинників або привітання переможців певного конкурсу. У всіх трьох інших плитках відображаються новини. Кожна плитка складається з назви новини, картинки, яка відповідає даній новині, короткого опису та власне посилання на саму новину. Редагувати новини може тільки адміністратор системи.

Одним із основних модулів є модуль для роботи з користувачами. Даний модуль виконує велику кількість функцій. Адміністратор має доступ до двох вкладок в адміністраторській частині даної системи – “Студенти” та “Викладачі”. При переході на вкладку “Студенти” адміністратор отримує доступ до всіх студентів даної системи. Він може додати нового студента в систему. Для цього потрібно натиснути на кнопку додати та вказати ім’я, прізвище, почтову адресу, рік народження та інші дані студента. Після цього студент буде доданий

в систему. Пароль для даного студента буде створений автоматично. Таким студент може авторизуватися в системі, ввівши свою поштову адресу та створений пароль. В системі для навчання та керування навчальним процесом існує 4 типи користувачів, які навчаються – абітурієнт, студент, випускник та спеціаліст. Абітурієнт – це користувач, що записався на проходження певного курсу, який ще не розпочався. Даний користувач має обмежений доступ до навчальних матеріалів. Він може перечитати тільки вступні лекції по тим чи іншим предметам та переглянути вступне відео. Користувач даного типу може змінювати інформацію про себе та переглядати новини веб-додатку. Студент – це користувач, який проходить певний курс та якому надано доступ до навчальних матеріалів даного курсу. Він може повноцінно переглядати лекції та відеоматеріали з курсу, який він проходить. Даний користувач має доступ до файлів, які використовуються для даного курсу, наприклад, коду програми чи верстки певної сторінки. Випускник – користувач, який закінчив певний курс із позитивною відміткою. Він має доступ до всіх матеріалів курсу, який він закінчив. Користувач, який закінчив декілька курсів із позитивною відміткою отримує статус спеціаліст. Він отримує доступ до всіх матеріалів курсів, які він закінчив. Йому надається скидка на інші курси даного навчального закладу. Він має можливість безпосередньо в системі переглядати список можливих вакансій, відповідно до отриманих навичок. При створенні нового користувача, адміністратор обов’язково повинен обрати його тип із перерахованих вище. Адміністратор має можливість здійснювати фільтрацію студентів за напрямом навчання, курсом навчання та здійснювати безпосередній пошук студента за його прізвищем. Дані типи фільтрації допомагають швидко знайти потрібного студента. Відповідно на вкладці “Викладачі” адміністратор може додати викладачів для даної системи. Існують два основні можливі типи користувачів, які викладають певний матеріал – основний викладач та резервний викладач. Обидва користувачі мають однакові права доступу до матеріалів в системі. Вони можуть створювати текстові матеріали для свого курсу, додавати файли для кожного уроку, такі як: код програми, верстка сторінки, стилі сторінок, схема

структури бази даних, а також відеоматеріали. Вони мають повний доступ до даних матеріалів, можуть їх редагувати, надавати до них доступ студентам та видаляти їх в разі необхідності.

Кожний зареєстрований користувач має вкладку “Мій профіль”. Зайшовши на дану вкладку, користувач може переглянути інформацію про себе, таку як: ім’я, прізвище, дата народження, місто народження, адреса проживання та іншу інформацію, а також може її відредагувати та зберегти. На даній сторінці користувач має можливість виконати дві важливі операції – змінити свій пароль та змінити свою поштову адресу. Для зміни паролю користувачу необхідно ввести свій старий пароль, ввести новий пароль та ще раз ввести новий пароль у іншому полі для того, щоб переконатися, що він не помилився в певному символі при введенні нового пароля та ввів його правильно. Для зміни своєї поштової адресу користувачу необхідно ввести свою стару та нову поштові адреси.

Модуль для створення груп забезпечує створення, редагування та видалення груп. Даний модуль доступний тільки адміністратору сайту. Він може створити нову групу, вказавши її назву, початок занять, термін навчання студентів даної групи, місце проведення занять та додати потрібних студентів до даної групи. При виборі студентів адміністратору надається можливість додати будь-якого студента даної системи. Адміністратору виводиться ім’я та прізвище студентів. Якщо студента не існує в цій системі, потрібно спочатку на вкладці “Студенти” додати його і потім він автоматично буде відображатися на вкладці для створення груп. Для зручності роботи адміністратор може здійснювати фільтрацію груп за різними критеріями такими, як: напрям, за яким навчається група; статус групи, який може бути “поточним”, для групи, в якій зараз проходить навчання, “набір” – для групи, яка в майбутньому буде розпочинати навчання та завершений; дата початку навчання груп та дата завершення навчання певної групи.

Модуль філіалів дозволяє адміністратору сайту створювати, редагувати та видаляти філіали. Кожний навчальний заклад може мати декілька філіалів – місць для навчання, які територіально розташовані в різний районах або містах.

Для зручного керування та слідкування за студентами, кожна група прив’язується до філіалу, що дозволяє точно визначити, де саме відбувається навчання цієї групи. Кожний філіал містить фотографію його керівника, назву, ім’я та прізвище керівника, адресу, за якою він знаходиться та телефон для зв’язку з ним.

Одним із важливих та складних модулів є модуль методичних матеріалів. Він дозволяє відображати в браузері файлову систему певної директорії з файлами та користуватися ними для навчальних цілей. Так, викладач певного курсу після проходження першого уроку може додавати файли, які до нього відносяться та допоможуть студентам краще засвоїти матеріал або переглянути його. Це може бути код програми, блок-схема та всі інші необхідні документи для студентів. Викладач може налаштовувати права доступу до даних файлів, може дозволяти тільки перегляд файлів або відображення файлів тільки для студентів певного напряму навчання. Таким чином, викладач після кожної лекції чи уроку додає необхідні файли та налаштовує права доступу до них. Студенти ж після відвідування певного уроку отримують доступ до всіх документів та матеріалів, які були розглянуті на уроці. Домашнє завдання у вигляді якоїсь частини коду, яку потрібно виправити чи дописати, також розміщується в методичних матеріалах. Адміністратор відповідно має доступ до всіх файлів, які розміщені в методичних матеріалах. Додавання та редагування файлів здійснюється за допомогою зручного інтерфейсу через браузер користувача. Непотрібно підключатися до сайту через консоль або іншими способами і створювати папки та файли вручну. Адміністратор може переглядати вміст всіх файлів та папок в даній системі та переходити з директорії в іншу директорію. Даний модуль дозволяє викладати файли та документи для кожного курсу, формувати власну структуру директорій відповідно до проведених уроків із власними назвами папок, додавати потрібні файли та домашні завдання, а також регулювати доступ до доданих файлів та документів.

Після того як було створено певну групу, адміністратор системи та викладачі мають можливість створювати розклад уроків відповідно до

навчальних планів. Для цього потрібно перейти на вкладку системи “Успішність” та обрати групу. Після вибору групу буде відображено перелік уроків для даної групи. Якщо група ще не розпочала навчання, ми отримаємо пустий список. Адміністратор та викладачі системи мають можливість додавати уроки. Для додавання уроку достатньо натиснути на відповідну кнопку, ввести назву та обрати дату проведення уроку. Після того, як конкретний урок буде додано, він відобразиться в загальному списку уроків. При натисканні на конкретний урок буде відображено в модальному вікні його назву, дату проведення та список всіх студентів даної групи із можливістю проставлення кожному студенту оцінки за даний урок. В якості системи оцінювання використовується 5-бальна система. Оцінювання відбувається за наступною схемою:

– 0 балів – студент був відсутній на уроці;

– 1 бал – студент був присутні на уроці;

– 2-3 бали – студент був присутній на уроці та виконав домашнє завдання;

– 4-5 балів – студент був присутній на уроці, виконав домашнє завдання та був активний і відповідав на поставлені питання.

Відповідно до даної системи кожний студент отримує певну оцінку за кожний урок. Адміністратор та викладачі системи можуть змінювати оцінку в будь-який момент. Їм надається можливість переглядати всі оцінки студентів даної групи відповідно до кількості доданих уроків та сумарний бал кожного студента. Так, при натисненні на кнопку “Переглянути оцінки”, відображається весь перелік уроків та всі студенти, які належать до даної групи. Навпроти кожного студента та уроку відображається оцінка і в кінці створено поле, в якому відображається сумарний бал кожного студента.

Модуль для виконання розсилки повідомлень призначений для масового надсилання повідомлень певній групі студентів. Лише адміністратор системи має доступ до даного модуля. Для виконання ціле направленої розсилки повідомлень, створено набір фільтрів, які дозволяють відфільтрувати студентів за певними критеріями. Адміністратор має можливість здійснити фільтрацією

користувачів за напрямом навчання та певним курсом, якщо потрібно надіслати повідомлення студентам, які навчаються за відповідним напрямом чи курсом. Є можливість надіслати повідомлення користувачам з певним статусом, це можуть бути як студенти, так і викладачі. Адміністратор також може виконати фільтрацію за філіалом та певною групою. Якщо потрібно виконати надсилання повідомлень тільки певним студентам з різним напрямів, курсів, філіалів, є можливість обрати конкретних студентів, які повинні отримати повідомлення. Велика кількість параметрів для фільтрації, дозволяє виконати цілеспрямовану розсилку конкретним користувачам системи або ж всім користувачам, якщо не обрати жодного параметру.

3.2. Структура бази даних

База даних системи для навчання та керування навчальним процесом складається із 14 таблиць, кожна з яких відповідає за збереження певних даних.

Всі таблиці даної системи мають префікс lc. Це зроблено для того, щоб назви таблиць були унікальними. Це забезпечує можливість здійснювати інтеграцію цієї системи з іншими системами. Вам просто потрібно встановити дану систему в потрібне місце та виконати міграції і сіди за допомогою спеціальних команд Laravel. Міграції являють собою класи із описом полів, які потрібно створити, їх типом, унікальними та зовнішніми ключами(рис. 3.2).

```
public function up()
{
    if ( ! Schema::hasTable('users') ) {
        Schema::create( table: 'users', function (Blueprint $table) {
            $table->increments( column: 'id');
            $table->string( column: 'name',  length: 50)->nullable();
            $table->string( column: 'surname',  length: 150)->nullable();
            $table->string( column: 'email',  length: 100)->unique();
            $table->string( column: 'password',  length: 255);
            $table->string( column: 'telephone',  length: 50)->nullable();
            $table->enum( column: 'type', ['entrant', 'student', 'graduate','specialist','teacher_main','teacher_reserve','admin']);
            $table->integer( column: 'group_id')->unsigned()->nullable();
            $table->softDeletes();
            $table->rememberToken();
            $table->timestamps();
        });

        Schema::table( table: 'users', function (Blueprint $table) {
            $table->foreign( columns: 'group_id')->references('id')->on('groups');
        });
    }
}
```

Рисунок 3.2. Міграція для створення таблиці користувачів

Кількість міграцій(класів) відповідає кількості таблиць. Тобто кожній таблиці відповідає свій клас. Laravel надає можливість як запускати окремі міграції, так і виконувати послідовний запуск всіх міграцій. Оскільки міграції можуть містити зовнішні ключі, послідовність їх створення є досить важливою. При некоректному запуску міграцій ви можете отримати помилку про неможливість створення тієї чи іншої таблиці. При розробці міграцій потрібно продумати структура бази даних та почати їх створення із таблиць, які не мають зовнішніх ключів. Після створення міграцій для даних таблиць, потрібно послідовно проаналізувати інші таблиці та створювати міграції таким чином, щоб не виникало ситуацій, коли ми намагаємося створити таблицю, яка посилається на іншу таблицю, яка ще не створена. Тому потрібно уважно проаналізувати та продумати структура бази даних і правильно організувати створення міграцій.

Сіди використовуються для заповнення бази даних початковими даними (рис 3.3).

```php
public function run()
{
    DB::table('courses')->insert([
        'name' => 'Web design',
        'weight' => 1,
        'parent_id' => null
    ]);

    DB::table('courses')->insert([
        'name' => 'Frontend',
        'weight' => 4,
        'parent_id' => null
    ]);

    DB::table('courses')->insert([
        'name' => 'Development',
        'weight' => 7,
        'parent_id' => null
    ]);
```

Рисунок 3.3. Сіди для автоматичного створення курсів

Вони також являють собою окремі класи із методами, які повертають асоціативний масив даних, в якому назва ключа відповідає назві поля, а значення – значенню поля. Вони запускаються після створення таблиць бази даних та дозволяють наповнити її тестовими даними для проведення тестування того чи

іншого функціоналу. Потрібно враховувати послідовність запуску команд заповнення даними відповідно до послідовності створених міграцій.

3.2.1. Структура таблиць та зв'язків між ними

Система для навчання студентів та керування навчальним процесом має наступну структуру бази даних(рис. 3.4).

