

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 122 «Комп'ютерні науки»
(освітня програма «Комп'ютерна графіка та розробка ігор»)
на тему:

**«Web-орієнтована освітня платформа для занять
музикою»**

Виконав студент групи КН-20-1
НЕЛЕП Вадим Геннадійович

Керівник роботи:
СУГОНЯК Інна Іванівна

Рецензент:
ГРАФ Марина Сергіївна

Житомир – 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

ЗАТВЕРДЖУЮ

Завідувач кафедри комп'ютерних наук

Марина ГРАФ

«25» січня 2024 р.

ЗАВДАННЯ

на кваліфікаційну роботу освітнього ступеня «бакалавр»
за спеціальністю 122 «Комп'ютерні науки»
(освітня програма «Комп'ютерна графіка та розробка ігор»)

Здобувач вищої освіти: **НЕЛЕП Вадим Геннадійович**

Керівник роботи: **СУГОНЯК Інна Іванівна**

Тема роботи: **«Web-орієнтована освітня платформа для занять музикою»**,

затверджена наказом закладу вищої освіти від **«25» січня 2024 р., №22/с**

Термін здачі закінченої роботи: **«05» червня 2024 р.**

Вихідні дані роботи (зазначається предмет і об'єкт дослідження):

об'єктом дослідження є технології та підходи розробки web-орієнтованих систем у визначених предметних областях.

предметом дослідження є застосування web-орієнтованих технологій для проєктування та розробки освітньої платформи для занять музикою.

Консультанти випускної кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Дата	
		Завдання видав	Завдання прийняв
1	Сугоняк І.І.		
2	Сугоняк І.І.		
3	Сугоняк І.І.		

Календарний план

№ з/п	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Постановка задачі. Пошук і огляд програм-аналогів.		Виконано
2	Формулювання технічного завдання.		Виконано
3	Проектування структури системи.		Виконано
4	Написання програмного коду.		Виконано
5	Тестування системи і виправлення неполадок.		Виконано
6	Оформлення пояснювальної записки		Виконано
7	Створення презентації		Виконано

Здобувач вищої освіти **Вадим НЕЛЕП**
Керівник **Інна СУГОНЯК**

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра складається з веб-додатка платформи для занять музикою та пояснювальної записки. Пояснювальна записка до кваліфікаційної роботи містить текстову частину, викладену на 75 сторінках друкованого тексту. Список використаних джерел містить 23 найменування і займає 2 сторінки. У роботі наведено 42 рисунки та 40 таблиць. Загальний обсяг роботи – 86 сторінок.

Метою роботи є створення якісної онлайн платформи для занять музикою, яка надасть зручне середовище для перегляду навчальних матеріалів та організації занять зі вчителями.

Розробка web-орієнтованої освітньої платформи для занять музикою включала аналіз потреб користувачів і ринку, вибір клієнт-серверної архітектури з використанням PHP, Laravel, HTML, CSS, JavaScript, Bootstrap та MySQL. Було визначено основні сценарії використання, розроблено структуру платформи, базу даних, а також весь програмний код.

Платформа пройшла тестування, готова до впровадження та використання.

Ключові слова: ПЛАТФОРМА, БАЗА ДАНИХ, PHP, LARAVEL, JAVASCRIPT, SANCTUM, OPEN SERVER, PHPMYADMIN.

					КН.КР.Б – 122 – 24 – ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата	Web-орієнтована освітня платформа для занять музикою Пояснювальна записка	Літ.	Арк.	Аркушів
Розроб.	В.Г. НЕЛЕП						3	86
Керівник	І.І. СУГОНЯК					Житомирська політехніка, група КН-20-1		
Рецензент	М.С. ГРАФ							
Н. контр.								
Зав. каф.	М.С. ГРАФ							

ABSTRACT

The bachelor's thesis consists of a web application of a music learning platform and an explanatory note. The explanatory note to the qualification work contains a text part set out on 75 pages of printed text. The list of references includes 23 titles and occupies 2 pages. The work contains 42 illustrations and 40 tables. The total volume of the work is – 86 pages.

The purpose of the work is to create a high-quality online music learning platform that will provide a convenient environment for viewing educational materials and organizing classes with teachers.

The development of an online music learning platform included an analysis of user and market needs, the choice of client-server architecture using PHP, Laravel, HTML, CSS, JavaScript, Bootstrap, and MySQL. The main use cases were identified, the platform structure, and all program code were developed.

The platform has been tested and is ready for implementation and use.

Keywords: PLATFORM, DATABASE, PHP, LARAVEL, JAVASCRIPT, SANCTUM, OPEN SERVER, PHPMYADMIN.

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						4
Змн.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

РЕФЕРАТ	4
ВСТУП	6
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ	8
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ	8
1.1 Постановка задачі	8
1.3 Вибір архітектури	15
1.4 Обґрунтування вибору інструментальних засобів та вимог до апаратного забезпечення.	18
Висновки до розділу 1	21
РОЗДІЛ 2. ПРОЄКТУВАННЯ ВЕБ-ПЛАТФОРМИ.....	23
2.1 Проєктування структури системи	23
2.2 Розробка бази даних	30
2.3 Розробка системних компонентів	42
Висновки до розділу 2	50
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ ВЕБ-ДОДАТКУ	51
3.1 Порядок встановлення та налаштування системи.....	51
3.2 Інтерфейс платформи	54
3.3 Тестування платформи	66
Висновки до розділу 3	69
ВИСНОВКИ.....	71
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	73
ДОДАТКИ.....	75

ВСТУП

Актуальність теми. У сучасному суспільстві, яке характеризується стрімким розвитком інформаційних технологій, все більшого значення набувають дистанційні форми навчання. В умовах глобалізації та цифрової трансформації навчання через інтернет стало не лише модним трендом, але й необхідністю. Особливо це актуально для музичної освіти, де важливо забезпечити можливість регулярних занять, незалежно від місця проживання учня або викладача.

Традиційні методи навчання музиці стикаються з багатьма викликами, такими як обмежений доступ до кваліфікованих викладачів у віддалених регіонах, труднощі з організацією індивідуальних уроків та висока вартість навчання. Впровадження онлайн платформи для занять музикою дозволяє подолати ці проблеми, забезпечуючи гнучкість, доступність та ефективність навчального процесу.

Онлайн платформи відкривають нові горизонти для музичної освіти, надаючи можливість створення інтерактивних курсів, проведення віртуальних уроків, зберігання та обміну навчальними матеріалами. Вони також сприяють розвитку творчих здібностей учнів через використання мультимедійних технологій, інтерактивних вправ та зворотного зв'язку з викладачем.

Метою даної кваліфікаційної роботи є створення якісної онлайн платформи для занять музикою, яка надасть зручне середовище для перегляду навчальних матеріалів та організації занять зі вчителями.

Для досягнення поставленої мети необхідно вирішити такі основні **завдання:**

1. Провести аналіз існуючих онлайн платформ для навчання музиці, визначити їх функціональні можливості, переваги та недоліки.
2. Здійснити вибір оптимальних інструментів та технологій для розробки платформи, оцінити їхню ефективність та доцільність використання.

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						6
Змн.	Арк.	№ докум.	Підпис	Дата		

3. Розробити детальну архітектуру платформи, що включатиме всі основні компоненти системи, реалізувати основні функції платформи, інтегрувати необхідні модулі та забезпечити їхню взаємодію.

4. Провести всебічне тестування платформи з використанням різних методик для виявлення та усунення можливих помилок.

Об'єктом дослідження є технології та підходи розробки web-орієнтованих систем у визначених предметних областях.

Предметом дослідження є застосування web-орієнтованих технологій для проектування та розробки освітньої платформи для занять музикою.

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						7
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

1.1 Постановка задачі

Завдяки можливості отримувати знання дистанційно, учні та викладачі можуть взаємодіяти в зручний для них час, не залежачи від географічного положення та інших обмежень, що сприяє доступності та гнучкості навчального процесу. Онлайн платформи для занять музикою дозволяють поєднувати традиційні методи викладання з сучасними цифровими технологіями, забезпечуючи таким чином більш ефективний і персоналізований підхід до кожного учня. Однак, для створення дійсно ефективної онлайн платформи необхідно вирішити ряд важливих задач, які стосуються не лише технічної реалізації, але й зручності користування, безпеки та підтримки якісного контенту. З огляду на це, важливо забезпечити комплексний підхід до розробки, враховуючи всі аспекти, а також забезпечити її надійну роботу і відповідність сучасним вимогам освітніх технологій.

Основною метою даної роботи є створення онлайн платформи для занять музикою, яка забезпечить інформаційне та навчальне середовище, сприятиме розвитку музичних навичок та творчих здібностей учнів, а також надасть зручний та безпечний доступ до якісної музичної освіти.

Платформа повинна надавати користувачам можливість записуватися на майстер-класи та уроки до викладачів, залишати відгуки про проведені заняття, а також переглядати різноманітні інформаційні ресурси, такі як статті, відеоуроки, інтерв'ю, огляди та рекомендації, відео та аудіоматеріали. Цей функціонал має бути реалізований таким чином, щоб забезпечити максимальну зручність та ефективність для користувачів. Важливо, щоб платформа забезпечувала доступ до високоякісних навчальних матеріалів, які будуть корисними як для початківців, так і для досвідчених музикантів.

З точки зору адміністрування, необхідно забезпечити зручні інструменти для управління контентом, включаючи додавання нових сторінок,

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						8
Змн.	Арк.	№ докум.	Підпис	Дата		

управління навчальними матеріалами та моніторинг. Адміністратори повинні мати можливість швидко та легко оновлювати інформацію, додавати нові ресурси, а також аналізувати взаємодію користувачів з платформою. Для цього необхідно обрати оптимальні технологічні рішення, які забезпечать гнучкість, масштабованість та безпеку платформи.

Крім того, важливо розробити зручний та інтуїтивно зрозумілий інтерфейс користувача, який забезпечить легкість навігації та взаємодії з платформою. Інтерфейс повинен включати можливість запису на уроки та майстер-класи, перегляд відгуків, а також доступ до навчальних матеріалів у зручному форматі. Важливим є також забезпечення можливості перегляду відео-уроків, інтерв'ю та інших мультимедійних матеріалів безпосередньо на платформі, що підвищить її інтерактивність та привабливість для користувачів.

Забезпечення безпеки та конфіденційності даних користувачів є критично важливим аспектом. Необхідно впровадити механізми аутентифікації та авторизації, а також забезпечити захист даних від несанкціонованого доступу. Це включає впровадження надійних протоколів шифрування, регулярне оновлення системи безпеки та моніторинг потенційних загроз. Забезпечення високого рівня безпеки даних є ключовим фактором для довіри користувачів до платформи.

Тестування та оптимізація платформи є наступними важливими етапами. Платформа повинна бути піддана всебічному тестуванню для виявлення та усунення можливих помилок, а також оптимізації продуктивності для забезпечення стабільної роботи при високих навантаженнях. Регулярне оновлення та вдосконалення платформи на основі зворотного зв'язку від користувачів також є важливим для підтримки її актуальності та відповідності потребам користувачів.