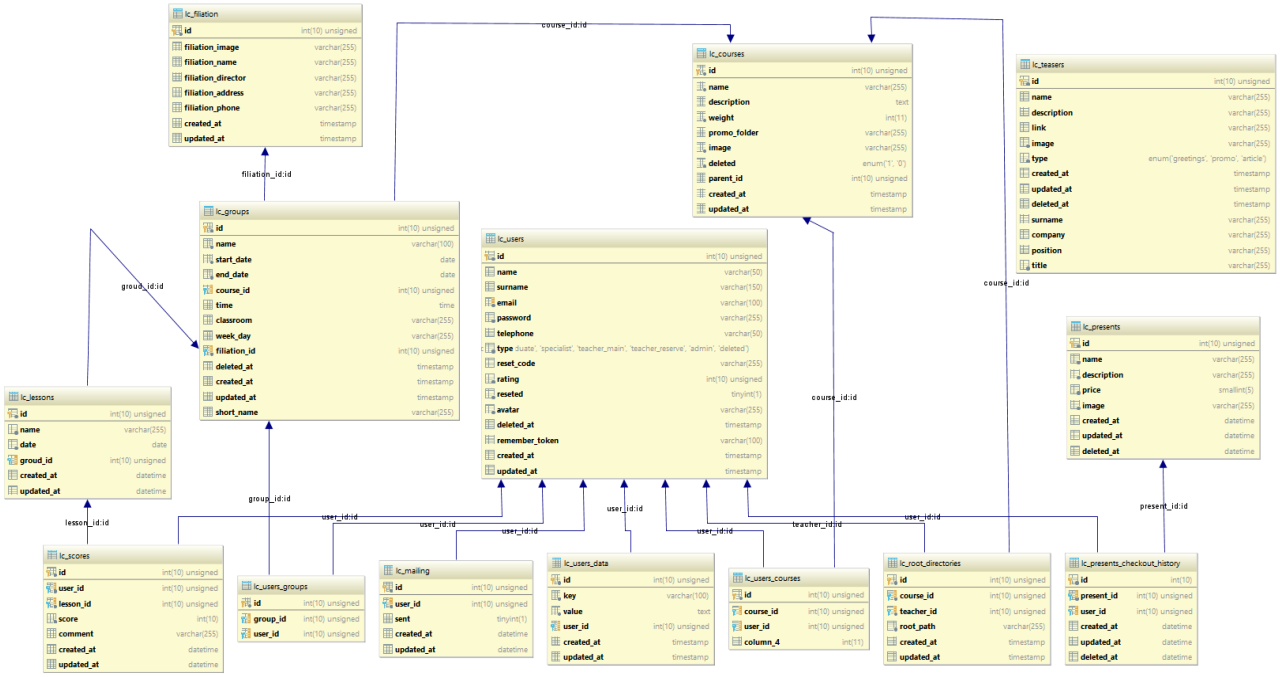

Рисунок 3.4. Структура бази даних

Таблиця lc_users складається з багатьох полів та призначена для зберігання даних про користувачів. Вона має такі поля, як: ім'я, прізвище, поштова адреса, пароль, телефон, тип користувача, його рейтинг, фотографію, а також поля, за допомогою яких можна переглянути, коли був створений даний користувач та коли останній раз відбулося оновлення даних для цього користувача.

Lc_users_data призначена для зберігання всієї додаткової інформації користувачів. Дана таблиця має два основні поля – це ключ та значення, які відповідно призначені для збереження певного ключа та його значення. Основна ціль цієї таблиці – це збереження всіх необов'язкових даних користувачів. Так, наприклад, деякі користувачі мають декілька мобільних телефонів. Ми не можемо знати наперед, скільки у кожного користувача може бути мобільних телефонів, тому, якщо користувач бажає ввести більше ніж один номер, ми

зберігаємо перший номер до основної таблиці lc_users, а всі інші до lc_users_data з відповідними ключами. Дани метод дозволяє уникнути розростання полів таблиці lc_users і зберігати до неї тільки основну важливу інформацію.

Таблиця lc_courses призначена для збереження даних про поточні курси. Вона має такі поля: назва, опис курсу, вага курсу, яка використовується для сортування курсів в системі, чим менше вага, тим вище буде виводитися курс в загальному списку курсів, картинка, id головного курсу для створення ієрархічної структури курсів, дата створення курсу та дата оновлення даних певного курсу.

Lc_users_courses – проміжна таблиця для створення зв'язку багато до багатьох між таблиця lc_users та lc_courses. Кожний користувач може одночасно проходити декілька різних курсів, тобто у нас виникає зв'язок один користувач до багатьох курсів. Аналогічно і один курс можуть проходити багато користувачів. Для реалізації такого типу зв'язку використовується проміжна таблиця, в якій зберігаються id курсу та id користувача.

Таблиця lc_groups призначена для збереження даних групи. Вона має наступні поля: назва групи, дата початку навчання, дата закінчення навчання, курс, за яким навчається дана група, номер класу та філіал, в якому будуть проходити заняття, день тижня, коли буде проходити навчання, дата створення та дата оновлення даних групи. Аналогічно до таблиці lc_users_courses, ми маємо для кожної групи таблицю lc_users_groups. Оскільки кожний користувач може належати до декількох груп одночасно, нам потрібна дана проміжна таблиця для реалізації зв'язку багато до багатьох.

Lc_teasers призначена для збереження даних про новини в системі. В дану таблицю зберігаються привітання та власне самі новини. Вона складається з таких полів: назва новини або привітання, опис, посилання на повний текст новини, картинка, тип новини, дата створення та дата оновлення даних про новину.

Таблиця lc_filliation призначена для збереження даних про філіали навчального закладу. Навчальний заклад може мати декілька корпусів або місць

для навчання, тому важливим є розуміння, в якому корпусі навчається та чи інша група. Дана таблиця складається з наступних полів: картинка філіалу, назва, директор чи декан даного філіалу, адреси філіалу, контактного телефону, дати створення та оновлення даних про філіал.

Lc_presenters призначена для збереження можливих призів та сертифікатів, які надає те й чи інший навчальний заклад. Дана таблиця складається з таких полів: власне назва призу, опис, ціна, картинка, дата створення та оновлення даних про нього. Тобто в дана таблиця призначена для збереження подарункових та інших сертифікатів, кожний з яких можна купити за певну ціну. Оскільки потрібно знати, кому належить певний сертифікат та який користувач його придбав, використовується ще одна додаткова таблиця lc_presenters_history. В даній таблиці зберігається дані про користувача, який купив певний сертифікат, власне посилання на сертифікат, який був куплений та дата купівлі сертифікату.

Навчальний матеріал кожного курсу поділений на уроки, які повинен пройти кожний студент та виконати потрібні завдання. Для створення уроків та збереження даних про них використовується таблиця ls_lessons, яка має наступні поля: назва уроку, дата проведення уроку, група, для якої було або буде проведений даний урок, дата створення та оновлення. Кожний студент групи за кожний урок отримує оцінку по 5-бальні шкалі. Для збереження всіх оцінок студентів використовується таблиця lc_scores. В даній таблиці зберігаються дані про користувача, урок, оцінку та коментар викладача.

Таблиця lc_root_directories використовується для збереження даних про директорії, які відповідають кожному курсу. Система має розділ методичних матеріалів, які дозволяють студентам переглядати матеріали певного уроку чи напряму, переходячи по директоріям та папкам, які призначені нього. Саме для даних цілей використовується дана таблиця. В ній зберігаються дані про викладача, який створив ту чи іншу директорію, напрям, для якого вона була створена та повний шлях до даної директорії.

3.2.2. Створення індексів для полів таблиць

Перш за все для створення індексів потрібно проаналізувати систему та виділити запити, які будуть найбільш частіше виконуватися користувачами та адміністратором бази даних.

Першим, що бачить користувач при вході в систему – це форма авторизації. Коли користувач вводить свою поштову адресу та пароль, ми виконуємо пошук користувача за його поштовою адресу і потім порівнюємо паролі. Оскільки в нас в системі може бути зареєстрована велика кількість користувачів – як студентів, так і викладачів, пошук повним переборов всіх записів буде займати досить багато часу, тому перший індекс, який був створений – індекс для поля email таблиці lc_users. Це дозволить зменшити час на пошук потрібного користувача. Для створення індексу для певного поля бази даних ми будемо використовувати міграції. Оскільки для даного поля вже створено унікальний індекс, воно не потребує ніяких змін(рис. 3.5)

```
$table->string('email', 100)->unique();
```

Рисунок 3.5. Створення унікального індексу для поля email

Після того, як користувач авторизувався в системі, ми йому показуємо новини. Перша новина, як правило, є привітанням переможців певного конкурсу чи іменинників, а три інші просто останні та важливі новини. Оскільки їх всього 4 та редагування відбувається за допомогою безпосереднього натискання на новину, немає необхідності в створенні для них індексів.

Методичні матеріали дозволяють отримати доступ до файлів того чи іншого уроку, як правило, здійснюється виведення тільки доступних для даного користувача матеріалів. Ми маємо всього 8 матеріалів, які можуть бути доступні користувачам. Так як пошук чогось серед всього лише 8 записів не є досить трудомісткою операцією, створювати індекси для даної таблиці немає необхідності.

Пошук студентів є досить трудомісткою операцією, оскільки їх кількість є доволі великою. Ми вже додавали індекс для поля email, тому це поле вже можна не розглядати. На даній вкладці ми можемо фільтрувати студентів за напрямом, курсом, статусом та здійснювати пошук за прізвищем. Тому потрібно додати індекс для поля surname таблиці lc_users, що забезпечить зменшення часу виконання запиту з пошуку певного студента за його прізвищем.

Групи, які існують в системі, можна фільтрувати за напрямом, курсом, статусом, початковою та кінцевою датою. Ми маємо не таку велику кількість груп, тому додавання індексу є необов'язковим, тому поки залишимо таблицю, в якій зберігаємо дані про групу без змін.

Викладачі – це по суті ті ж самі користувачі, що і студенти. Ми можемо здійснювати фільтрацію викладачів за напрямом, курсом та статусом викладача. Для користувачів може бути доцільним додавання індексів для напряму та курсу, адже ці операції будуть виконуватися доволі часто і будуть потребувати значного часу, тому додатково додамо індекси для даних полів.

Ми маємо всього три філіали, тому виконання запитів по їх пошуку не є трудомісткою операцією і немає сенсу додавати для них будь-які індекси.

На вкладці "Успішність" ми вибираємо зі всіх можливих груп певну групу та виконуємо перегляд уроків даної групи. Відображається весь список уроків даної групи. Дана операція не є досить трудомісткою, тому немає необхідності у додаванні додаткових індексів до неї.

3.3. Розробка моделей для роботи з базою даних

Оскільки ми маємо базу даних, яка складається з 14 таблиць. Нам відповідно потрібно буде створити 12 моделей для роботи з кожною таблицею. Всі користувачі системи для навчання студентів та керуванням навчальним процесом зберігаються в таблиці users, тому розглянемо модель для даної таблиці.

Модель для таблиці users має наступний вигляд(рис 3.6)

```php
class User extends Authenticatable
{
    use Notifiable;

    use ModelMap;

    const LIMIT = 10;

    const DEFAULT_AVATAR = 'storage/avatars/default.png';

    /**
     * The attributes that are mass assignable.
     *
     * @var array
     */
    protected $fillable = [
        'name',
        'surname',
        'telephone',
        'email',
        'password',
        'type',
        'course_id',
    ];

    protected $casts = [
        'id' => 'integer',
        'course_id' => 'integer',
        'rating' => 'integer',
    ];

    protected $table='lc_users';

    /**
     * The attributes that should be hidden for arrays.
     *
     * @var array
     */
    protected $hidden = [
        'password',
        'remember_token',
        'reseted',
        'reset_code'
    ];
```

Рисунок 3.6. Модель для роботи з користувачам

Можна помітити, що в даній моделі наявні 4 основні властивості *fillable*, *casts*, *table* та *hidden*. Розглянемо детально, за що відповідає кожна з ним та які особливості вони мають.

Властивість *fillable*, відповідає за поля, які можуть бути додані в модель при її створенні, як асоціативний масив. Даних масив повинен складатися із ключів, які відповідають назві відповідного поля та відповідних їм значень. Це дозволяє в зручний спосіб здійснювати створення нових моделей та зберігати дані до бази даних. Вам потрібно всього лише передати масив в конструктор певної моделі і ви отримуєте повноцінний клас, в якому властивостями є назви полів, а значеннями відповідно передані вами значення. Даний клас має всі методи для роботи з базою даних.

Наступною властивістю є *casts*. Як вже зазначалося раніше, кожна модель має всі необхідні методи для роботи з базою даних. Досить часто виникає необхідність у чіткому приведенні певних значень полі до деякого типу. Так, наприклад, вам потрібно, щоб поле rating завжди перед збереженням до бази даних приводилося до цілочисельного значення та зберігалося в базу даних, як число. Таким чином, дана властивість забезпечує чітке приведення типів для полів *id, course_*id та *rating*.

Фреймворк *Laravel* автоматично визначає назву таблиці, до якої належить модель за допомогою наступної логіки – назва таблиці в базі даних є назвою моделі в однині. Наприклад, якщо модель має назву *User*, значить він автоматично передбачає, що таблиця в базі даних буде мати назву *users*. Оскільки в нас таблиця користувачів має назву, відмінну від *users,* ми за допомогою властивості *table* чітко вказуємо назву таблиці, з якою повинна працювати дана модель.

Останньою властивістю є *hidden*. Дана властивість забороняє заповнювати перераховані поля моделі, просто передавши їх у вигляді масиву чи *json*. Це здійснюється для забезпечення надійності в роботі системи та для уникнення збереження в базу даних важливих полів із некоректними значеннями.

Якщо таблиця має зв’язки у вигляді зовнішніх ключів з іншими таблицями, вони всі за допомогою спеціальних методів повинні бути прописані в моделях. Вони дозволять у зручний спосіб виконувати запити до пов’язаних моделей, які відповідають іншим таблицям та витягувати всі потрібні дані, викликавши лише одним метод. При виконанні виклику даного методу відбувається ліниве завантаження іншої моделі та отримання потрібних даних.

Так, як таблиця lc_users, має велику кількість зв’язків з іншими таблицями, відповідно в моделі даної таблиці також повинні бути прописані всі зв’язки з іншими моделями, які відповідають за інші таблиці. Розглянемо деякі з ним для розуміння загальної особливості у створенні зв’язків між моделями(рис 3.7)

```php
/**
 * Get all additional user data
 *
 * @return \Illuminate\Database\Eloquent\Relations\HasMany
 */
public function usersData()
{
    return $this->hasMany(UserData::class );
}

/**
 * Get course that selects user
 *
 * @return \Illuminate\Database\Eloquent\Relations\HasMany
 */
public function courses()
{
    return $this->hasMany( Course::class, 'lc_users_courses' );
}
```

Рисунок 3.7 – Зв’язки між моделями

З даного фрагменту коду можна зрозуміти, що модель users має два методи *userData* та *courses*, при виклику яких відбувається виконання запиту до даних таблиць через їх моделі та отримання даних, які пов’язані з певним конкретним користувачем з іншим таблиць. Таким чином, при розробці логіки програми дана особливість та моделі загалом дозволяють працювати з базою даних на високому рівні без виконання SQL запитів. Всі операції з базою даних виконуються через моделі, які є свого роду проміжним прошарком між власне базою даних та нашим програмним кодом.