Реалізація цих завдань дозволить створити ефективну та зручну онлайн платформу для занять музикою, яка задовольнить потреби як учнів, так і

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						9
Змн.	Арк.	№ докум.	Підпис	Дата		

викладачів, забезпечуючи високий рівень взаємодії та доступу до якісних освітніх ресурсів. Це сприятиме не лише розвитку музичних навичок та творчих здібностей учнів, але й загальному підвищенню рівня музичної освіти, роблячи її доступною для ширшої аудиторії.

1.2. Аналіз аналогів програмного продукту

Важливо провести детальний аналіз існуючих рішень на ринку. Аналіз аналогів програмного продукту дозволяє вивчити сильні та слабкі сторони вже існуючих платформ, що дає можливість визначити найкращі практики та уникнути помилок, які могли б вплинути на якість та ефективність нового продукту. У цьому розділі ми розглянемо декілька провідних онлайн платформ для навчання музиці, таких як The Maestro Online, Aulart та Tonara.

The Maestro Online – це платформа, яка пропонує унікальний підхід до музичної освіти. Вона поєднує індивідуальні уроки, онлайн курси, майстер-класи від знаменитих музикантів, та велику бібліотеку навчальних матеріалів. Метою платформи є допомогти музикантам досягти високих стандартів якості та стати унікальними виконавцями, а не просто копіювати відомі зразки.

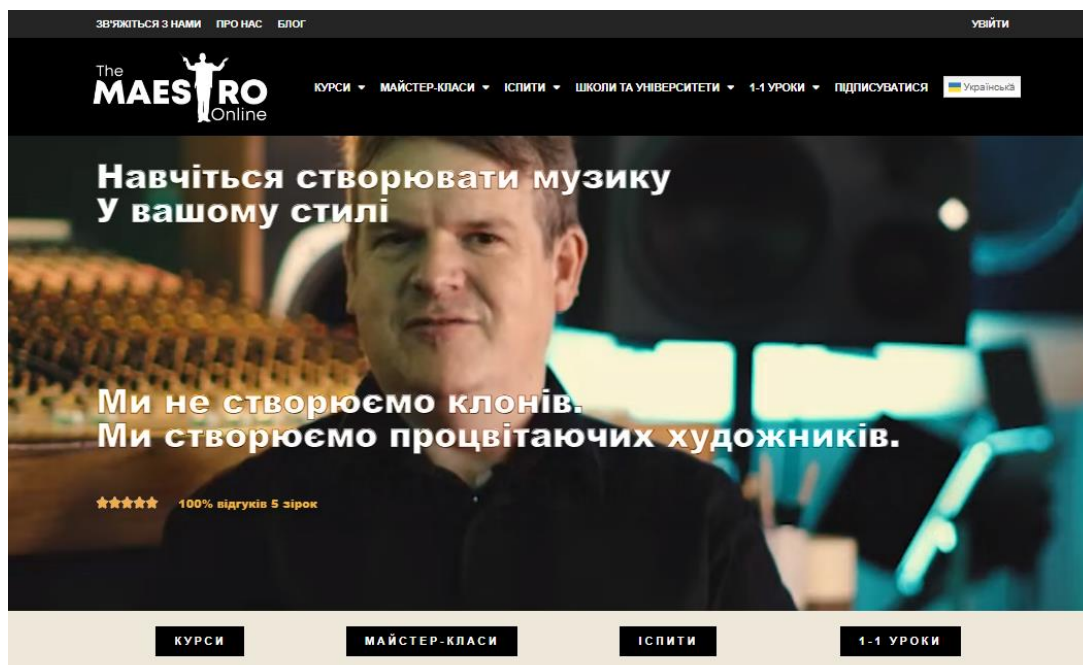


Рис 1.1. – Головна сторінка The Maestro Online

Переваги:

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						10
Змн.	Арк.	№ докум.	Підпис	Дата		

1. Індивідуальні уроки та майстер-класи: можливість записуватись на персональні заняття з відомими музикантами та викладачами. Це забезпечує високу якість навчання, оскільки учні отримують безпосередню увагу від фахівців. Такий підхід дозволяє швидше досягати прогресу, оскільки викладачі можуть зосередитись на конкретних потребах кожного учня та надавати персоналізовані рекомендації.

2. Високоякісні навчальні матеріали: платформа пропонує широкий спектр навчальних матеріалів, таких як відеоуроки та інтерв'ю з експертами. Це дозволяє учням отримувати знання від провідних фахівців та збагачувати свій досвід різноманітними поглядами та методиками.

3. Доступ до додаткових ресурсів: платформа надає доступ до статей, оглядів, рекомендацій та мультимедійних матеріалів. Це допомагає учням розширювати свої знання та знаходити додаткові джерела інформації для самостійного вивчення.

4. Взаємодія учнів та викладачів: Можливість залишати відгуки про заняття сприяє створенню інтерактивного середовища, де учні можуть ділитися своїми враженнями та отримувати зворотний зв'язок від викладачів.

Недоліки:

1. Вартість: персональні заняття з відомими викладачами можуть бути дорогими, що може обмежити доступ до платформи для деяких учнів. Високі ціни можуть стати бар'єром для тих, хто має обмежені фінансові ресурси, але прагне отримати якісну музичну освіту.

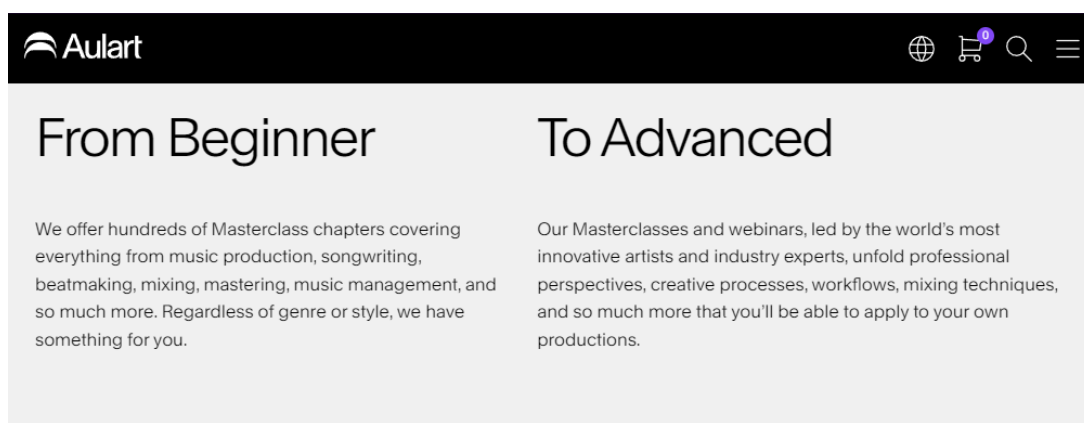
2. Орієнтація на обмежену аудиторію: Платформа може бути менш доступною для початківців через високі вимоги до рівня підготовки. Це може обмежити кількість користувачів, які можуть скористатися послугами платформи, та звужити її аудиторію до більш досвідчених музикантів.

3. Високий рівень самостійності: The Maestro Online вимагає від учнів високого рівня самостійності та дисципліни. Оскільки багато уроків та матеріалів доступні онлайн і потребують самостійного опрацювання, учні

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						11
Змн.	Арк.	№ докум.	Підпис	Дата		

повинні мати навички самоконтролю та самоорганізації. Для деяких студентів це може бути викликом, особливо якщо вони звикли до більш структурованого та керованого середовища навчання. Відсутність постійного контролю з боку викладача може вплинути на прогрес тих, хто не може самостійно мотивувати себе до регулярного навчання та практики.

Aulart – це онлайн-освітня платформа, заснована на майстер-класах і живих курсах, на яких міжнародно відомі митці, такі як продюсери, звукорежисери та інші професіонали в музичній індустрії, демонструють свій унікальний спосіб роботи та створення музики. Aulart дає можливість дізнатися про прийоми та досвід, які допомогли їм стати тими, ким вони є, спеціалізується на наданні високоякісних відеоматеріалів, які охоплюють різні аспекти музичного виробництва та творчого процесу.



Join our community of over
30,000 music creators.

Рис 1.2. – Головна сторінка Aulart

Переваги:

1. Відомі інструктори: Aulart пропонує майстер-класи від відомих музикантів, продюсерів та інженерів звукозапису. Це забезпечує високу якість навчання, оскільки учні отримують знання від найкращих у своїй галузі.

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						12
Змн.	Арк.	№ докум.	Підпис	Дата		

Відомі інструктори мають багатий досвід та можуть поділитися унікальними інсайтами, що робить навчання більш захоплюючим та корисним.

2. Інтерактивні заходи: платформа надає можливість брати участь у вебінарах та інших інтерактивних заходах, що дозволяє учням взаємодіяти з інструкторами в реальному часі. Це підвищує залученість учнів та надає можливість отримати негайний зворотний зв'язок та відповіді на свої питання.

3. Висока якість відео: всі навчальні матеріали зняті на професійному рівні, що забезпечує чіткість та якість зображення та звуку. Це важливо для музичного навчання, де деталі мають велике значення. Високоякісні відео дозволяють учням краще розуміти матеріал і слідувати за інструкціями інструкторів.

Недоліки:

1. Відсутність індивідуального підходу: Платформа більше орієнтована на масові курси, ніж на персоналізоване навчання. Це може бути недоліком для тих учнів, які потребують індивідуальної уваги та підтримки, щоб досягти кращих результатів у навчанні.

2. Обмеженість взаємодії: хоча платформа пропонує можливість участі у вебінарах та інших інтерактивних заходах, рівень взаємодії між учнями та викладачами все ще обмежений. Відсутність постійного зворотного зв'язку та можливості задавати питання в реальному часі може вплинути на процес навчання, особливо для учнів, які потребують постійної підтримки та корекції.

3. Відсутність структурованої навчальної програми: на платформі відсутні чітко структуровані навчальні програми, що ускладнює побудову послідовного навчального плану. Учні можуть відчувати певну розгубленість через наявність великої кількості окремих майстер-класів без чіткої структури та рекомендацій щодо проходження курсів.

Топага – це веб-додаток, що спеціалізується на управлінні музичним навчальним процесом, пропонує інструменти для відстеження прогресу

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						13
Змн.	Арк.	№ докум.	Підпис	Дата		

учнів, створення завдань та надання зворотного зв'язку в реальному часі, включаючи систему змагань та досягнень.

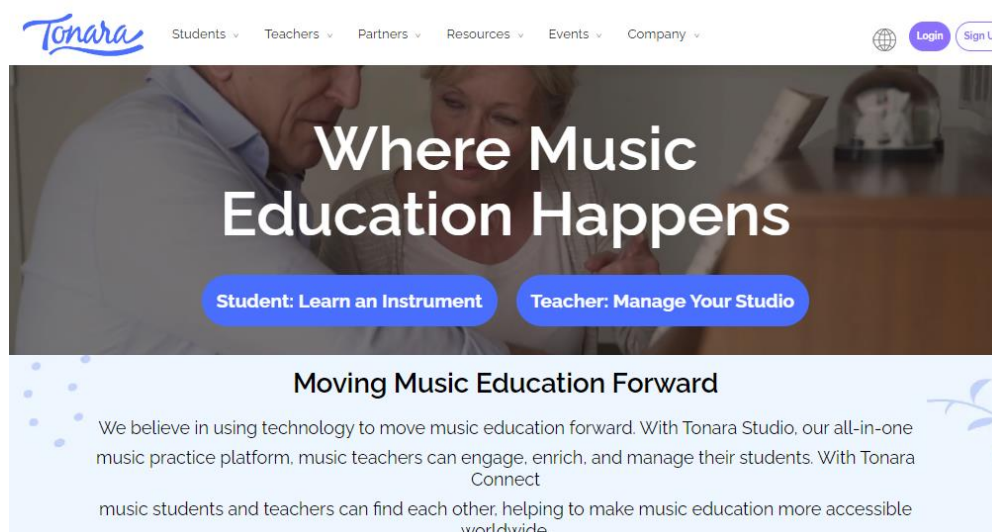


Рис 1.3. – Головна сторінка Tonara

Переваги:

1. Керування навчальним процесом: Tonara надає можливість структурувати та організовувати навчальний процес викладачам, відстежувати прогрес учнів у реальному часі та надавати додаткову підтримку.
2. Мотивація учнів: Tonara використовує елементи гейміфікації для підвищення мотивації. Система досягнень допомагає нагороди та визнання за зусилля та результати, що сприяє підвищенню мотивації та самовпевненості.
3. Можливість групового та індивідуального обговорення: платформа дозволяє учням взаємодіяти один з одним та з викладачами через групові та індивідуальні обговорення, що створює спільноту та підтримку серед учасників.

Недоліки:

1. Необхідність платної підписки: для доступу до всіх функцій платформи необхідна платна підписка, що може стати перешкодою для деяких користувачів.
2. Обмежена спеціалізація: Tonara спеціалізується на обмеженій кількості музичних інструментів, що може обмежити її корисність для учнів,

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						14
Змн.	Арк.	№ докум.	Підпис	Дата		

які хочуть навчитися грати на інших інструментах. Це зменшує універсальність платформи та її привабливість для широкої аудиторії.

3. Обмеження на синхронне навчання: хоча платформа пропонує інструменти для управління навчальним процесом та відстеження прогресу, вона не завжди включає можливість проведення інтерактивних уроків у реальному часі, що може бути важливим для деяких учнів та викладачів.

1.3 Вибір архітектури

Архітектура програмного забезпечення — це високорівнева структура програмної системи. Вона визначає основні компоненти системи та їх взаємодію. Архітектура описує, як система розбита на модулі та яким чином ці модулі взаємодіють.

Основа будь-якої надійної та ефективної веборієнтованої платформи полягає у виборі відповідної архітектури ПЗ. Враховуючи вимоги до нашої платформи для занять музикою, було обрано клієнт-серверну архітектуру з використанням шаблону проєктування MVC (Model-View-Controller). Це рішення забезпечує необхідну гнучкість, масштабованість, продуктивність і безпеку.

Клієнт-серверна архітектура є однією з найпопулярніших моделей для побудови сучасних веб-додатків. Вона базується на розподілі функцій між клієнтом, який взаємодіє з користувачем, та сервером, який обробляє дані і виконує бізнес-логіку.

У цій архітектурі клієнт надсилає запити до сервера для отримання даних або виконання певних дій. Сервер обробляє ці запити, виконуючи необхідні операції, такі як доступ до бази даних, обчислення або інша обробка даних, і повертає результати клієнту. Це дозволяє розподілити навантаження між двома компонентами: клієнт займається відображенням інтерфейсу та взаємодією з користувачем, а сервер виконує складні обчислення та управління даними.

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		15

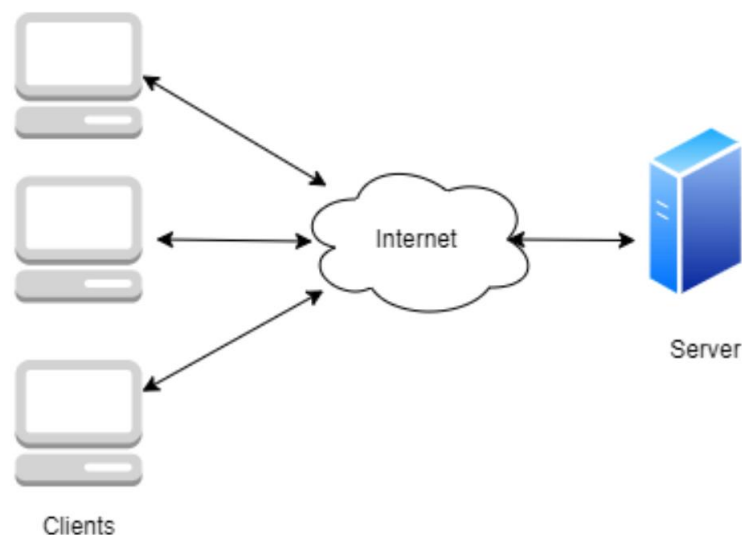


Рис 1.4. – Модель клієнт-сервер

Клієнт-серверна архітектура забезпечує кілька важливих переваг.

По-перше, вона дозволяє легко масштабувати систему, додаючи нові сервери для обробки запитів. Це особливо корисно для великих платформ з великою кількістю користувачів.

По-друге, ця архітектура дозволяє розробляти клієнтську та серверну частини незалежно одна від одної, що полегшує підтримку та оновлення системи.

З точки зору безпеки, клієнт-серверна архітектура також має переваги. Сервер може централізовано контролювати доступ до даних та ресурсів, що забезпечує кращий захист інформації. Крім того, сервер може обробляти і зберігати конфіденційні дані, зменшуючи ризик несанкціонованого доступу з боку клієнта.

Централізоване управління на сервері дозволяє легко реалізовувати функції резервного копіювання і відновлення даних, моніторингу системи, а також впровадження оновлень і патчів без необхідності втручання з боку користувачів. Це підвищує загальну надійність і стабільність роботи додатка, оскільки будь-які проблеми або уразливості можуть бути швидко і ефективно усунуті на серверному рівні.

MVC (Model-View-Controller) — це шаблон проєктування, який розділяє додаток на три основні компоненти: Модель (Model), Представлення (View) і Контролер (Controller). Такий підхід допомагає організувати код і розділити відповідальності, що робить розробку, підтримку та розширення додатка більш зручними та ефективними.

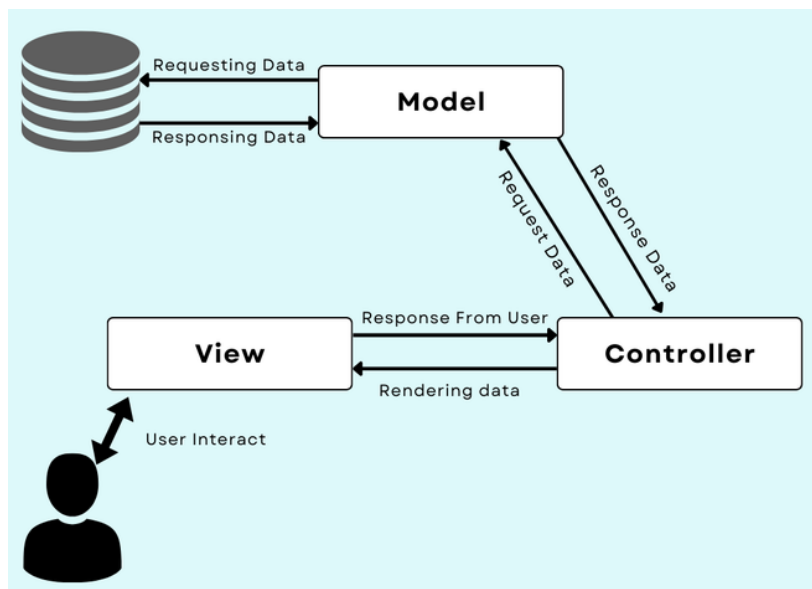


Рис 1.5. – Модель MVC

Кожен з компонентів шаблону має свою чітко визначену роль. Модель відповідає за управління даними і бізнес-логікою, Представлення займається відображенням інформації користувачу, а Контролер обробляє запити від користувачів і координує взаємодію між Моделлю та Представленням. Це розділення забезпечує чітку структурованість коду, робить його більш організованим і зручним для підтримки та розширення.

Завдяки модульній структурі MVC, додаток можна легко масштабувати та розширювати. Нові функціональні можливості можуть бути додані без значних змін у існуючому коді, що спрощує адаптацію системи до нових вимог і умов. Це особливо важливо для платформ, які мають перспективи зростання і розвитку, оскільки дозволяє швидко реагувати на зміну потреб користувачів та ринку.

MVC полегшує процес підтримки та тестування додатка. Оскільки компоненти Моделі, Представлення і Контролера є незалежними, їх можна

тестувати окремо, що підвищує надійність і знижує ризик помилок. Це дозволяє забезпечити стабільну роботу додатка і швидко впроваджувати нові функції або виправляти помилки.

1.4 Обґрунтування вибору інструментальних засобів та вимог до апаратного забезпечення.

Детальне обґрунтування вибіру інструментів та обладнання для розробки будь-якого програмного забезпечення є надзвичайно важливим з кількох причин. По-перше, воно забезпечує впевненість у тому, що обрані інструменти найкращим чином відповідають потребам проекту, його технічним вимогам та очікуванням користувачів. Це дозволяє оптимізувати ресурси, зменшити витрати та уникнути непередбачених проблем під час розробки та експлуатації платформи. По-друге, обґрунтування сприяє узгодженості та стандартизації процесів розробки. Використання перевірених та добре задокументованих інструментів забезпечує легкість у підтримці та розвитку проекту.

Обрана для розробки мова програмування PHP є однією з найпопулярніших мов для веб-розробки завдяки своїй простоті, продуктивності та широкому використанню. PHP забезпечує ефективну роботу серверної частини додатків, дозволяючи швидко імплементувати та впроваджувати веб-системи. Велика спільнота розробників та багата екосистема бібліотек роблять PHP ідеальним вибором для створення веб-додатків.

Laravel, як фреймворк на основі PHP, значно спрощує розробку додатків, використовуючи шаблон проектування MVC (Model-View-Controller). Laravel пропонує велику кількість вбудованих інструментів, таких як Eloquent ORM для роботи з базами даних, Blade для шаблонізації, і систему маршрутизації для управління запитами. Це дозволяє швидко створювати безпечні та продуктивні додатки з чистим та організованим кодом.

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						18
Змн.	Арк.	№ докум.	Підпис	Дата		

MySQL є вибраною реляційною базою даних для зберігання даних нашої платформи. Ця система управління базами даних відома своєю надійністю, високою продуктивністю та здатністю обробляти великі обсяги даних. MySQL підтримує складні SQL-запити, що дозволяє ефективно працювати з даними та забезпечує швидкий доступ до інформації. Широка підтримка та велика кількість документації роблять MySQL зручним вибором для розробників.

MySQL відома своєю високою продуктивністю та ефективністю обробки даних. Завдяки оптимізації запитів та ефективному управлінню пам'яттю, MySQL може обробляти великі обсяги даних та забезпечувати швидкий доступ до інформації. Це є критично важливим для нашої платформи, де швидкість обробки запитів безпосередньо впливає на досвід користувачів.