3.4. Розробка RESTfull API

На наступному етапі було проведено розробку Restful API(рис. 3.8) Оскільки ми маємо повністю закриту систему, для доступу до якої користувачу спочатку потрібно авторизуватися в системі, всі можливі URL системи будуть мати *middleware*, який перед надсиланням даних до контролеру та виконання певної логіки виконує перевірки прав поточного користувача і якщо користувачу дозволено виконувати дану дію, йому відкриється сторінка, яку він запитував. Якщо у користувача недостатньо прав доступу, йому буде відображена

стандартна сторінка із повідомленням, що в нього недостатньо прав для доступу до даної сторінки.

```php
/**
 * users routes
 */
Route::post('/users/filter/{name}', 'UserController@filterStudents')
    ->where(['name' => '[A-Za-z]+'])->middleware('admin');
Route::post('/users/filterbysurname/{sname}', 'UserController@getStudentsBySurname')
    ->where(['name' => '[A-Za-z]+'])->middleware('admin');
Route::post('/users/delete/{id}','UserController@destroy')->middleware('admin');
/**
 * groups routes
 */
Route::group(['middleware' => ['auth','admin'], 'prefix' => 'groups'], function (){

    Route::get('/', 'GroupController@index')->name('groups');

    Route::get('/data',  'GroupController@getGroupsData');

    Route::post('/add', 'GroupController@store');

    Route::put('/update/{groupId}', 'GroupController@update')
        ->where(['groupId' => '[0-9]*']);

    Route::post('/filter', 'GroupController@filterGroups');

    Route::delete('/delete/{id}','GroupController@destroy');

    Route::post('/get-teacher-by-course-and-direction','GroupController@getTeacher');

    Route::post('{id}/restore','GroupController@restore');
});
```

Рисунок 3.8. REST API системи для навчання студентів та керування навчальним процесом

З даного фрагменту можна помітити, що для задання *middleware*, потрібно передати асоціативний масив, де ключем масиву буде виступати власне назва цього помічника, а параметрами виступає масив, в який передається метод, який повинен бути викликаний для перевірки прав користувача на виконання даних дій та які прав повинен мати даний користувач.

Наступний параметром масиву передається *prefix*, який забезпечує додавання до всіх роутів даного введеного розробником слова. Тобто, якщо розглянути перший роут для груп, то він буде виглядати наступний чином – *groups/*.

Кожний роут системи складається певний частин та відповідає надсилання даних до конкретного контролера системи. На початку кожного правила йде зарезервоване ключове слово *Route*. Далі через двокрапку вказується назва HTTP методу, який бути задіяний. Існує 5 основних HTTP методів, які використовують для виконання різних дій в системі(табл. 3.1)

Таблиця 3.1. HTTP методи

Назва методу	Дія, яку повинен виконувати
POST	Створення певного нового запису
GET	Читання певних даних без внесення змін до них. Використовується, наприклад, для отримання контенту головної сторінки
PUT	Оновлення та повна заміна даних
PATCH	Оновлення та зміна певного поля для певної таблиці
DELETE	Видалення даних

Після того, як визначено метод, який повинен використовуватися в залежності від виконуваних дій, прописується URL, при введенні в командному рядку браузера якого, відбувається виклик визначеного контролера. Наступним параметром власне передається назва контролера та назва конкретного методу, який повинен викликатися при введенні в браузері даного конкретного роута. Додатковий метод *name* із передачею в нього назви, використовується для того, щоб можна були динамічно формувати на сторінці, яка відображається користувачу правильну URL адресу, вказавши всього лише назву роуту, для якого ми хочемо її сформувати. При натисненні користувача на дану адресу, буде автоматично виклика команда(роут), який має дане ім’я.

При використанні зарезервованого слова *resource* ситуація виглядить іншим чином. Справа в тому, що дане слово передбачає наявність в контролері 7 методів для роботи з певними даними. Дані методи є однаковими для кожного контролера і описують всі дії, які потрібні для роботи з певним типом матеріалів. Це є досить зручним та скорочує кількість програмного коду, оскільки дозволяє одним словом задати 7 методів. Також в даних правилах можна вказувати, перелік методів, які повинні використовуватися за допомогою ключового слова *only* або вказувати методи, які не потрібно використовувати за допомогою слова *expect*.

Для роботою з групами ми маємо наступну структуру методів (табл. 3.2)

Таблиця 3.2. Структура роутів та методів для виконання операцій над групами

Метод	Адреса	Назва методу	Дія для компанії
GET	groups/	Index	Відображення html шаблону
GET	groups/data	GetGroupsData	Отримання всієї інформації про всі групи
POST	groups/add	Strore	Створення
PUT/PATCH	groups/updated/{groupId}	update	Редагування
POST	/ groups /filter	filterGroups	Фільтрація груп
DELETE	groups/delete/{ groupId }	Destory	Видалення
POST	groups /{groupId} /restore	Restore	Відновлення

Список методів наведених в таблиці повністю покривають логіку роботи з групами. Можна помітити, що в даному списку є два досить схожі методи *groups/* та *groups/add*. Основна відмінність даних методів полягає в тому, що перший метод відповідає за завантаження базових елементів сторінки для відображення груп, таких як: головне меню сторінки, шапку сторінки, поточного зареєстрованого користувача та іншу інформацію. При завантаженні сторінки підключається скрипт, який виконує додатковий запит до бекенд частини та який отримує всю необхідну інформацію про групи, які наявні в даній системі. Далі виведення груп, їх обробка, оновлення, редагування та фільтрація відбувається динамічно за допомогою javascript та обміном між бекендом даними за допомогою ajax запитів.

3.5. Чітка валідація даних та механізм обробки помилок

Кожна система повинна мати чітку валідацію всіх даних, які приходять від користувача. Це дозволить уникнути непередбачених ситуацій та помилок, які вже будуть власне наслідком некоректних даних, а не їх причиною. Оскільки дана система використовує методологію MVC, кожний запит проходить стандартний шлях виконання. При завантаженні певного URL в нас відбувається надсилання запиту в певний метод певного класу, назва якого залежить від власне шляху, на який ми відправляємо запит. Всі можливі шляхи та відповідні методи, які відповідають за їх обробку, прописуються в окремому файлів та

використовуються фреймворком для пошуку шляхів та виклику відповідного методу. Отже, після введення певного шляху в URL, відбувається виклик певного методу із параметрами. Значення цих параметрів ми визначаємо власноруч. Існує механізм, який дозволяє створювати свої власні класи із набором правил і передавати їх, як параметри до методу. При передачі цих параметрів відбувається аналіз даних, які приходять від користувача та порівняння їх зі значеннями, які задані в певному параметрі. Якщо всі значення задовольняють допустимі значення параметра, вони зберігаються та передаються далі на виконання. Тобто, перед тим, як починати використовувати дані, які ми отримуємо від користувача, вони спочатку повністю перевіряються і тільки вже потім передаються на обробку та виконання певної логіки над ними.

За перевірку введених даний, як вже зазначалося, відповідає метод певного параметру, де під параметром ми можемо розуміти певний клас. В класі є метод rules, який відповідає за перевірку значень(рис. 3.9)

```php
/**
 * Get the validation rules that apply to the request.
 *
 * @return array
 */
public function rules()
{
    return [
        'name' => 'required|string|min:2|max:50',
        'surname'=> 'required|string|min:2|max:150',
        'email' => 'email|unique:lc_users',
        'telephone' => 'required|string|min:9|max:50',
        'type' => 'required|in:entrant,student,graduate,specialist,teacher_main,teacher_reserve',
        'course.*' => 'required|integer',
        'userData.sex' => 'string|in:men,women|nullable',
        'userData.birthday_date' => 'string|min:6|nullable',
        'userData.city' => 'string|min:2|nullable',
        'userData.address' => 'string|min:2|nullable',
        'userData.note' => 'string|min:2|nullable',
    ];
}
```

Рисунок 3.9. Правила валідації даних перед виконання їх обробки

Даний метод повинен повертати масив наступного типу: кожний ключ масиву є назвою поля, яке потрібно перевірити, а значенням цього ключа є власне набір правили, які потрібно перевірити. Наприклад, якщо ми хочемо перевірити поштову адресу користувача, ключем масиву буде значення email, а значенням цього елементу рядок із наступним вмістом – “*required|email|unique:users*”. За допомогою даного символу “|” відбувається

поділ рядку на правила, тобто в даному випадку для поля email ми маємо три правила. Перше правило "*required*" вказує на те, що дане поле є обов'язковим і завжди повинно бути при отриманні даних. Якщо ж поле не обов'язкове, тобто воно може бути, а може і бути пустим чи взагалі відсутнім, дане правило не вказується. Наприклад, реальна адреса користувача є необов'язковим полем, користувач може його ввести або просто залишити пустим. Наступне правило "*email*" вказує на те, що в даному полі повинна бути поштова адреса користувача і для його перевірки будуть застосовуватися регулярні вирази для перевірки всіх необхідний атрибутів поштової адреси. Остання правило "*unique:lc_users*" вказує на те, що дане поле повинно бути унікальним для кожного користувача або запису в таблиці *lc_users*. Для ідентифікації користувачів використовується якесь певне унікальне поле, яке б дозволяло зрозуміти, прийшов новий користувач чи даний користувач вже зареєстрований в системі. Так, наприклад, якщо користувач вже є в системі і ви намагається зареєструватися, потрібно це виявити і відправити йому повідомлення, що такий запис вже є в системі і він вже реєструвався раніше. Як правило, унікальним ідентифікатором є власне email користувача, оскільки він є унікальним. В даному правилу вказується, що дане поле повинно бути унікальний для таблиці *lc_users* і перед тим, як здійснювати обробку даних відбувається порівняння даного поля з полем email таблиці на наявність записів з такою поштовою адресом. Отже, набір правил дозволяє провести перевірку даних, які ми отримуємо від користувачів, перед їх обробкою та уникнути помилок в системі. При отриманні некоректних даних відбувається відправлення помилок із вказанням поля, яке або які не пройшли валідацію та власне конкретної помилки.

3.6. Розробка логіки роботи основних модулів

На головній сторінці даної системи розміщуються привітання та новини. За відображення та роботу із головною сторінкою системи для навчання студентів та керування навчальним процесом відповідає *TeaserController*, який має наступну структуру(рис. 3.10)

```php
class TeaserController extends Controller
{
    public function getTeasers()
    {
        $greetings = $this->getTeaserBuilder( take: 1, type: 'greetings');
        $promo = $this->getTeaserBuilder( take: 1, type: 'promo');
        $articles = $this->getTeaserBuilder( take: 2, type: 'article');
        return response([
            'greetings' => $greetings,
            'promo' => $promo,
            'articles' => $articles,
            'can_edit' => Auth::user()->type === 'admin',
        ]);
    }

    public function create(TeasersRequest $request){
        $teaser = new Teaser();
        $data = $request->all();
        if($request->hasFile('image'))
            $data['image'] = parent::saveImage($request->file('image'), previousFile: Teaser::DEFAULT_IMAGE,
                defaultImage: Teaser::DEFAULT_IMAGE, storagePath: 'teasers');
        return response($teaser->map($data));
    }

    public function edit(Request $request)
    {
        $teaser = Teaser::findOrFail($request->id);
        $data = $request->all();
        unset($data['image']);
        if($request->hasFile('image')) {
            $data['image'] = parent::saveImage($request->file('image'), $teaser->image,
                defaultImage: Teaser::DEFAULT_IMAGE,  storagePath: 'teasers');
        }
        return response($teaser->map($data));
    }

    public function destroy(Request $request)
    {
        Teaser::findOrFail($request->id)->delete();
    }
```

Рисунок 3.10. Контролер та логіка роботи головної сторінки

При завантаженні головної сторінки відбувається виклик методу *getTeasers*, який здійснює виконання запитів до бази даних через моделі та повертає дані у вигляді json об’єкту. Можемо помітити, що в нас відбувається виконання запиту до бази даних для отримання останнього доданого привітання, оскільки при виклику методу передається значення 1. Потім відбувається виклик методів для отримання статей та промо-банерів, які створені для повідомлення користувачів про цікаві курси та можливі скидки. Всі отримані дані збираються в асоціативний масив із відповідними ключами та віддаються на фронтенд. Можна помітити, що останнім параметром даного асоціативного масиву є ключ, який визначає права користувача, що в даним момент зайшов у систему. Дане поле служить для того, що знати на фронтенд частині про права поточного користувача та дозволяти редагування новин системи лише користувачам, які права адміністратора.

В даному контролері присутні також наступні методи:

– create – для створення відповідно новин та привітань в системі. В даному методі відбувається перевірка отриманих даних та їх збереження до бази даних. Після виконання збереження відбувається надсилання нового запису із оновленими даними;

– edit – для редагування даних вже існуючої новини. Відбувається пошук новини за її унікальним ідентифікатором. Якщо новина знайдена, відбувається оновлення даних та повернення оновленої новини, інакше – відбувається повернення помилки про неможливість здійснити зміни із вказанням конкретної помилки;

– destroy – для видалення новини із бази даних. Якщо новина знайдена, її буде успішно видалено, інакше – повернено повідомлення про помилку.

Аналогічно до даного контролера для виконання всіх інших дій в системі створені відповідні контролери. Можна помітити, що даних контролер називається *TeaserController*, так для створення груп використовується відповідно *GroupsController*, який має аналогічну структуру до розглянутого контролеру, має аналогічні методи та виконує дії пов’язані із обробкою та отримання даних про групи. Контролери можуть мати багато інших методів, якщо вони потрібні для реалізації певної складної логіки в системі, але загалом всі мають перераховані та розглянуті методи, які виконують аналогічні дії, але мають свою логіку їх виконання відповідно до зв’язків між іншими таблицями та операцій, які потрібно спершу виконати над отриманими даними.