MySQL також забезпечує високу безпеку даних, підтримуючи різноманітні механізми аутентифікації та авторизації. Це особливо важливо для веб-додатків, які працюють з конфіденційною інформацією користувачів. Завдяки своїй популярності, MySQL має багату спільноту, що надає підтримку та ресурси для вирішення будь-яких проблем.

phpMyAdmin — це обраний веб-інтерфейс для управління базою даних MySQL. Однією з найбільших переваг phpMyAdmin є його зручний та інтуїтивно зрозумілий веб-інтерфейс. Він дозволяє користувачам легко взаємодіяти з базами даних через браузер, що значно спрощує адміністрування. Інтерфейс забезпечує доступ до всіх основних функцій MySQL, включаючи створення і редагування баз даних, таблиць, виконання SQL-запитів, а також експорт та імпорт даних. Це робить phpMyAdmin ідеальним інструментом для адміністраторів і розробників, які хочуть ефективно керувати базами даних без необхідності використовувати командний рядок.

phpMyAdmin підтримує всі основні функції управління базами даних MySQL, такі як створення, редагування та видалення баз даних, таблиць,

індексів та записів. Крім того, він дозволяє виконувати складні SQL-запити, створювати та редагувати процедури і тригери, а також керувати привілеями користувачів. Цей широкий функціонал робить phpMyAdmin потужним інструментом для ефективного управління базами даних, забезпечуючи всі необхідні інструменти для розробників і адміністраторів.

Для створення клієнтської частини буде використано наступний набір інструментів: HTML, CSS, JavaScript, jQuery та Bootstrap.

HTML є основною мовою розмітки для створення веб-сторінок, що визначає структуру та організацію контенту. Використання HTML є обов'язковим для будь-якого веб-додатка, оскільки він визначає, як контент буде відображатися у браузері. HTML підтримує семантичну розмітку, що покращує доступність веб-сторінок для користувачів з обмеженими можливостями та полегшує індексацію контенту пошуковими системами.

CSS використовується для стилізації та візуального оформлення веб-сторінок. Вона дозволяє відокремити зміст HTML від його візуального представлення, забезпечуючи гнучкість та контроль над зовнішнім виглядом додатка. За допомогою CSS можна визначати шрифти, кольори, розміри елементів, відступи, розташування на сторінці та багато іншого. CSS також забезпечує адаптивність веб-сторінок, дозволяючи оптимально відображати контент на різних пристроях та екранах різного розміру.

Bootstrap є популярним фреймворком для розробки адаптивних та мобільно-орієнтованих веб-додатків. Він містить готові компоненти та стилі, що дозволяють швидко створювати привабливі та функціональні інтерфейси користувача. Bootstrap забезпечує підтримку адаптивного дизайну, що дозволяє веб-сторінкам коректно відображатися на пристроях з різними розмірами екранів. Однією з головних переваг Bootstrap є його легкість у використанні та висока швидкість розробки. Завдяки набору готових компонентів, таких як кнопки, форми, навігаційні панелі та інші елементи,

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						20
Змн.	Арк.	№ докум.	Підпис	Дата		

розробники можуть швидко створювати складні інтерфейси без необхідності писати багато коду.

JavaScript є основною мовою програмування для створення інтерактивних та динамічних веб-додатків. Вона дозволяє додавати функціональність до веб-сторінок, забезпечуючи взаємодію з користувачем у реальному часі. JavaScript використовується для створення анімацій, обробки подій, валідації форм, асинхронного завантаження контенту та багатьох інших завдань. JavaScript також підтримує роботу з DOM (Document Object Model), що дозволяє динамічно змінювати структуру та зміст веб-сторінки без перезавантаження.

jQuery – це легковагова JavaScript-бібліотека, яка значно спрощує роботу з JavaScript. Вона забезпечує простий та ефективний спосіб взаємодії з DOM, обробки подій, створення анімацій та виконання асинхронних запитів (AJAX). jQuery дозволяє писати менш об'ємний код для виконання тих самих завдань, що робить розробку швидшою та простішою. Однією з ключових переваг jQuery є її кросбраузерна сумісність, що дозволяє розробникам писати код, який буде працювати у всіх сучасних браузерах без додаткових налаштувань. Крім того, jQuery має велику кількість плагінів, які розширюють її функціональність та дозволяють швидко додавати нові можливості до веб-додатка.

Висновки до розділу 1

У першому розділі кваліфікаційної роботи було проведено детальний аналіз сучасних платформ, які схожі за ціллю та функціоналом до розроблюваної платформи для занять музикою. Мета дослідження була чітко визначена, а також були сформульовані конкретні завдання проекту.

Було визначено середовище програмного забезпечення для розробки платформи та набір інструментів, які будуть використовуватися. Зокрема, обрано PHP та Laravel для серверної частини проекту та MySQL для управління базами даних. Інструменти PhpStorm та phpMyAdmin були обрані

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						21
Змн.	Арк.	№ докум.	Підпис	Дата		

для розробки та адміністрування баз даних, відповідно. Для створення клієнтської частини платформи було обрано HTML, CSS, JavaScript, jQuery та Bootstrap. Цей вибір був обґрунтований їх ефективністю, швидкодією та можливістю легкої інтеграції та масштабованості.

Особливу увагу було приділено архітектурним аспектам платформи. Було визначено, що патерн проектування MVC (Model-View-Controller) є найкращим вибором для даного проекту, оскільки він дозволяє чітко розділити логічні компоненти системи, забезпечуючи гнучкість і модульність коду.

Таким чином, у першому розділі були описані ключові принципи та фундаментальні рішення для подальшої розробки та реалізації платформи, визначаючи її стратегічний напрямок та основні технічні параметри. Ці рішення закладають міцну основу для створення ефективної, надійної та зручної у використанні платформи для занять музикою.

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						22
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2. ПРОЄКТУВАННЯ ВЕБ-ПЛАТФОРМИ

2.1 Проєктування структури системи

Платформа для занять музикою матиме ряд особливостей, які підкреслять її виїнятковість та функціональність:

1. Мультифункціональність: Платформа надає користувачам широкий спектр можливостей, такі як запис на уроки до викладачів та на майстер-класи, перегляд статей, відеоуроків, навчальних матеріалів, а також можливість залишати відгуки про вчителів.

2. Адаптивний дизайн: Інтерфейс платформи розроблятиметься з урахуванням адаптивності, що дозволяє зручно використовувати її на різних пристроях, таких як комп'ютери, планшети та смартфони.

3. Можливості для адміністрування: Адміністратори платформи мають доступ до спеціальних функцій для управління контентом, включаючи додавання нових статей, відео, майстер-класів та керування користувачами.

4. Масштабованість: Архітектура платформи дозволяє легко розширювати її функціональність та масштабувати у відповідності до зростаючих потреб користувачів.

Одним з ключових аспектів проєктування є визначення випадків використання, які відображають функціональні можливості системи з точки зору користувачів. Випадки використання описують, як різні типи користувачів взаємодіють із системою для виконання своїх завдань.

Для платформи визначено три основні категорії користувачів: неавторизовані користувачі, авторизовані користувачі та адміністратори. Кожна з цих категорій має свої специфічні функції та взаємодії з системою.

Неавторизовані користувачі можуть здійснювати базові дії, такі як перегляд інформації про напрямки діяльності та сторінок вчителів. Вони також мають можливість зареєструватися або увійти до системи для отримання повного доступу до функцій платформи.

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						23
Змн.	Арк.	№ докум.	Підпис	Дата		

Авторизовані користувачі мають доступ до більш широкого спектру функцій. Вони можуть корисну інформацію (статті, відео-уроки), записуватися на навчання до вчителів, керівника та на майстер-класи, переглядати додаткові матеріали та залишати відгуки про вчителів.

Адміністратори в свою чергу мають доступ до функцій управління системою. Вони можуть керувати контентом та створювати його, мають доступ до учнівських записів, вчителів тощо.

Для візуалізації цих категорій користувачів з системою доцільно створити UML-діаграму випадків використання (Use Case Diagram). Use Case діаграма надасть змогу планувати роботу більш ефективно, оскільки візуальне представлення сприяє чіткому розумінню структури системи та допомагає ефективніше керувати розробкою. Діаграма Use Case дозволяє виявити всі можливі сценарії використання системи, що допомагає уникнути пропусків у функціональних вимогах та забезпечити повноту проєктування.

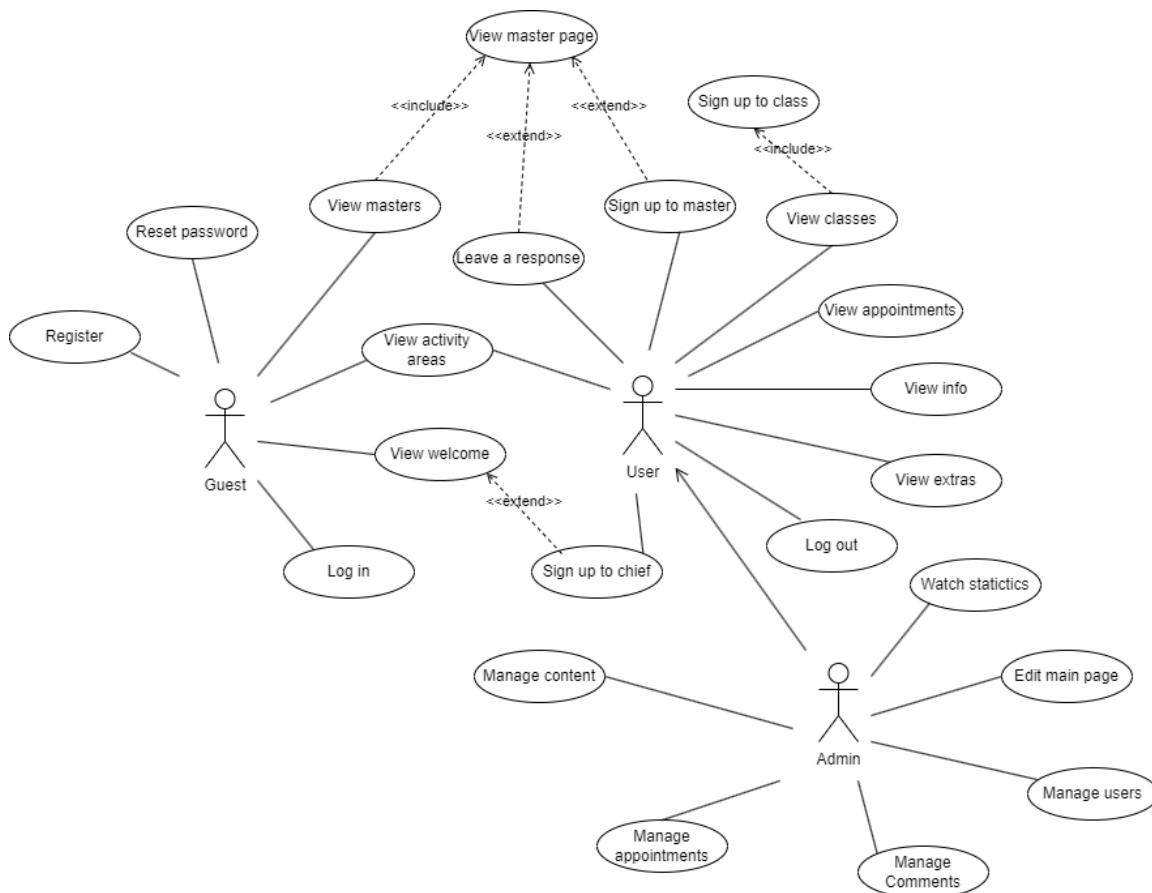


Рис 2.1. – Діаграма Use Case

Опис прецедентів діаграми наведено в табл. 2.1. – 2.22.