Оскільки користувачі є центральними сутностями бази даних та системи загалом, розглянемо алгоритм створення нового користувача в системі рис. 3.11 Перш за все в нас відбувається фільтрація отриманих даних та перевірка їх типів та коректності. Якщо отримано коректні дані, відбувається визначення типу користувача, який нам потрібно створити. Якщо тип, в запитів, який нам прийшов із фронтенд частини відповідає одному із можливих ключів студентів, значить нам потрібно створити студента, інакше – потрібно створити викладача.

```
$status = 200;
$statuses = User::getUserStatuses();
try {
    if( in_array($request->type, $statuses[ self::STUDENTS_KEY ] ) ) {
        $userType = self::STUDENTS_KEY;
    } else {
        $userType = self::TEACHERS_KEY;
    }
    $user = new User([
        'name' => $request->name,
        'surname' => $request->surname,
        'email' => $request->email,
        'telephone'=>$request->telephone,
        'password' => bcrypt('12345'),
        'type' => $request->type,
    ]);
    $user->save();

    $user->courses()->attach($request->course);

    if( ! empty ( $request->userData ) ) {

        $additionalUserData = [];
        foreach ( $request->userData as $key => $value ) {
            if( null !== $value  ) {
                $additionalUserData[] = new UserData(['key' => $key, 'value' => $value]);
            }
        }
        $user->usersData()->saveMany( $additionalUserData );
    }
    $students = $filter->filtration([],$userType);
    $amountStudents = $students->count();
    $responseData = [
        'students' => self::sliceCollection($students, offset: 0, amount: self::LIMIT),
        'amountStudents' => $amountStudents,
    ];

} catch (\Exception $exception) {
    $status = 500;

    if(Auth::user()->isAdmin()) {
        $message = $exception->getMessage();
    } else {
        $message = 'При сохранении данных произошла ошибка. Пожалуйста, обратитетсь к администратору';
    }

    $responseData = [
        'error' => $message
    ];
}
```

Рисунок 3.11. Алгоритм створення користувачів

На наступному кроці відбувається створення нової моделі для користувача та автоматичне заповнення її даними, які ми отримали. Після цього відбувається збереження даних до бази. Далі відбувається створення зв’язків між новим користувачем та курсом, який він бажає проходити.

Наступний кроком є перебір всі інших даних користувача та запис їх до таблиці, в якій зберігаються всі необов’язкові дані користувачів. Для цього

отриманий асоціативний масив переглядається та відбувається створення нових моделей, де ключем є ключ поля асоціативного масиву, а значенням – власне значення даного конкретного поля. Після обробки всіх даних відбувається їх збереження до відповідної таблиці із встановленням зв'язків для запису цього користувача в таблиці *lc_users*.

На останньому етапі відбувається підрахунок кількості всіх користувачів та повернення всіх користувачів системи із відповідними статусами. Можна помітити, що весь даний метод огорнутий у конструкцію *try-catch*, яка забезпечує перехоплення помилки в разі її виникнення при створенні нового користувача на основі отриманих даних. В разі виникнення помилки, ви отримуємо повідомлення, в якому пояснюється причина її виникнення та повертаємо її адміністратору для подальшого розгляду та виконання дії, що уникнути дану помилку в майбутньому.

3.7. Впровадження системи кешування даних Redis

Система кешування дозволяє зберігати дані у вигляді ключ значення та здійснювати їх швидкий пошук за ключем. Швидкість досягається за рахунок зберігання даних в оперативній пам'яті комп'ютера, але при цьому потрібно пам'ятати, що при перезавантаженні серверу, всі дані з оперативної пам'яті видаляється, тому доцільно розміщувати дану систему на окремому сервері відмінному від серверу, на якому розміщується власне веб-додаток. Потім здійснюється налаштування його для веб-додатку, шляхом встановлення ip адреси та порту, за яким ми можемо звертатися до серверу в Redis. Спираючись на проведений вище аналіз, можна зробити висновок, що найбільше в кешуванні даних потребує таблиця з данини про користувачів. Тому аналогічно до створення індексів, кешувати будемо дані для таблиці студентів. Для цього ми допишемо логіку в процес виконання запитів до даної таблиці наступним чином: при виконанні запиту на пошук певного користувача за його email адресою або прізвищем, ми спочатку будемо виконувати запит до серверу Redis за ключем, якщо це поштова адреса, то відповідно ключ може бути user.email, де замість

email ми повинні динамічно підставляти реальну адресу користувача. Якщо такий запис знайдено, то ми далі відразу віддаємо даний результат на обробку та показ користувачу. Якщо ми нічого не знайшли, в даному випадку виконується запит до бази даних та здійснюється пошук. Після успішно знайденого результату, ми його беремо та записуємо в систему кешування під ключем user.email і потім віддаємо результат користувачу. Таким чином, ми кожний раз будемо наповнювати базу Redis та при виконанні запиту наступного разу вже отримувати дані з неї. Оскільки в нас користувачі можуть змінювати свої особисті дані, постає питання в збереженні достовірних даних в базі даних Redis та власне в базі даних MySQL. Для цього на потрібно проаналізувати всі запити, пов’язані зі зміною даних користувачів та при кожній такій зміні здійснювати їх оновленні і в базі даних Redis. Наприклад, якщо користувач вирішив змінити адресу проживання або пароль, ми повинні виконати всі необхідні операції для цього, зберегти оновлені дані до звичайної бази даних і після цього обов’язково оновити базу даних системи кешування. Найбільш складною буде операція користувачів зі зміни поштової адреси, адже в нас ключ формується з використанням поштової адреси користувачів. В даному випадку потрібно виконувати пошук даних користувача з використанням минулої поштової адреси та здійснити видання цього ключа з бази даних. Після цього можна створити новий запис із новий ключем. Потрібно пам’ятати про ці всі важливі дії, оскільки в іншому випадку все це може призвести до того, що дані будуть відрізнятися і ми будемо відображати користувачу недостовірні дані.

3.8. Виконання складних задач за допомогою cron

Всі веб-додатки розміщується на серверах. Як правило, на сервері використовується unix-подібна операційна система, яка забезпечує належний захист веб-додатку та серверу від вірусів та інших шкідливих програм. Операційна система windows може застосовуватися, але це є не доцільним, оскільки вона є дуже вразливою і може призвести до злому вед-додатку та інших неправомірних дій. Unix – подібні операційні системи мають утиліту, яка власне

і називається cron. Вона дозволяє здійснювати виконання скриптів у автоматичному режимі в певний час. Тобто створюється файл для неї і у файлів прописується розклад, в який час повинен запуститися той чи інший файл, яким чином він повинен запуститися та шлях до самого файлу. Це дозволяє в дуже зручний спосіб здійснювати складні операції без стороннього втручання. В системі для навчання студентів та керування навчальним процесом є можливість масово надсилати повідомлення користувачам. Якщо ми маємо 100.000 користувач, складно уявити скільки буде виконуватися відправка повідомлень користувачам. В даному випадку адміністратору даної системи доведеться запускати надсилання повідомлень та чекати дуже багато часу, поки всі повідомлення будуть відправлені. До того ж адміністратор не може слідкувати в режимі реального часу, сам процес виконання надсилання повідомлень кожному користувачу. Тобто, наприклад, він запустив команду на виконання і він не може переглянути, якому користувачу в даний момент відбувається надсилання повідомлення, не може в режимі реального часу бачити, що певному користувачу не вдалося надіслати повідомлення через певні проблеми чи некоректну поштову адресу. Тому доцільно використовувати вбудовану в операційні системи утиліту для автоматичного запуску команд та скриптів. Для виконання масової розсилки буде використовувати наступний розклад. Можна помітити, що спочатку в нас прописано за допомогою “*” та “/”, коли повинна виконуватися дана команда. Ми маємо 5 параметрів, якими ми можемо керувати для задання часу виконання команди. Перший параметр – це хвилини, другий – години, третій – дні місяця, четвертий – порядковий номер місяця, п’ятий – дні неділі від 0 до 6, де 0 – це неділя, а 1 – понеділок. Якщо ми в якості параметру використовуємо символ “*”, це означає, що ми хочемо, щоб команда виконувалася кожну хвилину, годину, місяць в залежності від номеру параметра, якому ми задаємо. В нашому випадку всі 5 параметрів мають значення “*”, тобто дана команда буде запускатися кожну хвилину. Виконувати надсилання повідомлень відразу 100.000 користувачів досить трудомісткий і тривалий процес, оскільки потрібно спочатку отримати з бази даних 100.000 користувачів,

потім їх помістити в оперативну пам’ять для роботи з ними і потім намагатися їх всіх обробити. Саме тому, ми будемо вибирати з бази даних всього 1.000 користувачів і виконувати їх надсилання повідомлень. Для того, щоб знати кому ми вже надіслали повідомлення ми створимо ще одну додаткову таблицю, куди будемо записувати користувачів, які вже отримали повідомлення. Таким чином, при виконанні запиту до бази даних для отримання користувачів ми додамо перевірку на те, чи отримав даний користувач повідомлення, яке відноситься до поточної розсилки. Наступне питання, яке потрібно вирішити – це уникнення колізій та помилок із відміткою студентів, які отримали повідомлення. Нехай крон запустив скрипт, який виконує надсилання повідомлень 1000 користувачів і для виконання даної дії потрібно 3 хвилини. Оскільки виконання запитів до бази даних на оновлення та запис даних в циклі є поганою практикою, тому що збільшується навантаження на базу даних, тому рекомендується зберігати всі потрібно запити і потім їх виконати відразу за одне звернення до бази даних. Крон запускає скрипт кожну хвилину і кожний раз скрипт завершує всі дії протягом 3 хвилин. Тобто, при першому запускі скрипта, ми отримуємо з бази даних першу 1000 користувачів, через хвилину крон запускає скрипт знову і ми знову отримаємо тих самий користувачів, оскільки скрипт, який був запущений першим, виконує помітку про те, що користувач отримав повідомлення після надсилання всіх повідомлень і в кінці своєї роботи. Таким чином, ми отримаємо ситуацію, коли одні і ті ж самі користувачі отримають декілька разів одне і те ж саме повідомлення, що не задовольняє наші вимоги. Тому ми в логіку роботи скрипта додамо перевірку на те, чи на даний момент немає вже запущених скриптів, які виконують надсилання повідомлень. Якщо такі скрипти є, то ми просто нічого не виконуємо, якщо немає – починаємо виконувати розсилку. Вся логіка буде залишатися в самому коді скрипта і після одного правильного налаштування процесу все буде перевірятися і виконуватися автоматично. Адміністратору потрібно буде тільки в обрати, кому він хоче надсилати повідомлення і натиснути кнопку “Розіслати”. Після натискання на кнопку ми записуємо в базу даних подію, що нам потрібно виконати розсилку і

адміністратору відображається повідомлення, що розсилка буде виконана автоматично. Як ми пам’ятаємо, в нас крон запускає скрипт кожну хвилину, після його автоматичного запиту та звернення до бази даних для отримання інформації, про розсилку і чи потрібно її виконувати в даний момент, скрипт виконає всю необхідну логіку і виконає надсилання повідомлень(рис. 3.12)

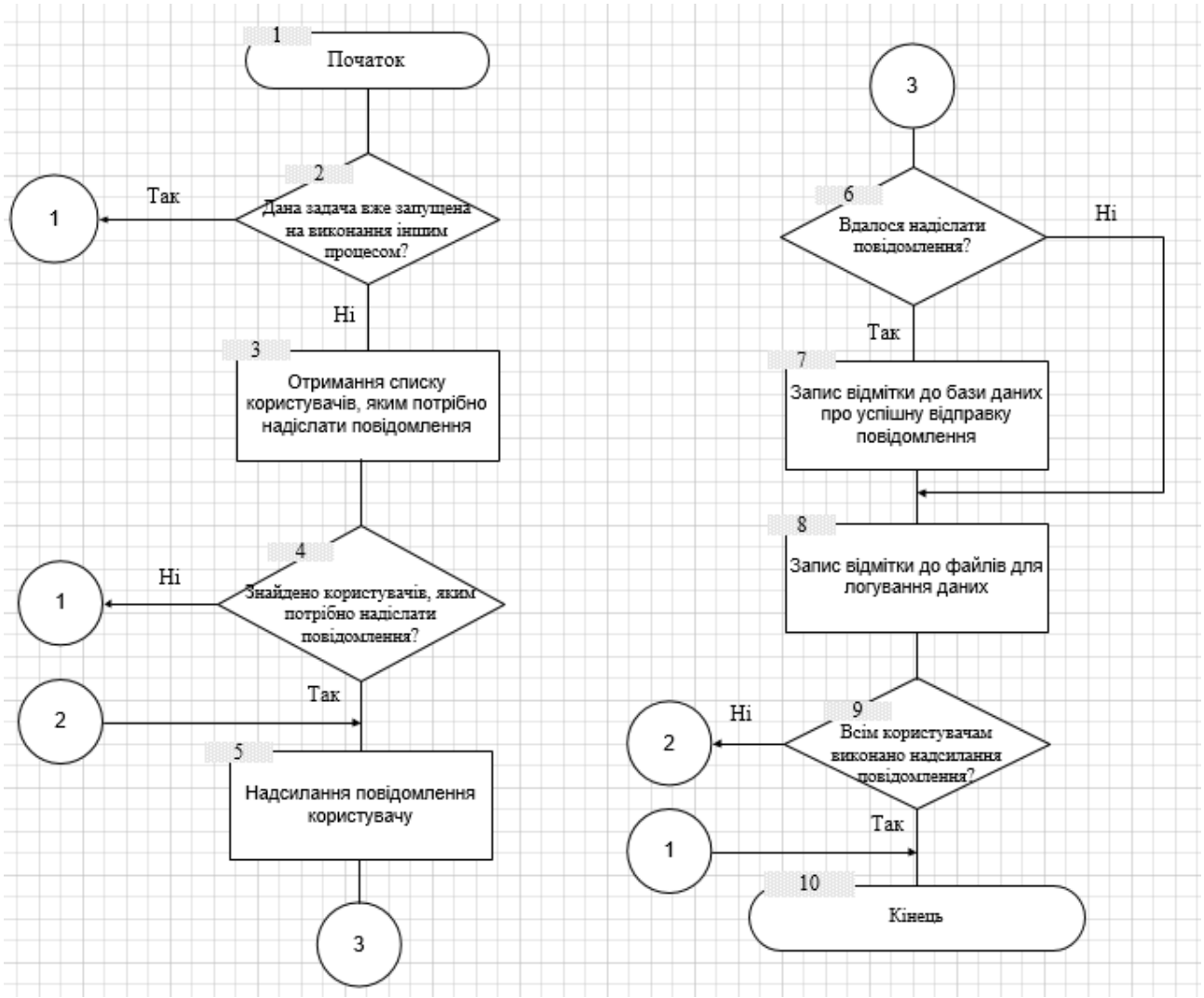

Рисунок 3.12. Алгоритм виконання розсилки повідомлень

При виконання роботи скрипт автоматично буде записувати всі свої дії у файл, який потім адміністратор може переглянути та проаналізувати.

3.9. Написання unit тестів до програми

Для перевірки програми, а точніше її компонентів, на наявність помилок використовують unit тестування. Його особливість полягає в тому, що вибирається певний компонент або метод класу і для нього ми знаємо, які

повинні бути вхідні та вихідні дані. Далі, використовуючи бібліотеку PHPUnit ми вибираємо потрібний нам метод і пишемо, що на вхід такого-то компоненту повинні прийти такі дані, а в результаті його роботи повинні вийти такі результати. Потім ми запускаємо написаний unit тест на виконання і отримуємо результат, в якому відображається чи співпадає очікуваний результат з дійсним. Цей метод тестування є досить зручним, адже дозволяє знайти всі помилки в програмі при роботі, наприклад, в тестовому оточенні та при перенесенні системи на робочий сервер.

Для перевірки правильності виконання алгоритму підрахунку загального балу студентів були написані наступні unit тести (рис. 3.13).

```
class UserRaitingTest
{
    public function testingCommonStudentRaiting1()
    {
        $raiting = new Raiting();

        $this->assertEquals(30, $raiting->calculate(User::fintOrfail(1)));
    }

    public function testingCommonStudentRaiting2()
    {
        $raiting = new Raiting();

        $this->assertEquals(20, $raiting->calculate(User::fintOrfail(2)));
    }

    public function testingCommonStudentRaiting3()
    {
        $raiting = new Raiting();

        $this->assertEquals(80, $raiting->calculate(User::fintOrfail(3)));
    }

    public function testingCommonStudentRaiting4()
    {
        $raiting = new Raiting();

        $this->assertEquals(50, $raiting->calculate(User::fintOrfail(4)));
    }
}
```

Рисунок 3.13. Unit тести для розрахунку рейтингу студента

На даному рисунку ми можемо побачити 4 unit тести. Назва кожного методу unit тесту є унікальною. Прочитавши назву, можна зрозуміти, що перевіряється в даному тесті. Так, в першому тесті ми передаємо інформацію про студента, який тільки почав навчання та має невеликий рейтинг. В даному методі

ми викликаємо метод бібліотеки *PHPUnit assertEquals*. Даний метод бібліотеки приймає на вхід два параметри: значення, яке очікується отримати після роботи методу та значення, яке початково передається в даний метод. Після того, як виконується тест в метод *assertEquals* викликає переданий другим параметром метод із вхідними даними і після його виконання він звіряє отримані результати з тим, що переданий першим параметром і якщо вони рівні, то тест вважається успішно виконаним і метод працює правильно. Аналогічно і інші 3 тести, тільки на вхід методу розрахунку рейтингу передаються інформація про іншого студента.

В результаті виконання тестів отримано наступний результат(рис. 3.14).

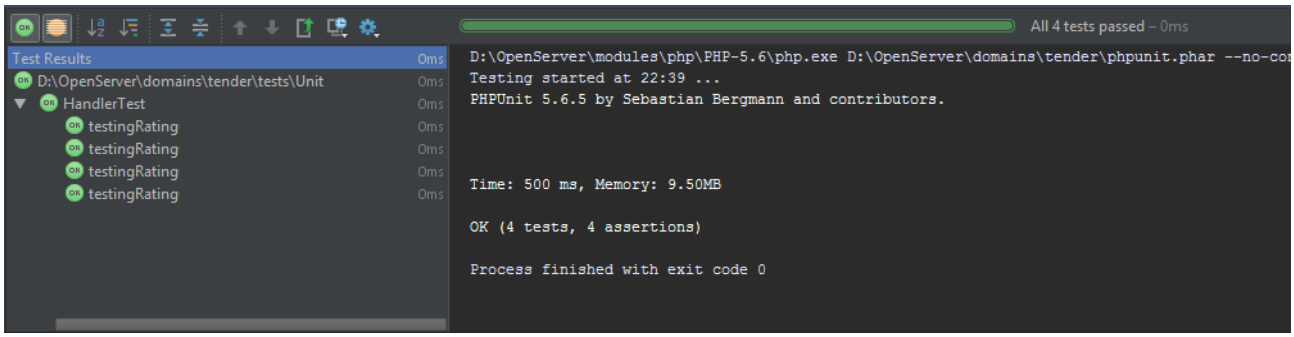

Рисунок 3.14 – Результат виконання unit тестів

Всі тести успішно виконалися. В правому вікні в нас відображається перелік тестів та статус їх виконання. В лівому вікні після виконання тестів відображається загальна інформація, така як: час виконання, об’єм використаною пам’яті, яка кількість тестів виконана. В тому разі, коли ми отримуємо у роботі методу помилку і тест вважається невиконаним, в лівому вікні поряд з назвою тесту з’являється знак оклику в оранжевому колі. При натисненні на невиконаний тест в лівому вікні з’явиться інформація про конкретний тест. Як правило, там пишеться причина невиконання тесту, значення, яке очікувалось отримати та значення, яке отримали в результаті виконання методу певного класу, а також відображається посилання на отримання відмінностей. Таким чином, якщо ми змінимо певне значення в написаних тестах на неправильне, ми отримаємо наступний результат (рис 3.15)

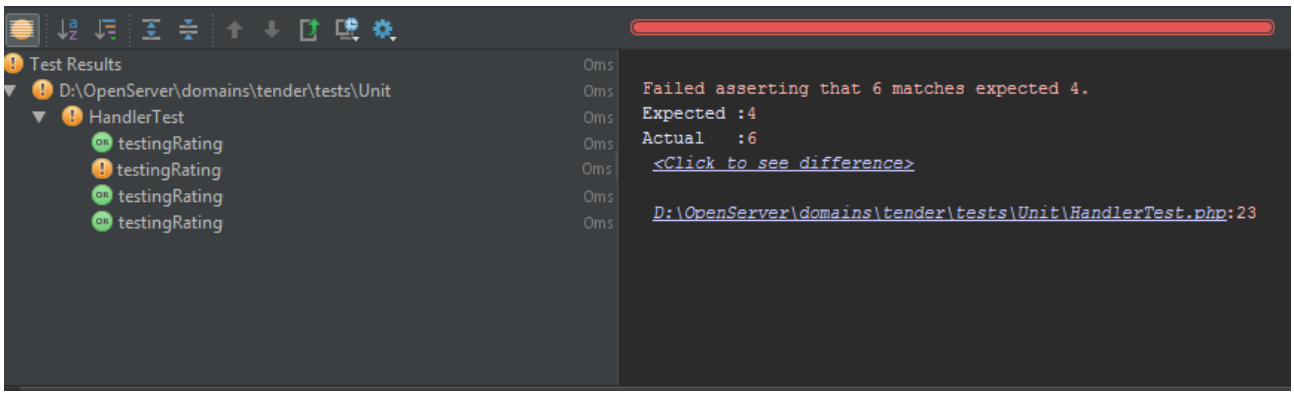

Рисунок 3.15. Інформація про невиконаний тест

Аналогічні тести були написані для всіх інших методів веб-орієнтованої системи для навчання студентів та керування навчальним процесом та були виконані. Методи, для яких тести не виконались, були перевірені на наявність помилок та було здійсненне їх виправлення та повторне виконання тестів. Така процедура виконувала для всіх методів, поки не було отримано потрібних результатів.

3.10. Масштабування серверів за допомогою Amazon EC2

Кількість користувачів інтернету є доволі великою, тому в наш час використання одного серверу для веб-додатку є недостатнім. Якщо ж говорити про систему для навчання студентів та керування навчальним процесом, якою будуть користуватися велика кількість студентів, то тим більше потрібно використовувати декілька серверів при збільшенні навантаження на систему. Amazon EC2 дозволяє в досить легкий спосіб здійснити налаштування серверів та виконати їх масштабування. Для створення нового серверу потрібно виконати декілька простих кроків – обрати конфігурацію серверу та обрати тип серверу. Ми можемо обрати програми, які нам потрібно, щоб були встановлені на даному сервері та його операційну систему. Amazon має заздалегідь заготовлені типи серверів, які мають певну кількість оперативної пам’яті, різні типи процесорів, величину оперативної та звичайної пам’яті та інші параметри. Це дозволяє у зручному веб-інтерфейсі в два кроки створити новий сервер. Після створення

серверу ви можете змінювати його конфігурацію та переглядати його основні параметри (рис. 3.16).

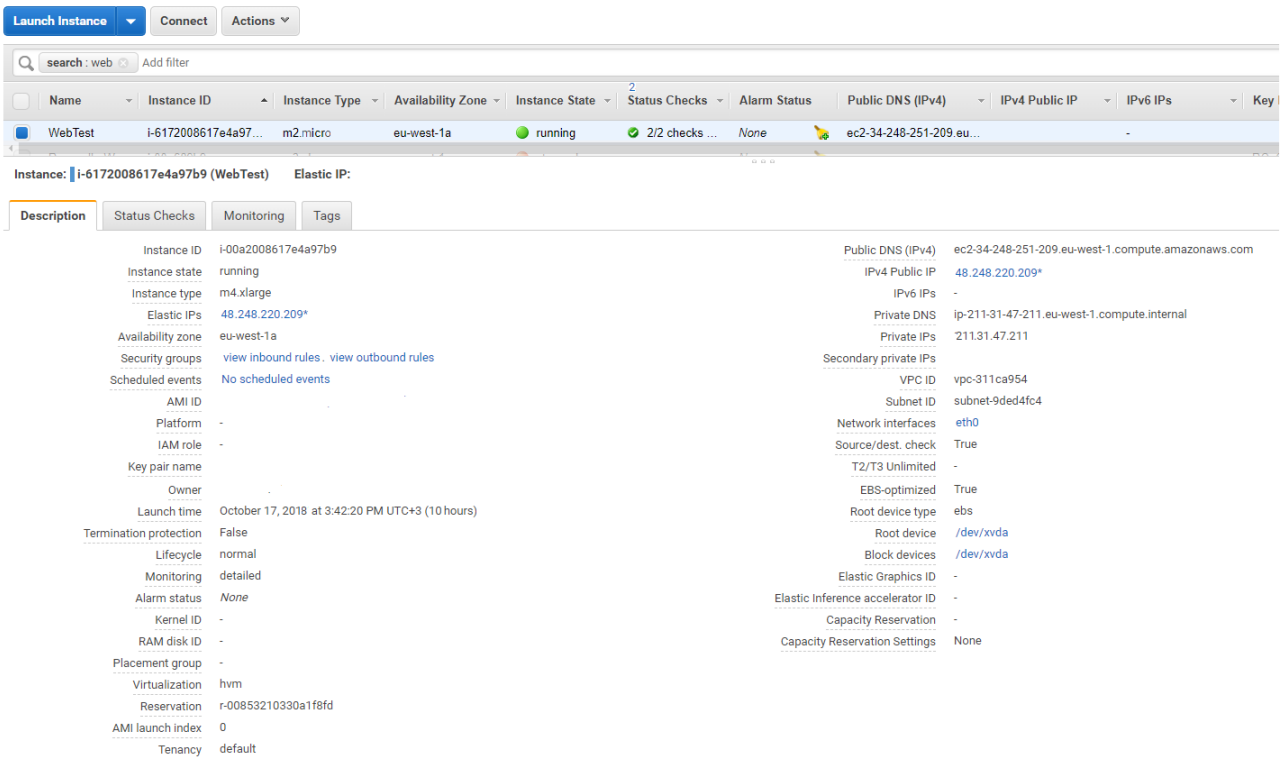

Рисунок 3.16. Розгорнутий сервер на Amazon

Після створення серверу потрібно налаштувати групу безпеки для нього. Вона являє собою набір правил, які визначають хто і яким чином може здійснити підключення до серверу та куди може здійснити підключення власне сам сервер. Група безпеки складається з набору правил, які визначають протокол, за яким можна здійснити ту чи іншу дію, діапазон дозволений IP адрес, з яких можна здійснити певну дію та тип підключення – для вхідного чи вихідного потоку застосовувати дане правило. Для нашого серверу ми обрали програмне забезпечення з використання операційної системи CentOS, в якості типу серверу використовується один із можливих типів Amazon EC2. Створена нова група безпеки, яка дозволяє підключатися до серверу по SSH тільки з визначеної нами IP адреси. Сервер та власне веб-додаток доступні всім користувачам через протокол HTTPS. Для забезпечення масштабування серверів, зі створеного серверу зроблено копію та додано в групу масштабування та систему, які здійснює балансування навантаження. Виставлено правило, що при зростанні

навантаження на сервер більш ніж на 70%, виконується автоматичне підняття ще одного серверу для розподілу навантаження.

3.11. Аналіз отриманих результатів та швидкодії системи

Проведемо аналіз швидкодії системи, проаналізувавши час завантаження різних її сторінок, використовуючи консоль розробника браузеру.