Таблиця 2.1.

Прецедент «Реєстрація»

Назва прецеденту	Реєстрація
Актори	Неавторизований користувач
Опис	Користувач має можливість створити новий обліковий запис на платформі, ввівши необхідні дані.

Таблиця 2.2.

Прецедент «Реєстрація»

Назва прецеденту	Реєстрація
Актори	Неавторизований користувач
Опис	Користувач має можливість створити новий акаунт на платформі ввівши нові дані.

Таблиця 2.3.

Прецедент «Вхід»

Назва прецеденту	Вхід
Актори	Неавторизований користувач
Опис	Користувач входить у свій обліковий запис, вводячи логін і пароль.

Таблиця 2.4.

Прецедент «Вихід»

Назва прецеденту	Вихід
Актори	Авторизований користувач, адміністратор
Опис	Користувач виходить зі свого облікового запису.

Таблиця 2.5.

Прецендент «Відновлення паролю»

Назва преценденту	Відновлення паролю
Актори	Неавторизований користувач
Опис	Неавторизований користувач може отримати посилання для скидання паролю на електронну пошту і відновити його, якщо він забув свій поточний.

Таблиця 2.6.

Прецендент «Перегляд головної сторінки»

Назва преценденту	Перегляд головної сторінки
Актори	Неавторизований користувач, авторизований користувач, адміністратор
Опис	Користувач може переглянути головну сторінку платформи з вітальною інформацією.

Таблиця 2.7.

Прецендент «Перегляд напрямків діяльності»

Назва преценденту	Перегляд напрямків діяльності
Актори	Неавторизований користувач, авторизований користувач, адміністратор
Опис	Користувач може переглядати статті про основні та додаткові напрямки діяльності.

Таблиця 2.8.

Прецендент «Перегляд вчителів»

Назва преценденту	Перегляд вчителів
Актори	Авторизований користувач, адміністратор
Опис	Користувач може переглядати статті про основні та додаткові напрямки діяльності.

Таблиця 2.9.

Прецендент «Перегляд сторінки вчителя»

Назва преценденту	Перегляд сторінки вчителя
Актори	Авторизований користувач, адміністратор
Опис	Користувач може переглядати інформацію про конкретного вчителя.

Таблиця 2.10.

Прецендент «Перегляд майстер-класів»

Назва преценденту	Перегляд майстер-класів
Актори	Авторизований користувач, адміністратор
Опис	Користувач має можливість переглядати список доступних майстер-класів.

Таблиця 2.11.

Прецендент «Перегляд корисної інформації»

Назва преценденту	Перегляд корисної інформації
Актори	Авторизований користувач, адміністратор
Опис	Користувач може переглядати корисну інформацію (статті, відео-уроки, інтерв'ю тощо) .

Таблиця 2.12.

Прецендент «Перегляд додаткових матеріалів»

Назва преценденту	Перегляд додаткових матеріалів
Актори	Авторизований користувач, адміністратор
Опис	Користувач може переглядати додаткові матеріали (рекомендії, огляди тощо).

Таблиця 2.13.

Прецендент «Запис на майстер-клас»

Назва преценденту	Запис на майстер-клас
Актори	Авторизований користувач, адміністратор
Опис	Користувач може записатися на обраний майстер-клас.

Таблиця 2.14.

Прецендент «Запис до вчителя»

Назва преценденту	Запис до вчителя
Актори	Авторизований користувач, адміністратор
Опис	Користувач може записатися до конкретного вчителя.

Таблиця 2.15.

Прецендент «Запис до керівника»

Назва преценденту	Запис до керівника
Актори	Авторизований користувач, адміністратор
Опис	Користувач може записатися на навчання до керівника.

Таблиця 2.16.

Прецедент «Залишення відгуку»

Назва прецеденту	Залишення відгуку
Актори	Авторизований користувач, адміністратор
Опис	Користувач може залишити відгук про вчителя.

Таблиця 2.17.

Прецедент «Перегляд записів»

Назва прецеденту	Перегляд записів
Актори	Авторизований користувач, адміністратор
Опис	Користувач може переглядати та відслідковувати стан записів до керівника, вчителів та на майстер-класи.

Таблиця 2.18.

Прецедент «Залишення відгуку»

Назва прецеденту	Залишення відгуку
Актори	Авторизований користувач, адміністратор
Опис	Користувач може залишити відгук про вчителя.

Таблиця 2.19.

Прецедент «Керування користувачами»

Назва прецеденту	Керування користувачами
Актори	Адміністратор
Опис	Адміністратор може переглядати список користувачів та змінювати їм роль.

Таблиця 2.20.

Прецендент «Керування контентом»

Назва преценденту	Керування контентом
Актори	Адміністратор
Опис	Адміністратор може додавати, редагувати, приховувати та видаляти контент.

Таблиця 2.21.

Прецендент «Редагування головної сторінки»

Назва преценденту	Редагування головної сторінки
Актори	Адміністратор
Опис	Адміністратор може окремо редагувати головну сторінку сайту.

Таблиця 2.22.

Прецендент «Перегляд статистики»

Назва преценденту	Перегляд статистики
Актори	Адміністратор
Опис	Адміністратор має можливість переглядати статистичну інформацію.

2.2 Розробка бази даних

База даних є основою для зберігання усієї необхідної інформації на платформі. Перед тим, як безпосередньо створювати базу даних за допомогою моделей у Laravel, необхідне попереднє її проєктування: визначення структури бази даних, її таблиць, рядків та взаємодій. Діаграма сутність-зв'язок надасть можливість візуально представити усі компоненти БД.

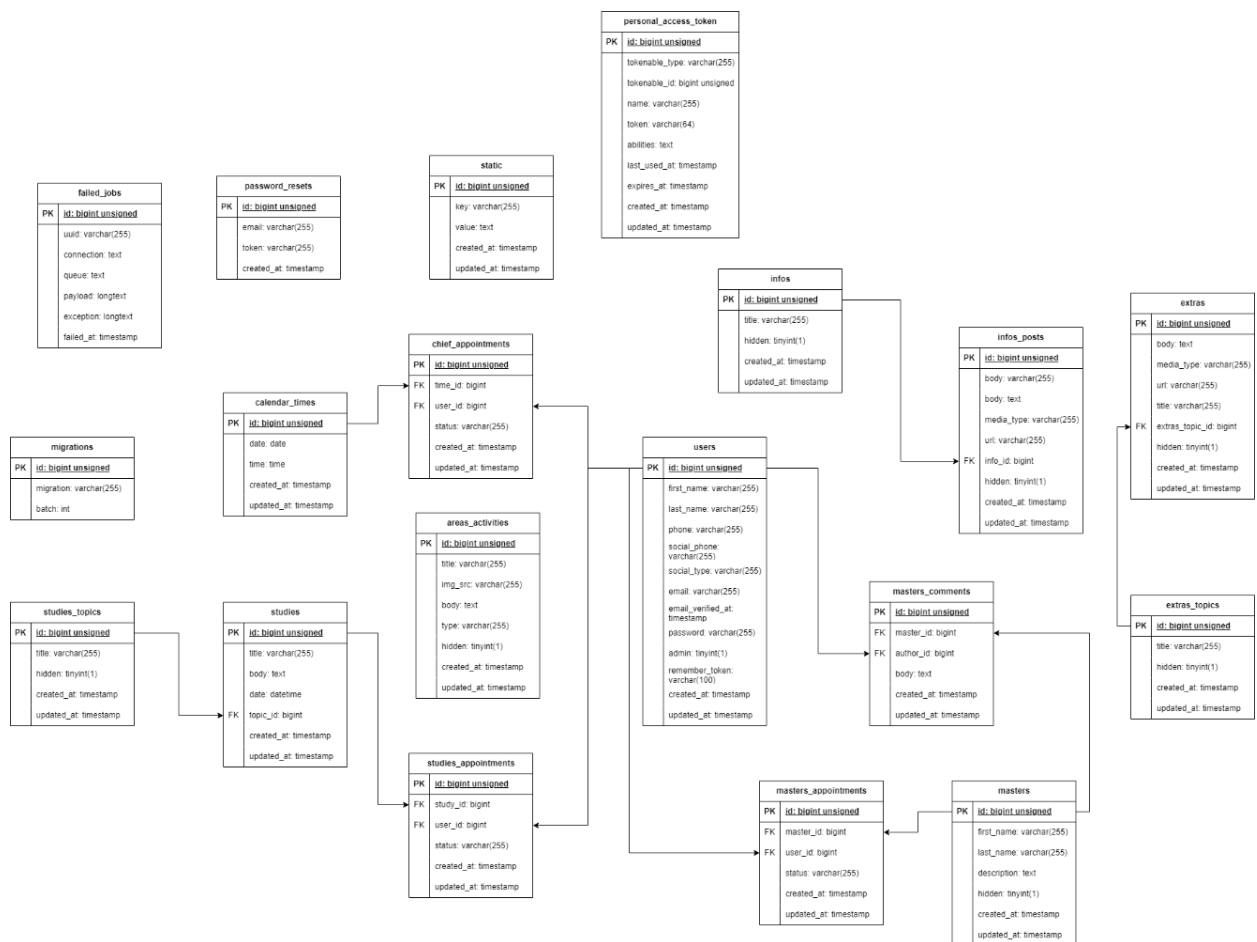


Рис 2.2. – Діаграма сутність-зв'язок

База даних налічує дев'ятнадцять таблиць:

1. users – таблиця, що містить інформацію про користувачів платформи.
2. masters – таблиця, що містить інформацію про вчителів.
3. masters_comments – таблиця, яка зберігає відгуки про вчителів.
4. masters_appointments – таблиця, яка зберігає записи до вчителів.
5. chief_appointments – таблиця, яка зберігає записи до керівника.
6. calendar_times – таблиця, яка зберігає доступні дати запису до керівника.
7. studies – таблиця, що зберігає інформацію про майстер-класи.
8. studies_topics – таблиця категорій майстер-класів.
9. studies_appointments – таблиця, яка зберігає записи на майстер-класи.
10. infos_posts – таблиця, що зберігає дописи корисної інформації.
11. infos – таблиця категорій корисної інформації.

12. extras – таблиця, що зберігає додаткові дописи.
 13. extras_topics – таблиця категорій додаткового.
 14. areas_activities – таблиця, що зберігає вміст сторінок про основні та додаткові напрямки діяльності.
 15. password_reset – таблиця, що зберігає токени для скидання паролю.
 16. static – таблиця, що зберігає контент головної сторінки платформи.
 17. failed_jobs – таблиця, що зберігає записи про невдалі спроби виконання черг (jobs).
 18. personal_access_tokens – таблиця, що зберігає особисті токени доступу користувачів для авторизації та аутентифікації в системі через API.
 19. migrations – таблиця, що містить інформацію про міграції.
- Детальний опис таблиць БД наведено у табл. 2.23 – 2.40.