Час завантаження головної сторінки становить 1-2 секунди, що я досить хорошим показником(рис. 3.17)

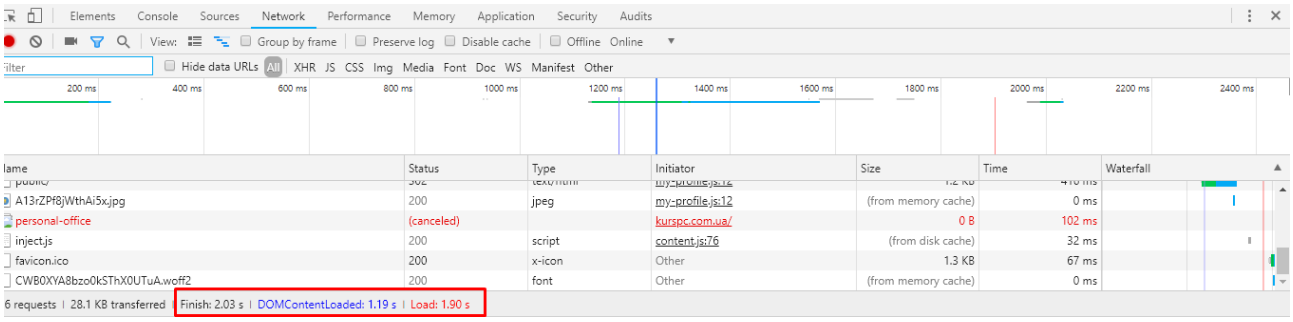

Рисунок 3.17. Час завантаження головної сторінки

Час завантаження сторінки, яка містить методичні матеріали для студентів, які навчаються(рис. 3.18)

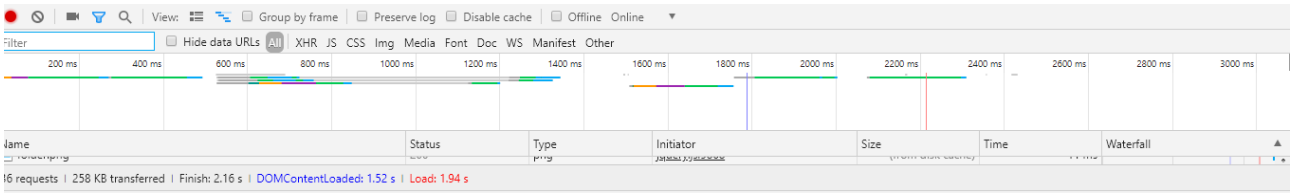

Рисунок 3.18. Час завантаження сторінки із методичним матеріалами

Маємо аналогічний час завантаження для сторінок із переліком всіх студентів, які на момент тестування системи налічувалося 255 чоловік(рис. 3.19)

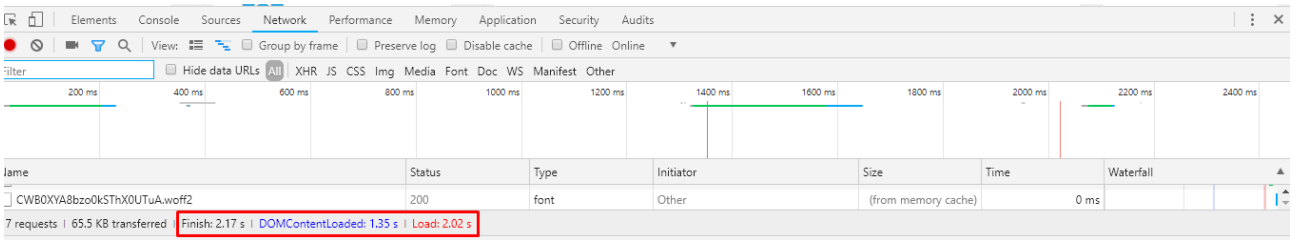

Рисунок 3.19. Час завантаження сторінки зі списком всіх студентів

Час завантаження сторінки, де відображається успішність студентів та перелік всіх уроків обраної групи(рис. 3.20)

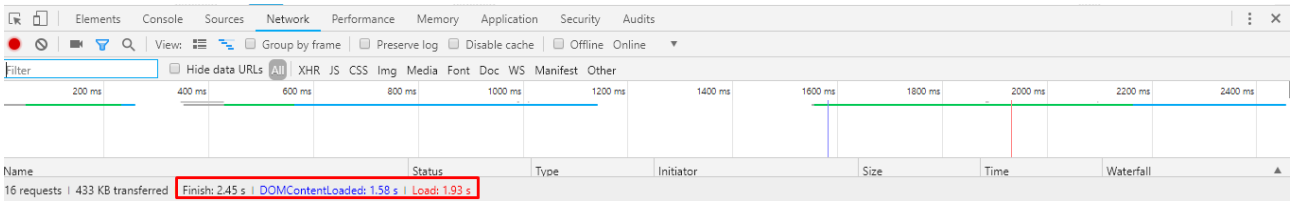

Рисунок 3.20. Час завантаження сторінки успішності студентів

Отже, проаналізувавши всі отримані результати, можна зробити висновок, що система має належну швидкодію та завдяки цьому є дуже зручною у користуванні.

Висновки до розділу

В даному розділі було здійснено опис архітектури системи для навчання студентів та керування навчальним процесом. Визначено основні модулі системи ти проведено їх детальний опис. Створено структуру бази даних. Розроблено моделі для роботи з базою даних, визначено властивості кожної моделі та сформовано зв'язки між моделями аналогічно до зв'язків між таблицями в базі даних. Проведено аналіз системи, визначено найчастіше виконувані запити в системі та створено індекси для даних полів бази даних. Розроблено RESTful API для роботи даної системи. Визначено правила валідації даних та розроблено механізм перехоплення помилок в системі та відображення їх адміністратору. Написано логіку роботи основних модулів системи. Налаштовано систему Redis для кешування даних та реалізовано логіку для роботи з неї в системі. Налаштовано розклад для запуску команд утилітою cron та написано логіку роботи команд таким чином, щоб не виникало помилок та колізій при зверненні до бази даних. Налаштовано сервер для роботи веб-додатку та забезпечено його належну масштабованість.

РОЗДІЛ 4. МАРКЕТИНГОВИЙ АНАЛІЗ СТАРТАП-ПРОЕКТУ

Проведемо маркетинговий аналіз стартап проекту для визначення можливості його ринкового впровадження.

4.1. Опис ідеї проекту

Головна ідея даного стартап проекту полягає в розробці простої та надійної системи для навчання студентів та керування навчальним процесом. Важливим фактором даної системи є зручність у користуванні та належна її швидкодія. Завдяки всім цим особливостям дана система повинна витримувати значні навантаження та велику кількість користувачів, які одночасно користуються нею та дозволяти адміністраторам та викладачам вести облік студентів, а студентам переглядати відео та навчатися онлайн.

Розглянемо детальніше зміст ідеї, основні напрямки застосування та вигоди користувачів, які вони можуть отримують(табл. 4.1).

Таблиця 4.1 Опис ідеї стартап проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Розробити систему для навчання студентів та керування навчальним процесом	Використання веб-студіями для ведення обліку студентів, груп, викладачів. Виставлення оцінок	Веб-студії та начальні заклади отримують можливість здійснювати керування навчальним процесом за допомогою зручного інтерфейсу браузера та автоматизувати свою роботу за рахунок даної системи
	Використання студентами для навчання та перегляду корисних матеріалів	Студенти отримують можливість навчатися онлайн, мають постійний доступ до навчальних матеріалів та можуть переглядати свої результати навчання з того чи іншого предмету

Визначимо техніко-економічні переваги, які має даних додаток в порівнянні із вже існуючими рішення. В якості прямих аналогів до нашого додатку оберемо систему для навчання студентів ITVDN та Main Academy. Визначимо сильні, слабкі та нейтральні характеристики нашого додатку в порівнянні з обраними конкурентами(табл. 4.2).

Таблиця 4.2 Визначення сильних, слабких та нейтральних характеристик ідеї проекту

№	Техніко-економічні характеристики ідеї	Продукція конкурентів			W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
		Мій проект	ITVDN	Main Academy			
1	Сучасний дизайн	+	+	+		+	
2	Зручність у користуванні	+	-	+		+	
3	Швидкодія роботи	+	-	-			Має належну швидкодію роботи.
4	Можливість використання методичних матеріалів	+	-	-		+	+
5	Формування груп та уроків в системі	+	-	-			Надає можливість викладачу та адміністратору формувати групи та уроки
6	Виставлення оцінок та формування рейтингу групи	+	-	-			Кожний студент отримує певну оцінку за кожний урок і може переглядати свій сумарний бал
7	Відсутність нагромадження інформації на сторінці	+	-	-			+
8	Масштабування системи	+	+	+		+	

Визначений перелік слабких, сильних та нейтральних характеристик та властивостей ідеї потенційного товару є підгрунтям для формування його конкурентоспроможності. Із отриманих результатів можна зробити висновок, що даний стартап проект має переваги порівняно із аналогами.

4.2. Технологічний аудит ідеї проекту

Проведемо аудит технологій, за допомогою яких можна реалізувати ідею проекту(табл. 4.3)

Таблиця 4.3 Технології здійснення ідеї проекту

№	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
1	Розробка системи для навчання студентів та керування навчальним процесом і забезпечення її належної швидкодії	Використання мови програмування PHP	Наявна. Має велику кількість готових рішень для вирішення різних проблем. Має добре продумані фреймворки, які забезпечують належну безпеку веб-додатку та є добре розширювані.	Вільна
		Використання мови програмування Java	Наявна. Має один фреймворк для роботи з веб-додатками	Вільна
		Використання мови програмування C#	Наявна. Має один фреймворк для роботи з веб-додатками	Вільна
		Використання мови програмування Python	Наявна. Має один фреймворк для роботи з веб-додатками	Вільна
Обрана технологія реалізації проекту: PHP.				

Проаналізувавши інформацію про технології, які можуть використовувати для розробки веб-додатку для навчання студентів та керування навчальним процесом, можна зробити висновок, що найкраще підходить мова програмування PHP, оскільки вона має велику кількість стандартних рішень для вирішення тривіальних задач веб-додатків, має велику кількість фреймворків та систем для керування контентом сайту, які дозволяють для кожного конкретного веб-додатку обрати систему, яка найкраще підходить для вирішення його основних задач. В якості фреймворку будемо використовувати Laravel. Він має добре продуману модульну систему, має повноцінну документацію та велику кількість розробників, які його використовують. Все це дозволяє легко знайти

вирішення тієї чи іншої проблеми та розібратися із системою новому розробнику, який її не розробляв.

4.3. Аналіз ринкових можливостей запуску стартап-проекту

Визначимо ринкові можливості, які можна використати під час ринкового впровадження проекту, та ринкові загрози, які можуть перешкодити реалізації проекту.

Таблиця 4.4. Попередня характеристика потенційного ринку стартап проекту

№	Показники стану ринку	Характеристика
1	Кількість головних гравців, од	10
2	Загальний обсяг продаж, грн./ум.од	3000 в рік
3	Динаміка ринку	Зростає
4	Наявність обмежень	Відсутні
5	Специфічні вимоги до стандартизації та сертифікації	Відсутні
6	Середня норма рентабельності в галузі або по ринку, %	100%

Спираючись на проведене дослідження можна зробити висновок, що ринок є привабливим для входження та реалізації запланованої ідеї.

Визначимо потенційних клієнтів даного стартап проекту(табл. 4.5.)

Таблиця 4.5. Характеристика потенційних клієнтів стартап проекту

№	Потреба, що формує ринок	Цільова аудиторів(цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1	Наявність системи для навчання студентів та автоматизація процесу навчання студентів веб-студії чи університету	Цільовою аудиторією є веб-студії та навчальні заклади, які проводять навчання студентів за різними напрямами та бажають автоматизувати процес та завжди мати під рукою всі потрібні дані про студентів та навчальний процес.	Веб-студії та навчальні заклади в основному однаково зацікавлені у використанні даного веб-додатку, оскільки він дозволяє автоматизувати їх роботу	Головною вимогою є вміння користуватися сучасним веб-додатками та доступ до мережі інтернет

Провівши аналіз потенційних клієнтів стартап проекту, можна зробити однозначний висновок, що дана систему є особливо привабливою для веб-студій та навчальних закладів. Нею можуть зацікавитися також і велику університети, але в такому разі вона потребує доопрацювань та розширення базового функціоналу.

Здійснимо аналіз ринкового середовища. Спочатку визначимо основні фактори загроз, які можуть вплинути на продажі розробленої системи(табл. 4.6)

Таблиця 4.6 Фактори загроз

№	Фактор	Зміст загрози	Можлива реакція компанії
1	Сповільнення роботи системи при великій кількісті даних	Система працює досить швидко та користувачу не доводиться витрачати багато часу для отримання певних даних. При наповненні системи даними та зберіганні в системі великої кількості даних можливе сповільнення роботи системи та потреба у її вдосконаленні	Звернення до розробників, які підтримують систему, перенесення даних, що не використовують в іншу спеціальні таблиці або їх видалення, якщо вони непотрібні
2	Можливість виникнення певних помилок в систему	Система має певну зв’язність між різними модулями, тому для виконання певних дій потрібно спочатку створити відповідні дані.	Покращення документації щодо користування системою та підвищення рівня вмінь адміністратора системи

Розглянемо основні фактори можливостей, які можуть очікувати додаток при виході на ринок(табл. 4.7)

Таблиця 4.7 Фактори можливостей

№	Фактор	Зміст можливості	Можлива реакція компанії
1	Автоматизація роботи навчального закладу	За рахунок використання системи зменшується час потрібний на формування груп, класів, рейтингу студентів та виконання інших дії. Система дозволяє значно оптимізувати навчальний процес	Підвищення кваліфікації викладачів у користуванні даною системою
2	Виконання масової розсилки повідомлень	Система дозволяє здійснювати надсилання повідомлень конкретним користувачам чи групам користувачів	Підвищення кваліфікації адміністраторів системи у користуванні системою та запуска масової розсилки

На наступному кроці виконаємо ступеневий аналіз конкуренції(табл. 4.8)

Таблиця 4.8 Ступеневий аналіз конкуренції на ринку

№	Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства(можливі дії компанії, щоб бути конкурентноспроможною)
1	Тип конкуренції: чиста конкуренція	Існує велика кількість конкурентів, які пропонують розробку системи даного типу	Реклама, підвищення швидкодії, якості та зручності у користуванні системою
2	Рівень конкурентної боротьби: національний	Рішення може використовуватися в навчальних закладах та веб-студіях на території України	Розширення функціональних можливостей системи та її доопрацювання.
3	Галузева ознака: внутрішньогалузевий	Конкуренція в сфері навчання студентів та автоматизації навчального процесу	Покращення якості та зручності в роботі системи та її функціональне розширення
4	Конкуренція за видами товарів: товарно-видова	Дана конкуренція - конкуренція між товарами одного виду	Впровадження функціональності, яка відсутня у додатках інших розробників. Спрощення в роботі додатку та забезпечення його належної якості.
5	Характер конкурентних переваг: цінова	Співвідношення вартості системи до її надійності та якості	Продаж системи за адекватні кошти та забезпечення належної підтримки
6	За інтенсивністю: марочна	Наявність унікального зразка, який відрізняє даних продукт від продуктів-замінників	Впровадження власної назви та власного знаку.

Таблиця 4.9 Аналіз конкуренції в галузі за М. Портером

Складові	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товарозамінники
	ITVDN, Main Academy	Інші дистриб’ютори	Відсутні	Веб-студії та навчальні заклади	Відсутні
Висновки	Необхідно надавати унікальний фунціонал, забезпечити належну швидкодію та зручність у користуванні системою	При розширенні та доопрацюванні функціональності системи може стати конкурентом.	Відсутні	Кожний клієнт має певні побажання, які доцільно враховувати при розробці системи	Відсутні

Спираючись на проведені дослідження, можна зробити висновок, що конкуренція існує, проте кожний із відомих додатків має певні недоліки, тому для впровадження системи та забезпечення значних продаж, її потрібно розробити таким чином, щоб вона максимально задовольняла потреби користувачів, була зручною у користуванні та мала належну швидкодію.

Таблиця 4.10 Обгрунтування факторів конкурентоспроможності

№	Фактор конкурентоспроможності	Обгрунтування
1	Автоматизація навчального процесу	За рахунок автоматизації навчального процесу відбувається зменшення часу необхідного для ведення обліку груп, студентів та інших даних певного навчального закладу
2	Належна швидкодія системи	Система працює швидше за своїх конкурентів та забезпечує швидке відображення потрібної користувачу інформації
3	Масштабування системи	Система може бути легко масштабована та витримувати значне навантаження та велику кількість користувачів, які одночасно нею користуються

Таблиця 4.11 Порівняльний аналіз сильних та слабких сторін системи

№	Фактор конкуренто-спроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні з новою системою						
			-3	-2	-1	0	+1	+2	+3
1	Автоматизація навчального процесу	10			+				
2	Належна швидкодія системи	15		+					
3	Масштабування системи	10			+				

На основі отриманих даних, сформуємо SWOT – аналіз(матриці аналізу сильних Strength та слабких Weak сторін, загроз Troubles та можливостей Opportunities на основі виділених ринкових загроз та можливостей, а також сильних та слабких сторін). Перелік ринкових загроз та можливостей складається на основі аналізу факторів загроз та факторів можливостей маркетингового середовища. Ринкові загрози та можливості є наслідками впливу факторів.

Таблиця 4.12. SWOT-аналіз стартап-проекту

Сильні сторони (S): • належна швидкодія • зручність у користуванні • масштабованість	Слабкі сторони(W): • вартість розробки • вартість обслуговування
Можливості(O): • розширення функціоналу • покращення роботи системи	Загрози(T): • недостатнє фінансування • поява конкурентів

На основі SWOT-аналізу розробимо альтернативи ринкової поведінки для забезпечення виведення стартап проекту на ринок та орієнтовний оптимальний час їх ринкової реалізації з огляду на потенційні проекти конкурентів, що можуть бути також виведені на ринок. Визначення альтернатив здійснюється з точки зору строків та ймовірності отримання оптимальних ресурсів

Таблиця 4.13 Альтернативи ринкового впровадження стартап-проекту

№	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Безкоштовне надання певного функціоналу у користування споживачам на обмежений термін	Головний ресурс – люди, даний ресурс - наявний	1-3 місяці
2	Реклама	Залучення власних коштів для реклами товару	2-3 місяці
3	Написання статей та опис товару на відомих ресурсах	Головний ресурс – час, даний ресурс - наявний	2-3 тижні
4	Презентація товару на конференціях та інших ІТ заходах	Ресурс – час та гроші для участі, наявні	1 місяць

В результаті проведено аналізу, можна зробити висновок, що найбільш привабливою альтернативою для ринкового впровадження стартап проекту, є презентація товару на конференціях та інших ІТ заходах.

4.4. Розроблення ринкової стратегії проекту

Розроблення ринкової стратегії першим кроком передбачає визначення стратегії охоплення ринку.

Таблиця 4.14. Вибір цільових груп потенційних споживачів

№	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Програмісти. Вік: від 18 до 40. Місце проживання: не важливо. Сімейний стан: не важливий. Сфера зайнятості та рівень заробітної плати: IT сфера, від 30 тис. грн.	Даний продукт можна використовувати як засіб автоматизації роботи навчального закладу	1 ліцензія для навчального закладу чи веб-студії	Інтенсивність конкуренції в сегменті висока, оскільки існує великий попит систем даного типу	Сегмент дозволяє вийти на ринок та показати переваги даного продукту у контексті продуктів-аналогів
2	Підприємства. Сфера зайнятості – навчання студентів.	Система дозволяє автоматизувати роботу навчального закладу або веб-студії та забезпечити хороший механізм для навчання студентів.	1 ліцензія для навчального закладу або веб-студії	Інтенсивність конкуренції в сегменті висока, оскільки існує велика кількість виробників схожих продуктів.	Сегмент дозволяє вийти на ринок та показати переваги даного продукту у контексті продуктів-аналогів
Які цільові групи обрано: IT сфера та навчальні заклади, у яких виникає необхідність в автоматизації навчального процесу та керуванням ним.					

Оскільки цільовою групою виступають навчальні заклади та розробники різних сфер, оберемо стратегію масового маркетингу.

Таблиця 4.15 Визначення базової стратегії розвитку

Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
Надання функціональності що відсутня у товарів-замінників, підтримка клієнтів	Проведення реклами, освітлення унікальної функціональності через інтернет ресурси та інші канали, контакт напряму з споживачами; формування лояльності і прихильності споживачів	Зниження ступеню замінності товару; Прихильність клієнтів; Відмітні властивості товару; Відмітні характеристики товару;	Стратегія диференціації

Таблиця 4.16 Визначення базової стратегії конкурентної поведінки

Чи є проект «першопрохідцем» на ринку	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, які?	Стратегія конкурентної поведінки
Ні, оскільки є товари-замінники, але дані товари замінники не мають деякого необхідного функціоналу	Так, ціль компанії знайти нових споживачів та, частково, переманити існуючих у конкурентів задля задоволення потреб останніх	Компанія частково копіює характеристики товару конкурента, основна ціль компанії розробка нового унікального функціоналу, з підтримкою основного функціоналу конкурентів	Стратегія заняття конкурентної ніші

Таблиця 4.17 Визначення стратегії позиціонування

№	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту
1	Відмінні властивості товару	Стратегія диференціації	Унікальність функціоналу;	Надійність Актуальність
2	Підтримка з боку розробника	Стратегія диференціації	Підтримка клієнту; Ліцензії.	Клієнтоорієнтованість
3	Відповідність загальновживаним інтерфейсам	Стратегія диференціації	Підтримка та вдосконалення сучасних всім знайомих методів	Стабільність Зручність

Тобто, у якості базової стратегії розвитку було обрано стратегію диференціації та стратегію заняття конкурентної ніші, яка має базову стратегію конкурентної поведінки.

4.5. Розроблення маркетингової програми стартап-проекту

Таблиця 4.18 Визначення ключових переваг концепції потенційного товару

№	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)
1	Автоматизація роботи веб-студії або навчального закладу	Підвищення ефективності роботи веб-студії або навчального закладу за рахунок автоматизації навчального процесу	Існуючі конкуренти мають меншу швидкодію системи та обмежений функціонал
2	Можливість навчання студентів та перегляду відеоматеріалів	Студенти, які з певних причин були відсутні на заняттях мають можливість переглянути навчальний матеріал онлайн	Існуючі конкуренти не надаються повноцінний доступ до навчальних матеріалів, таких як: програмний код, розробки зроблені на занятті, відеоуроки

Здійснимо опис рівнів моделі товару

Таблиця 4.19 Опис трьох рівнів моделі товару

<table>
<tr><td>Рівні товару</td><td colspan="3">Сутність та складові</td></tr>
<tr><td>1. Товар за задумом</td><td colspan="3">Забезпечення автоматизації роботи веб-студії чи навчального закладу та надання користувачам можливості навчатися онлайн</td></tr>
<tr><td rowspan="7">2. Товар у реальному виконанні</td><td>Властивості/характеристики</td><td>М/Нм</td><td>Вр/Тх/Тл/Е/Ор</td></tr>
<tr><td>Зручність використання</td><td>-</td><td>Висока</td></tr>
<tr><td>Забезпечення швидкої роботи та обробки даних</td><td>Час виконання запитів</td><td></td></tr>
<tr><td>Вартість</td><td>Грн.</td><td>0</td></tr>
<tr><td colspan="3">Якість: відповідність загальновживаним нормам</td></tr>
<tr><td colspan="3">Пакування: ліцензія на використання системи</td></tr>
<tr><td colspan="3">Марка: Contact</td></tr>
<tr><td rowspan="2">3. Товар із підкріпленням</td><td colspan="3">До продажу: наявна повна документація, акції на придбання декількох ліцензій</td></tr>
<tr><td colspan="3">Після продажу: підтримка та доопрацювання продукту в разі необхідності та бажання замовника</td></tr>
<tr><td colspan="4">Проект буде захищено від копіювання за рахунок реєстрації назви програми, отримання патенту на програмний код даного продукту та окремих алгоритмів обробки даних</td></tr>
</table>

Таблиця 4.20 Визначення меж встановлення ціни

Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
інформація відсутня	приблизно 60 тис. грн..	від 50 тис. грн.	30-50 тис. грн

Таблиця 4.21 Формування системи збуту

Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
Отримання продукту	Ліцензії	Нульовий та/або однорівневий	Традиційна

Таблиця 4.22 Концепція маркетингових комунікацій

№	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
1	Отримання продукту / функціональності	Електронна пошта, телефон, факс, електронна форма на сайті	Унікальні властивості товару	Привернути увагу клієнтів, освітити унікальну функціональність	Творча та класична
2	Отримання підтримки від компанії		Якісна підтримка	Дати зрозуміти що клієнт може розраховувати на підтримку зі сторони розробника	

В результаті було створено ринкову програму, що включає в себе визначення ключових переваг концепції потенційного товару, опис моделі товару, визначення меж встановлення ціни, формування системи збуту та концепцію маркетингових комунікацій.

Висновки до розділу

В даному розділі було описано ідею проекту, здійснено аналіз потенційних техніко економічних переваг ідеї, а саме визначено сильні, нейтральні та слабкі

сторони даної ідеї. Проведено аудит описаної ідеї та визначення її технологічної здійсненності. На основі отриманих результатів було зроблено висновок, що дана ідея є здійсненною. Виконано попередню оцінку потенційного ринку стартап проекту, визначено потенційних клієнтів даного продукту, визначено основні фактори загроз та можливостей. Проведено ступеневий аналіз конкуренції на ринку, обгрунтовано фактори конкурентоспроможності та виконано порівняльний аналіз сильних та слабких сторін продукту та конкурентів. Проведено SWOT-аналіз та альтернативні шляхи впровадження даного продукту. На основі отриманих результатів можна зробити висновок, що даний проект може бути впроваджений та зайняти свою нішу.

ВИСНОВКИ

На першому етапі визначено мету, предмет та об’єкт дослідження, а також область, в якій воно проводиться. Проведено аналіз існуючих систем та визначено їх основні переваги та недолік. Розглянуто дві системи: ITVDN та Main Academy. Виконано аналіз та вибір технологій, які будуть задовольняти поставлені вимоги до системи. Описано обрані технології та їх основні особливості.

Визначено основні модулі системи для навчання студентів та керування навчальним процесом. Розроблено структуру бази даних, яка складається із 14 таблиць, та створено індекси для полів таблиць, за якими найчастіше відбувається пошук даних для забезпечення належної швидкодії системи.

Створено моделі для роботи з базою даних за допомогою ORM та описано всі зв’язки між таблицями. Розроблено RESTful API для обміну даними між бекендом та фронтендом. Створено механізм чіткої валідації всіх даних, які ми отримуємо від користувача, а також систему перехоплення помилок та їх логування в файли. Впроваджено систему кешування Redis, яка дозволяє здійснювати збереження найчастіше запитуваних даних у вигляді ключ-значення. Зберігання даних відбувається в оперативну пам’ять і за рахунок цього забезпечується висока швидкодія при записуванні та отриманні потрібних даних.

Розроблено механізм виконання складних та трудомістких задач у фоновому режимі за допомогою утиліти cron операційної системи Linux. Дана утиліта дозволяє в автоматичному режимів відповідно до налаштованого розкладу виконувати запуск будь-яких команд для виконання певних завдань.

Розгорнуто систему на сервері Amazon та створено базовий образ серверу для подальшого налаштування системи, яка здатна балансувати навантаження на сервер та в разі необхідності здійснювати в автоматичному режимі підняття додаткових серверів та розподілу навантаження на систему між ними.

В результаті виконання даної магістерської дисертації було проведено оптимізацію та підвищено швидкодію системи для навчання та керування навчальним процесом.