Таблиця 2.23.

Таблиця “users”

Поле	РК	FK	Тип даних	Призначення
id	+	-	bigint unsigned	Унікальний ідентифікатор
firstname	-	-	varchar(255)	Ім’я
lastname	-	-	varchar(255)	Прізвище
phone	-	-	varchar(255)	Номер телефону
social_phone	-	-	varchar(255)	Адреса месенджера
social_type	-	-	varchar(255)	Месенджер (Telegram/Viber)
email	-	-	varchar(255)	Електронна пошта
email_verified_at	-	-	timestamp	Час верифікації пошти
password	-	-	varchar(255)	Пароль
admin	-	-	tinyint(1)	Роль адміністратора (0/1)
remember_token	-	-	varchar(100)	Токен для сесій «remember me»

Таблиця 2.24.

Таблиця “masters”

Поле	PK	FK	Тип даних	Призначення
id	+	-	bigint unsigned	Унікальний ідентифікатор
firstname	-	-	varchar(255)	Ім'я
lastname	-	-	varchar(255)	Прізвище
description	-	-	text	Опис
img_src	-	-	varchar(255)	Фотографія
hidden	-	-	tinyint(1)	Видимість (0/1)
created_at	-	-	timestamp	Час створення
updated_at	-	-	timestamp	Час оновлення

Таблиця 2.25.

Таблиця “masters_comments”

Поле	PK	FK	Тип даних	Призначення
id	+	-	bigint unsigned	Унікальний ідентифікатор
master_id	-	+	bigint	Ідентифікатор вчителя
author_id	-	+	bigint	Ідентифікатор автора відгука
body	-	-	text	Текст відгука
created_at	-	-	timestamp	Час створення
updated_at	-	-	timestamp	Час оновлення

Таблиця 2.26.

Таблиця “masters_appointments”

Поле	PK	FK	Тип даних	Призначення
id	+	-	bigint unsigned	Унікальний ідентифікатор
master_id	-	+	bigint	Ідентифікатор вчителя
user_id	-	+	bigint	Ідентифікатор користувача

Продовження таблиці 2.26.

status	-	-	varchar(255)	Статус запису (wait_accept/accepted/rejected)
created_at	-	-	timestamp	Час створення
updated_at	-	-	timestamp	Час оновлення

Таблиця 2.27.

Таблиця “chief_appointments”

Поле	PK	FK	Тип даних	Призначення
id	+	-	bigint unsigned	Унікальний ідентифікатор
time_id	-	+	bigint	Дата і час запису
user_id	-	+	bigint	Ідентифікатор користувача
status	-	-	varchar(255)	Статус (wait/accept/reject)
created_at	-	-	timestamp	Час створення

Таблиця 2.28.

Таблиця “calendar_times”

Поле	PK	FK	Тип даних	Призначення
id	+	-	bigint unsigned	Унікальний ідентифікатор
date	-	-	date	Дата
times	-	-	time	Доступні години для запису
created_at	-	-	timestamp	Час створення
updated_at	-	-	timestamp	Час оновлення

Таблиця 2.29.

Таблиця “studies”

Поле	PK	FK	Тип даних	Призначення
id	+	-	bigint unsigned	Унікальний ідентифікатор

Провження таблиці 2.29.

title	-	-	varchar(255)	Заголовок
body	-	-	text	Опис майстер-класу
date	-	-	datetime	Дата
topic_id	-	+	big	ID типу майстер-класу
created_at	-	-	timestamp	Час створення
updated_at	-	-	timestamp	Час оновлення

Таблиця 2.30.

Таблиця “studies_topics”

Поле	PK	FK	Тип даних	Призначення
id	+	-	bigint unsigned	Унікальний ідентифікатор
title	-	-	varchar(255)	Назва категорії майстер класу
hidden	-	-	tinyint(1)	Видимість (0/1)
created_at	-	-	timestamp	Час створення
updated_at	-	-	timestamp	Час оновлення

Таблиця 2.31.

Таблиця “studies_appointments”

Поле	PK	FK	Тип даних	Призначення
id	+	-	bigint unsigned	Унікальний ідентифікатор
study_id	-	+	bigint	Ідентифікатор майстер-класу
user_id	-	+	bigint	Ідентифікатор користувача
status	-	-	varchar(255)	Статус запису (wait/ асеп/reject)
created_at	-	-	timestamp	Час створення
updated_at	-	-	timestamp	Час оновлення

Таблиця 2.31.

Таблиця “infos_posts”

Поле	PK	FK	Тип даних	Призначення
id	+	-	bigint unsigned	Унікальний ідентифікатор
body	-	-	text	Текст допису
media_type	-	-	varchar(255)	Тип медіа (youtube/image/video/audio)
url	-	-	varchar(255)	Посилання на медіа
info_id	-	+	bigint	Ідентифікатор категорії корисної інформації
hidden	-	-	tinyint(1)	Видимість (0/1)
created_at	-	-	timestamp	Час створення
updated_at	-	-	timestamp	Час оновлення

Таблиця 2.32.

Таблиця “infos”

Поле	PK	FK	Тип даних	Призначення
id	+	-	bigint unsigned	Унікальний ідентифікатор
title	-	-	varchar(255)	Назва категорії корисної інформації
hidden	-	-	tinyint(1)	Видимість (0/1)
created_at	-	-	timestamp	Час створення
updated_at	-	-	timestamp	Час оновлення

Таблиця 2.33.

Таблиця “extras”

Поле	PK	FK	Тип даних	Призначення
id	+	-	bigint unsigned	Унікальний ідентифікатор

Продовження таблиці 2.33.

body	-	-	text	Вміст допису
media_type	-	-	varchar(255)	Тип медіа (youtube/image/video/audio)
url	-	-	varchar(255)	Посилання на медіа
title	-	-	varchar(255)	Заголовок
extras_topic_id	-	+	bigint	Ідентифікатор категорії дописів
hidden	-	-	tinyint(1)	Видимість (0/1)
created_at	-	-	timestamp	Час створення

Таблиця 2.34.

Таблиця “extras_topisc”

Поле	PK	FK	Тип даних	Призначення
id	+	-	bigint unsigned	Унікальний ідентифікатор
title	-	-	varchar(255)	Назва категорії додаткових дописів
hidden	-	-	tinyint(1)	Видимість (0/1)
created_at	-	-	timestamp	Час створення
updated_at	-	-	timestamp	Час оновлення

Таблиця 2.35.

Таблиця “areas_activities”

Поле	PK	FK	Тип даних	Призначення
id	+	-	bigint unsigned	Унікальний ідентифікатор
title	-	-	varchar(255)	Заголовок статті про напрямок діяльності
img_src	-	-	varchar(255)	Зображення
hidden	-	-	tinyint(1)	Видимість (0/1)

Продовження таблиці 2.35.

created_at	-	-	timestamp	Час створення
updated_at	-	-	timestamp	Час оновлення

Таблиця 2.36.

Таблиця “password_reset”

Поле	PK	FK	Тип даних	Призначення
id	+	-	bigint unsigned	Унікальний ідентифікатор
email	-	-	varchar(255)	Електронна пошта облікового запису скидання паролю.
Token	-	-	varchar(255)	Токен для скидання паролю облікового запису
created_at	-	-	timestamp	Час створення

Таблиця 2.37.

Таблиця “static”

Поле	PK	FK	Тип даних	Призначення
id	+	-	bigint unsigned	Унікальний ідентифікатор
key	-	-	varchar(255)	Елемент головної сторінки (main_image/main_body)
value	-	-	text	Вміст елемента
created_at	-	-	timestamp	Час створення

Таблиця 2.38.

Таблиця “failed_jobs”

Поле	PK	FK	Тип даних	Призначення
id	+	-	bigint unsigned	Унікальний ідентифікатор

Продовження таблиці 2.38.

uuid	-	-	varchar(255)	Ідентифікатор завдання
connection	-	-	text	Ім'я завдання
queue	-	-	text	Ім'я черги, в якій знаходилося завдання
payload	-	-	longtext	Повний вміст завдання
exception	-	-	longtext	Повідомлення про exception, що став причиною невдачі
failed_at	-	-	timestamp	Дата і час, коли завдання не вдалось виконати

Таблиця 2.39.

Таблиця “personal_access_token”

Поле	PK	FK	Тип даних	Призначення
id	+	-	bigint unsigned	Унікальний ідентифікатор
tokenable_type	-	-	varchar(255)	Модель (тип), до якої належить токен
name	-	-	varchar(255)	Ім'я токена
token	-	-	varchar(64)	Токен
abilities	-	-	text	Дії і можливості, дозволені токену
last_used_at	-	-	timestamp	Дата і час останнього використання
expires_at	-	-	timestamp	Дата і час втрати дієвості токена
created_at	-	-	timestamp	Час створення
updated_at	-	-	timestamp	Час оновлення

Таблиця “migrations”

Поле	PK	FK	Тип даних	Призначення
id	+	-	bigint unsigned	Унікальний ідентифікатор
migration	-	-	varchar(255)	Ім'я міграції, яка була виконана
batch	-	-	int	Номер партії, до якої належить міграції

Першим у реалізації бази даних кроком є створення моделей у Laravel. Моделі представляють таблиці БД і визначають їх поля та атрибути. Для створення моделі виконується наступна команда (на прикладі User):

```
php artisan make:model User
```

Файли класів моделей містяться у каталозі app/models. Після створення моделі, необхідно визначити її поля:

```
namespace App\Models;
use Illuminate\Database\Eloquent\Factories\HasFactory;
class User extends Authenticatable
{
    use HasApiTokens, HasFactory, Notifiable;
    protected $fillable = [
        'name',
        'email',
        'password',
        'first_name',
        'last_name',
        'phone',
        'social_type',
        'social_phone',
        'admin'
    ];
    protected $hidden = [
        'password',
        'remember_token'
    ];
    protected $casts = [
        'email_verified_at' => 'datetime',
    ];
    public function setPasswordAttribute($password)
    {
        $this->attributes['password'] = bcrypt($password);
    }
}
```