ПЕРЕЛІК ПОСИЛАНЬ

1. Роберт Никонс. Создаем динамические веб-сайти с помощью PHP, MySQL, JavaScript, CSS и HTML5. 3-е изд. – СПб.: Питер, 2015. – 688с
2. Котеров Д.В. PHP7/ Д.В. Котеров, И.В. Симдянов. – СПб.: БХВ-Петербург, 2016. – 1088с
3. Мова розмітки HTML [Електронний ресурс]: https://uk.wikipedia.org/wiki/HTML
4. Мова стилів CSS [Електронний ресурс]: https://uk.wikipedia.org/wiki/CSS
5. Мова програмування Javascript [Електронний ресурс]: https://uk.wikipedia.org/wiki/JavaScript
6. Документація мови програмування JavaScript [Електронний ресурс]: https://learn.javascript.ru/
7. Документація Bootstrap [Електронний ресурс] : http://getbootstrap.com/
8. Мова програмування PHP [Електронний ресурс] : https://uk.wikipedia.org/wiki/PHP
9. Офіційна документація PHP [Електронний ресурс] : http://php.net/
10. Офіційна документація по фреймворку Laravel [Електронний ресурс] : https://laravel.com/
11. Документація MySQL [Електронний ресурс] : https://www.mysql.com/
12. Документація по фреймворку Laravel [Електронний ресурс] : https://laravel.com/
13. Опис технології RESTfull [Електронний ресурс] : https://uk.wikipedia.org/wiki/REST
14. Стандарти мови програмування PHP щодо написання коду [Електронний ресурс] : http://www.php-fig.org/psr/

ДОДАТКИ

ДОДАТОК А

Стаття

УДК 004.4

Сороченко Руслан Андрійович
Студент, НТУУ “КПІ ім. Ігоря Сікорського”
Крилов Євген Володимирович
К.т.н., доцент, НТУУ “КПІ ім. Ігоря Сікорського”
Анікін Володимир Костянтинович
Старший викладач, НТУУ “КПІ ім. Ігоря Сікорського”
Київ, Україна

ВИКОРИСТАННЯ УТИЛІТИ CRON ДЛЯ ВИКОНАННЯ СКЛАДНИХ ТА ТРУДОМІСТКИХ ЗАДАЧ ВЕБ-ДОДАТКІВ

Анотація. В даній статті розглядається утиліта cron операційної системи Linux, яка дозволяє запускати команди та виконувати певну логіку в автоматичному режимі відповідно до файлу з розкладом. Основні параметри та складові файлу з розкладом, який використовує дана утиліта. Її переваги та задачі, для яких її доцільно використовувати.

Ключові слова: cron, crontab, Linux, PHP, веб-додаток, операційна система, утиліта.

Sorochenko Ruslan A.
Student, NTUU “Igor Sikorsky Kyiv Polytechnic Institute”
Krylov Eugene V.
Ph. D., associate professor, NTUU “Igor Sikorsky Kyiv Polytechnic Institute”
Anikin Vladimir K.
Senior lecturer, NTUU “Igor Sikorsky Kyiv Polytechnic Institute”
Kiev, Ukraine

CRON UTILITY FOR COMPLITING COMPLEX TASKS OF WEB APPLICATIONS

Abstract. We discuss the Linux operation system cron utility in this article. It allows us to run commands and execute certain logic in an automatic mode according to a schedule file. That utility has the main parameters and components that it uses for running commands. We look at its main benefits and tasks that it can do.

Keywords: cron, crontab, Linux, PHP, operation system, web application, utility

Сороченко Руслан Андреевич
Студент, НТУУ “КПИ им. Игоря Сикорского”
Крылов Евгений Владимирович
К.т.н., доцент, НТУУ “КПИ им. Игоря Сикорского”
Аникин Владимир Константинович
Старший преподаватель, НТУУ “КПИ им. Игоря Сикорского”
Киев, Украина

ИСПОЛЬЗОВАНИЕ ПРОГРАММЫ CRON ДЛЯ ВЫПОЛНЕНИЯ СЛОЖНЫХ И ТРУДОЕМКИХ ЗАДАЧ ВЕБ-ПРИЛОЖЕНИЙ

Аннотация. В данной статье рассматривается программа cron операционной системы Linux, которая позволяет запускать команды и выполнять определенную логику в автоматическом режиме в соответствии с файлом, в котором хранится расписание. Основные параметры та компоненты файла расписания, который использует данная утилита. Ее основные преимущества и задачи, для которых ее целесообразно использовать.

Ключевые слова: cron, crontab, Linux, PHP, веб-приложение, операционная система, утилита.

Вступ

Ми живемо в час, коли технології розвиваються досить швидкими темпами. Якщо раніше для розвитку та удосконалення тієї чи іншої технології потрібні були декілька років, то тепер для цього потрібно всього лише один рік або навіть менше часу. Однією із найбільш популярних технологій є мережа Інтернет. Кількість користувачів з кожним роком невпинно зростає, кабелі для доступу до даної мережі почали прокладати в найвіддаленіші куточки світу або встановлювати спеціальні станції, які здатні надавати доступ бездротово. Розглядаючи дану технології з точки зору користувача, її основою є веб-додатки, які дозволяють користувачу переглянути певну інформацію та знайти потрібні йому матеріали. Відповідно до розвитку даної технології постійно зростають вимоги до веб-додатків та дуже швидкими темпами розвиваються технології, які використовуються для їх розробки. Раніше сайти складалися з декількох форм вводу інформації або певного меню з посиланнями, при натискання на яке відбувалося перезавантаження сторінки та користувачу доводилося доволі довго чекати, поки відкриється стаття. Форма складалася з прямокутних полів вводу, які не мали жодної валідації даних на фронтенд частині, користувачу доводилося натискати кнопку надсилання форми та чекати, поки він отримає певну відповідь з бекенд частини. Веб-додаток мав мінімальну кількість динамічних елементів або взагалі їх не мав і, як правило, там був досить простий функціонал, який дозволяв користувачам переглядати в ньому певну інформацію або написати адміністратору для уточнення певних даних чи отримання консультації. В наш час сайти представляють собою складні механізми, які мають складну логіку роботи та можуть виконувати велику кількість операцій. Під веб-додатком ми розуміємо систему для перегляду відео Youtube, безліч систем для вирішення певних математичних задач, системи для прокладання маршруту до певного місця, для обміну повідомленнями між користувачами, а також системи для навчання студентів. Всі вони мають складну логіку та виконують складні і

трудомісткі задачі, які можуть виконуватися годину і більше часу. Виконувати такі задачі під час запитів користувачів не є хорошою практикою. В даному випадку значно збільшиться час виконання самого запиту і це відповідно призведе до того, що користувач почне користуватися іншим додатком, адже доведеться чекати більше 3-5 хвилин для завантаження сторінки або виконання інших дій. Кожний веб-додаток має адміністративну частину, яка дозволяє додавати до веб-додатку певну інформацію, створювати нові блоки для відображення інформації користувачу та багато іншого. Доступ до даної частини має тільки адміністратор даного додатку. Виконання складних та трудомістких задач з даної частини додатку не спричинить жодних негативних наслідків для звичайних користувачів даним додатком, але є дуже не зручним для самого адміністратора. Йому доведеться запускати дану команду та чекати, поки вона виконається. Якщо певна задача буде виконуватися годину або дві, то власне адміністратору доведеться чекати одну-дві години. До того ж, він не зможе переглядати стан виконання даної задачі в режимі реального часу. Якщо ми говоримо про розсилку повідомлення 100.000 користувачів, то адміністратору та власнику додатка було б не погано знати, кому вже відправилося повідомлення, чи всім користувачам успішно відправилися повідомлення і кому в даний час ми намагаємося відправити повідомлення. Для вирішення всіх описаних проблем використовується утиліта cron операційної системи Linux, яка дозволяє у зручний спосіб виконувати складні та трудомісткі задачі, а також записувати всі результати надсилання у файл із можливістю їх перегляду.

Утиліта cron операційної системи Linux

Cron - утиліта операційної системи Linux, яка дозволяє запускати та виконувати певні команди чи задачі в автоматичному режимі. Вона досить проста в налаштуванні та управлінні, оскільки для її роботи потрібно лише створити файл із розкладом, за яким вона повинна виконувати ту чи іншу задачу. Для створення файлу з розкладом потрібно виконати команду crontab та вказати ім’я файлу. Після цього всі звернення при виконанні інших команд, будуть

виконуватися до цього файлу, кожний рядок якого складається із трьох складових[1].

Першою складовою є параметри дати та часу, коли повинна запуститися на виконання дана задача. Для задання часу використовується п’ять параметрів, кожний з яких задає значення в хвилинах, годинах, днях місяця, номеру місяця та день неділі. Перший параметр визначає хвилини, коли повинна запуститися команда. Значеннями даного параметру можуть бути відповідно числа від 0 до 59. Щоб команда запускалася кожну хвилину, потрібно вказати символ “*”. Якщо нам потрібно, щоб команда запускалася, наприклад, кожні п’ять хвилин, потрібно задати наступну конструкцію для даного параметру – “*/5”. З даного виразу стає зрозумілим, що запуск задачі на виконання буде виконуватися тільки в момент, коли ми отримаємо ціле число після ділення на 5, тобто команда буде запускатися в 5 хвилин, потім в 10 хвилин, потім в 15 хвилин і тд. Другий параметр відповідає за години, коли повинна запуститися на виконання команда. Для нього ми можемо застосовувати всі вищевказані конструкції. Відповідно третім параметром ми вказуємо дні місяця від 1 до 31, четвертим – номер місяця від 1 до 12. П’ятим значенням вказується дні неділі від 0 до 6, де під нулем розуміється неділя, а під шестіркою – субота. Між кожним параметром ставиться пробіл для їх коректного розпізнавання утилітою. Таким чином, якщо ми хочемо, що наша задача виконувалася кожний день, наприклад, в дві години ночі, ми повинні задати наступне значення – “0 2 * * *”. Якщо розглянути перший та другий параметри, то зрозуміло, що команда повинна виконуватися в 2:00 хвилин. Всі інші параметри задані за допомогою “*”, що означає в кожний день місяця, кожний місяць і кожний день неділі.

Другою складовою вказується, за допомогою якої програми потрібно запустити певну задачу та шлях до файлу, який містить логіку виконання певної задачі. Під виконанням задачі ми розуміємо певний скрип, в якому прописана логіка роботи з базою даних та виконання обчислень. Так, як більшість веб-додатків розроблені за допомогою мови програмування PHP, відповідно запускати скрипти ми будемо за допомогою неї. Операційна система має

спеціально виділену папку, в якій розміщуються файли веб-додатку. Таким чином, другою складовою буде наступна конструкція: $php\ var/www/web - application/tasks/mailing - task.php$[2].

Третя складова є необов'язковою, але, на мою думку, її завжди потрібно вказувати для відслідковування результату виконання тієї чи іншої задачі. В даному випадку ми вказуємо шлях та назву файлу, в який повинні записуватися результати виконання команди, такі як: отримані дані, результати обробки даних, результати виконання певний дій над користувачами, час виконання команди та інші. Вона має наступний вигляд: " >> log/mailing-task.log".

Отже, якщо ми складемо всі три складові, то отримаємо наступну конструкцію: $0\ 2 * * * php\ var/www/web - application/tasks/mailing - task.php \gg log/mailing - task.log$

Кожний рядок файлу з розкладом складається з таких конструкцій, що є досить зручними та простими у використанні.

Дану утиліту доцільно застосовувати для різних трудомістких та інших задач, які повинні виконувати певні дії кожну хвилину чи годину, наприклад, отримання даних з API та їх оновлення.

Розглянемо задачу із виконанням розсилки повідомлень всім користувачам певного додатку. Як вже було описано раніше, виконувати розсилку під час запитів користувачів недоцільно, оскільки в такому випадку час виконання запиту значно збільшиться. Запускати задачу через адміністративну частину сайту зручніше, але в даному випадку доведеться чекати значну кількість часу, поки всім користувачам буде надіслано повідомлення. Найоптимальнішим та найзручнішим варіантом є використання утиліти cron. В даному випадку ми в файлі з розкладом прописуємо, щоб дана задача запускалася кожну хвилину і робила запит до бази даних. В базі даних буде зберігатися поле, яке відповідатиме за виконання розсилки. Якщо в даному полі стоїть значення false, значить не потрібно виконувати ніяких дій, якщо ж в даному полі значення true – можна сміливо запускати надсилання повідомлень. Регулювати значення цього поля може адміністратор із адміністративної частини сайту. Якщо він натиснув

кнопку “Виконати надсилання повідомлень” – поле в базі даних автоматично отримає позитивне значення і після запуску команди через хвилину почнеться надсилання повідомлень із записом даних про результати розсилки. Після завершення розсилки, поле в базі даних автоматично буде встановлене в негативне значення. Таким чином, адміністратору не потрібно запускати та чекати, поки виконається команда, він натиснув кнопку і може далі спокійно займатися іншими справами, а через годину чи більше перевірити результати виконання тієї чи іншої задачі.

Висновок: розглянувши утиліту cron операційної системи Linux, можна помітити, що вона є досить простою в налаштуванні, оскільки для цього потрібно лише створити файл із розкладом. Кожний рядок даного файлу відповідає за виконання певної команди чи задачі. Як тільки наступить час, заданий в даному файлі для певної команди, вона відразу буде запущена. Використовувати дану утиліту доцільно для складних та трудомістких задач, які виконуються годину та більше часу. Також її доцільно використовувати, коли ми повинні кожну хвилину чи годину оновлювати дані веб-додатку, які ми отримуємо з API клієнта.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ ТА ЛІТЕРАТУРИ:

1. Cron[Електронний ресурс] – Режим доступу https://en.wikipedia.org/wiki/Cron

2. What are cron jobs? [Електронний ресурс] – Режим доступу https://support.hostgator.com/articles/cpanel/what-are-cron-jobs

3. Intro to the cron – Unix Geeks[Електронний ресурс] – Режим доступу http://www.unixgeeks.org/security/newbie/unix/cron-1.html

4. Crontab – Quick Reference[Електронний ресурс] – Режим доступу http://www.adminschoice.com/crontab-quick-reference

ДОДАТОК Б

Архітектура системи для навчання студентів та керування навчальним процесом

ДОДАТОК В

Структура бази даних

ДОДАТОК Г

Інтерфейс вікна “Журнал успішності студентів”

ДОДАТОК Ґ

Алгоритм виконання розсилки повідомлень

ДОДАТОК Д

Розгортання системи за допомогою Amazon

ДОДАТОК Е

Швидкодія системи