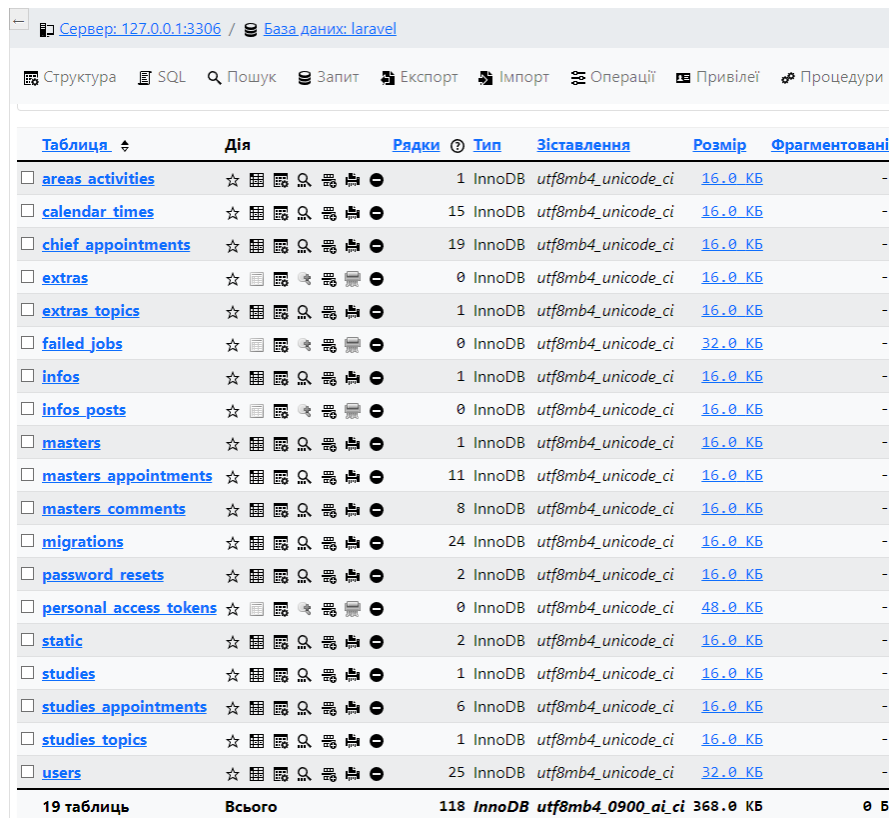
Наступним кроком є створення міграцій для таблиць бази даних. Міграції дозволяють визначити структуру та забезпечити автоматичне створення цих таблиць у базі даних MySQL. Міграціями можна вносити зміни до структури БД протягом усього життєвого циклу проєкту, тому, їх використання знадобитися не лише на цьому етапі розробки платформи. Для створення міграції використовується команда:

```
php artisan make:migration User
```

У каталозі database/migrations згенерується файл моделі, який знадобиться для міграції. Такі моделі і міграції потрібно створити для усіх таблиць бази за аналогічним сценарієм, після чого, міграцію можна буде виконати:

```
php artisan migrate
```

Команда виконає усі міграції, створивши необхідні таблиці у базі даних. Після виконання цих дій для кожної таблиці, структуру готової бази можна буде переглянути у інтерфейсі, що надає phpMyAdmin.



The screenshot shows the phpMyAdmin interface for a database named 'laravel'. The 'Tables' tab is selected, displaying a list of 19 tables. Each table entry includes a checkbox, the table name, a star icon, a table structure icon, a table data icon, a table relationships icon, a table options icon, the number of rows, the storage engine (InnoDB), the character set (utf8mb4_unicode_ci), the table size, and a link to the table's structure.

Таблиця	Дія	Рядки	Тип	Зіставлення	Розмір	Фрагментовані
☐ areas_activities	☆	1	InnoDB	utf8mb4_unicode_ci	16.0 KB	-
☐ calendar_times	☆	15	InnoDB	utf8mb4_unicode_ci	16.0 KB	-
☐ chief_appointments	☆	19	InnoDB	utf8mb4_unicode_ci	16.0 KB	-
☐ extras	☆	0	InnoDB	utf8mb4_unicode_ci	16.0 KB	-
☐ extras_topics	☆	1	InnoDB	utf8mb4_unicode_ci	16.0 KB	-
☐ failed_jobs	☆	0	InnoDB	utf8mb4_unicode_ci	32.0 KB	-
☐ infos	☆	1	InnoDB	utf8mb4_unicode_ci	16.0 KB	-
☐ infos_posts	☆	0	InnoDB	utf8mb4_unicode_ci	16.0 KB	-
☐ masters	☆	1	InnoDB	utf8mb4_unicode_ci	16.0 KB	-
☐ masters_appointments	☆	11	InnoDB	utf8mb4_unicode_ci	16.0 KB	-
☐ masters_comments	☆	8	InnoDB	utf8mb4_unicode_ci	16.0 KB	-
☐ migrations	☆	24	InnoDB	utf8mb4_unicode_ci	16.0 KB	-
☐ password_resets	☆	2	InnoDB	utf8mb4_unicode_ci	16.0 KB	-
☐ personal_access_tokens	☆	0	InnoDB	utf8mb4_unicode_ci	48.0 KB	-
☐ static	☆	2	InnoDB	utf8mb4_unicode_ci	16.0 KB	-
☐ studies	☆	1	InnoDB	utf8mb4_unicode_ci	16.0 KB	-
☐ studies_appointments	☆	6	InnoDB	utf8mb4_unicode_ci	16.0 KB	-
☐ studies_topics	☆	1	InnoDB	utf8mb4_unicode_ci	16.0 KB	-
☐ users	☆	25	InnoDB	utf8mb4_unicode_ci	32.0 KB	-
19 таблиць		Всього	118 InnoDB	utf8mb4_0900_ai_ci	368.0 KB	0 Б

Рис 2.3 – База даних у phpMyAdmin

2.3 Розробка системних компонентів

Веб-маршрути платформи у проекті Laravel зберігаються у вигляді списку у файлі `routes/web.php`. Тут маршрути зручно підв'язуються до контролерів та їх методів, які вже потім з представленнями генерують користувачам веб-сторінки.

```
Route::get("/admin/create-info-post", [InfosController::class, 'createPost']);
Route::get("/admin/edit-info-post/{id}", [InfosController::class, 'editPost']);
Route::post("/admin/save-info-post", [InfosController::class, 'savePost']);
Route::post("/admin/info-post-change-visible", [InfosController::class,
'changeVisiblePost']);
Route::post("/admin/info-post-delete", [InfosController::class, 'deletePost']);
Route::get('/get-infos-posts', [InfosController::class, 'getPosts']);
Route::get('/infos-posts/{id}', [InfosController::class, 'infosPosts']);
Route::get('/get-studies', [StudiesController::class, 'getStudies']);
Route::get("/studies/{id}", [StudiesController::class, 'studies']);
Route::get('/login', [SessionsController::class, 'create']);
Route::post('/login', [SessionsController::class, 'store']);
Route::get('/logout', [SessionsController::class, 'destroy']);
Route::get("/admin/get-users", [UserController::class, 'asyncUsers']);
Route::get("/admin/change-admin-users", [UserController::class, 'changeUsers']);
```

У файлі `routes/api.php` зберігається API Laravel Sanctum, що використовується для системи автентифікації користувачів на основі токенів.

```
Route::middleware('auth:sanctum')->get('/user', function (Request $request) {
    return $request->user();
});
```

Sanctum дозволяє видавати токени API / персональні токени доступу, які можуть використовуватися для автентифікації API-запитів до платформи. При виконанні запитів з використанням токенів, токен має бути включений в заголовок `Authorization`. Щоб видати токен для користувачів, модель `User` повинна використовувати трейт `Laravel\Sanctum\HasApiTokens`.

```
use Laravel\Sanctum\HasApiTokens;
class User extends Authenticatable
{
    use HasApiTokens, HasFactory, Notifiable;
}
```

Обробка GET чи POST запиту до веб-маршруту виконується контролерами. Контролер відповідальний за обробку авторизації називається `SessionController`.

```
class SessionsController extends Controller
{
    public function create()
    {
```

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						42
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        if (auth()->check()) return abort(404);
        return $this->view('auth.login');
    }

    public function store()
    {
        if (auth()->check()) return abort(404);
        if (auth()->attempt(request(['email', 'password'])) == false) {
            return back()->withInput(Request::except('password'))->withErrors([
                'message' => 'Електронна пошта чи пароль неправильні.'
            ])->with('login', true);
        }
        return redirect()->to('/');
    }

    public function destroy()
    {
        if (!auth()->check()) return redirect()->to('/');
        auth()->logout();

        return redirect()->to('/');
    }
}

```

Метод `create()` в `SessionController` це найпростіша обробка GET-запиту `/login` з перескеруванням на `view auth.login`. Метод `store()` реагує на POST-запит `/login` і виконує алгоритм входу в систему, включаючи обробку невдалого сценарію входу. `destroy()`, відповідно, здійснює вихід з системи. У коді був використаний впроваджений фреймворком глобальний хелпер `auth()`, який повертає екземпляр автентифікатора. До `auth()` зручно звертатися для керування автентифікацією в усьому коді, спрощуючи розробку та сам програмний код для читання та розуміння.

Розглянемо контролер `MastersController`, що відповідає за логіку обробки запитів на сторінки, пов'язані зі вчителями. Основні методи контролера:

1. `create()`: обробляє маршрут `/admin/create-master` (форма створення адміністратором нового вчителя), перевіряє автентифікацію та роль адміністратора в токени. При невинонанні умов токена повертає помилку, в іншому випадку повертає представлення `masters.edit`.

```

public function create()
{
    if (!auth()->check() || !auth()->user()->admin) return abort(404);
    return $this->view('masters.edit');
}

```

2. edit(Request \$request, \$id): обробляє маршрут /admin/edit-master/{id} (редагування вчителя адміністратором за його id), метод також перевіряє права адміністратора та вхід, шукає запис про вчителя за переданим ідентифікатором \$id у базі даних. Якщо рядок про вчителя у базі знайдено, метод повертає представлення masters.edit з переданими даними про вчителя, дані якого потрібно відредагувати.

```
public function edit(Request $request, $id)
{
    if (!auth()->check() || !auth()->user()->admin) return abort(404);
    $master = Master::where('id', '=', $id)->first();

    if (!$master) return abort(404);

    return $this->view('masters.edit', [
        'first_name' => $master->first_name,
        'last_name' => $master->last_name,
        'id' => $master->id,
        'description' => $master->description,
        'img_src' => $master->img_src,
    ]);
}
```

3. save(Request \$request): обробляє POST-запит за маршрутом /admin/save-master (заповнення адміністратором форми про вчителя). Отримуються вхідні дані з запиту у формі \$data, валідуються, та зберігаються чи перезаписуються, в залежності від того, був створений новий профіль вчителя чи відредагований існуючий.

```
public function save(Request $request)
{
    if (!auth()->check() || !auth()->user()->admin) return abort(404);
    $data = $request->input();
    if (!array_key_exists('id', $data)) {
        $data['id'] = '-1';
    }
    if (!array_key_exists('first_name', $data) || !array_key_exists('last_name', $data) || !array_key_exists('description', $data) || !array_key_exists('img_src', $data) || !$data['first_name'] || strlen($data['first_name']) < 1 || !$data['last_name'] || strlen($data['last_name']) < 1 || !$data['description'] || strlen($data['description']) < 1 || !$data['img_src'] || strlen($data['img_src']) < 1) return back()->withInput(\Illuminate\Support\Facades\Request::except(''))->withErrors("Жодне поле не може бути порожнім!");

    try {
        $finedByData = Master::where(["id", "!=", $data['id']], "first_name" =>
```

```

$data['first_name'], "last_name" => $data['last_name']])>first();
    if ($findedByData)
        return back()-
>withInput(\Illuminate\Support\Facades\Request::except(''))->withErrors([
    'error' => 'Вчитель з такими даними уже існує: <a href="' .
url("/admin/edit-master/" . $findedByData->id) . "'>' . $findedByData->last_name
. ' ' . $findedByData->first_name . '</a>'
]);
    $findedByData = Master::where(["first_name" => $data['first_name'],
"last_name" => $data['last_name']])>first();
    if ($findedByData && $findedByData['id'] == $data['id'] &&
$data['img_src'] == $findedByData['img_src'] && $data['description'] ==
$findedByData['description'] && $data['first_name'] ==
$findedByData['first_name'] && $data['last_name'] == $findedByData['last_name'])
        return back()-
>withInput(\Illuminate\Support\Facades\Request::except(''));
    $master = Master::firstOrCreate(['id' => $data['id']]);
    $master->img_src = $data['img_src'];
    $master->description = $data['description'];
    $master->first_name = $data['first_name'];
    $master->last_name = $data['last_name'];
    $master->save();

    return redirect()->to('/admin/edit-master/' . $master->id)-
>with('success', 'Майстра збережено успішно!');
} catch (\Exception $e) {
    file_put_contents("log.txt", $e->getMessage());
    return back()-
>withInput(\Illuminate\Support\Facades\Request::except(''))->withErrors([
    'message' => $e->getMessage()
]);
}
}
}

```

4. delete(Request \$request): обробляє POST-запит за маршрутом /admin/master-delete (адміністратор видаляє вчителя). Метод перевіряє права, а потім видаляє дані про вчителя з бази даних за вказаним ідентифікатором.

```

public function delete(Request $request)
{
    if (!auth()->check() || !auth()->user()->admin) return abort(404);
    $data = json_decode($request->getContent(), true);
    if (!array_key_exists('id', $data)) return json_encode(["error" => true,
"message" => 'Вчителя не знайдено!']);

    try {
        Master::where("id", $data['id']->delete();
        return json_encode(["error" => false, "message" => 'Вчителя видалено
успішно!']);
    } catch (\Exception $e) {
        return json_encode(["error" => true, "message" => $e->getMessage()]);
    }
}

```

5. `changeVisible(Request $request)`: обробляє POST-запит за маршрутом `/admin/master-change/visible` (адміністратор натискає кнопку приховування вчителя). Щоб вчитель став прихованим, у рядку вчителя в базі даних змінюється поле `hidden`: 0 – видимий, 1 – прихований. Зміни зберігаються, повертається `message` про результат.

```
public function changeVisible(Request $request)
{
    if (!auth()->check() || !auth()->user()->admin) return abort(404);
    $data = json_decode($request->getContent(), true);
    if (!array_key_exists('id', $data)) return json_encode(["error" => true,
"message" => 'Вчителя не знайдено!']);

    try {
        $master = Master::where("id", $data["id"])->first();
        $master->hidden = !$master->hidden;
        $master->save();

        $full_name = $master->first_name . " " . $master->last_name;

        $message = 'Вчитель більше не відображається у користувачів!';
        if (!$master->hidden) $message = 'Вчитель знову відображається у
користувачів!';

        return json_encode(["error" => false, "message" => $message, 'hidden' =>
$master->hidden]);
    } catch (\Exception $e) {
        return json_encode(["error" => true, "message" => $e->getMessage()]);
    }
}
```

6. `masters()`: обробляє маршрут `/masters` (сторінка вчителів). Метод не перевіряє автентифікацію, бо сторінка доступна для усіх, і повертає представлення `masters.all`, передаючи до нього дані про усіх вчителів, отримані з бази даних.

```
public function masters()
{
    $masters = Master::where('hidden', false)->get();
    return $this->view('masters.all', ['masters' => $masters]);
}
```

7. `master(Request $request, $id)`: обробляє маршрут `/master/{id}` (сторінка вчителя). Метод теж не перевіряє автентифікацію і повертає представлення `masters.index`, передаючи до нього дані про відповідного вчителя, отримані з бази даних.

Таким чином реалізовано сімнадцять контролерів:

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						46
Змн.	Арк.	№ докум.	Підпис	Дата		

1. AreasActivityController: обробляє запити, пов'язані з напрямками діяльності.

2. CalendarController: обробляє запити, пов'язані з календарем для запису до керівника.

3. ChiefAppointmentsController: обробляє запити, пов'язані з записами до керівника.

4. Controller: батьківський контроллер для усіх контроллерів, що містить метод view, який дозволяє відправити представлення користувачу (патерн Фабрика).

5. ExtrasController: обробляє запити, пов'язані з додатковою інформацією.

6. InfosController: обробляє запити, пов'язані з корисною інформацією.

7. MainController: обробляє запити адміністратора до вмісту системи.

8. MastersAppointmentsController: обробляє запити, пов'язані з записами до вчителів.

9. MasterCommentsController: обробляє запити, пов'язані з відгуками до вчителів.

10. MastersController: обробляє запити, пов'язані зі вчителями.

11. MediaController: контролер збереження медіафайлів у сховищі.

12. PasswordController: обробляє запити, пов'язані зі скиданням пароля.

13. RegistrationController: обробляє запити, пов'язані з реєстрацією.

14. SessionController: обробляє запити, пов'язані з автентифікацією.

15. StudiesAppointmentsController: обробляє запити, пов'язані з записами на майстер-класи.

16. StudiesController: обробляє запити, пов'язані з майстер-класами.

17. UserController: обробляє запити, пов'язані з користувачами.

Зупинимось на контролері MediaController. Контролер обробляє POST-запит /file-save. Він необхідний, щоб зберігати медіафайли дописів (відео, аудіо, зображення) та фото вчителів. Для цього у контролері розроблений

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						47
Змн.	Арк.	№ докум.	Підпис	Дата		

метод `save(Request $request)`: файлу надається унікальне ім'я, після чого файл збережеться у `storage`.

```
public function save(Request $request)
{
    try {
        $file = $request->file("file");
        if (is_null($file)) return json_encode(["filename" => 'null']);
        $filename = time() . "_" . $file->getClientOriginalName();
        Storage::putFileAs('public/' . $request->input('type') . 's/', $file,
        $filename);
        return json_encode(["filename" => $filename]);
    } catch (\Exception $e) {
        return json_encode(["filename" => 'null']);
    }
}
```

У формах, де є можливість завантажити медіа, завжди присутній спеціальний ідентифікатор. Якщо jQuery функція знайде його, буде викликаний метод завантаження, що генерує `FormData` та запускає метод `setFile(body, callback)`, який викликає попередньо описаний обробник запиту збереження файлу за допомогою `fetch API`.

```
function setFile(body, callback) {

document.querySelectorAll(".custom_loader").forEach(elem=>elem.classList.add("vis
ible"));
    fetch('{{url('/file-save')}}', {
        headers: {
            'X-CSRF-TOKEN': <?=json_encode(csrf_token())?>
        },
        method: 'POST',
        body
    }).then(res => res.json()).then(data =>
callback(data.filename)).finally(()=>document.querySelectorAll(".custom_loader").
forEach(elem=>elem.classList.remove("visible")));
}
```

```
function setImage() {
    const input = $("#img_input");
    const fd = new FormData;

    if (input.prop('files').length < 1) return;

    fd.append('file', input.prop('files')[0]);
    fd.append('type', 'image');

    setFile(fd,
        url => {
            document.querySelector("#img_src").value = url;
            document.querySelector("#img_src").dispatchEvent(new Event("change",
{
                bubbles: !0,
```

```

        cancelable: !1
    }));
});
}
$("#img_input").on("change", function () {
    setImage();
})

```

Розглянемо також контроллер PasswordsController. Його особливість полягає в тому, що він використовує Façade (статичний інтерфейс) Mail для відправки посилання представлення для скидання паролю на основі токєну на електронну пошту (токєни зберігаються в базі даних). Для того, щоб інтерфейс працював, необхідний SMTP-сервер та його облікові дані у файлі .env. За цей функціонал у контролері відповідає функція submitForgetPasswordForm(Request \$request):

```

public function submitForgetPasswordForm(Request $request)
{
    if (auth()->check()) return abort(404);
    $request->validate([
        'email' => 'required|email|exists:users',
    ]);

    $saved_tokens = DB::table('password_resets')->where('email', '=', $request->email);

    if ($saved_tokens->count() > 0) {
        $token = $saved_tokens->first()->token;
    } else {
        $token = Str::random(64);

        DB::table('password_resets')->insert([
            'email' => $request->email,
            'token' => $token,
            'created_at' => Carbon::now()
        ]);
    }

    Mail::send('email.forget-password', ['token' => $token], function ($message)
    use ($request) {
        $message->to($request->email);
        $message->subject('Скидання паролю');
    });

    return redirect()->to('/')->with('message', 'Вам на пошту відправлення посилання на скидання паролю!');
}

```

Представлення (views) зберігаються у каталозі /resources/views. Каталог розбитий на підкаталоги, імена котрих відповідають першим частинам веб-маршрутів. Таким чином, представлення розбиті на layouts (header, footer, file-

loader, calendar, menu, main-header), admin, areas-activities, auth, mail, extras, infos, masters, studies, users. Формат представлень: .blade.php (підтримується шаблонізатором Blade, що поставляє Laravel).

Висновки до розділу 2

Було здійснено комплексне проектування веб-платформи для занять музикою. Проектування структури системи включало визначення особливостей створюваної платформи, а також створення UML-діаграми випадків використання, яка чітко визначає взаємодію неавторизованих користувачів, авторизованих користувачів та адміністраторів із системою. Це дозволило детально описати функціональні можливості платформи та забезпечити повноту проектування.

У ході розробки бази даних визначено її структуру, що включає 19 таблиць для зберігання різноманітної інформації, такої як користувачі, вчителі, записи на майстер-класи тощо. Було створено діаграму сутність-зв'язок, яка візуалізує компоненти бази даних та їх взаємозв'язки. Для реалізації бази даних використовувалися моделі Laravel, а автоматичне створення таблиць у MySQL здійснювалося за допомогою міграцій.

Крім того, у розділі розглядалася розробка системних компонентів, зокрема налаштування веб-маршрутів, контролерів та представлень. Було описано використання Laravel Sanctum для аутентифікації користувачів на основі токенів, а також інші методи та взаємодії.

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						50
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ ВЕБ-ДОДАТКУ

3.1 Порядок встановлення та налаштування системи

Для розгортання проекту в продуктивному середовищі необхідно налаштувати веб-сервер для обслуговування системи. Для цього можна використати програмне середовище Open Server Panel, яке пропонує великий набір серверного програмного забезпечення.

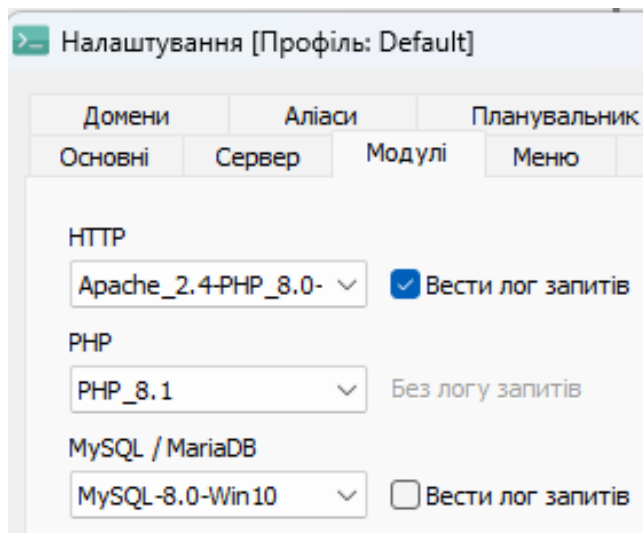


Рис 3.1. – Налаштування PHP, веб-сервера та бази даних в OSPanel

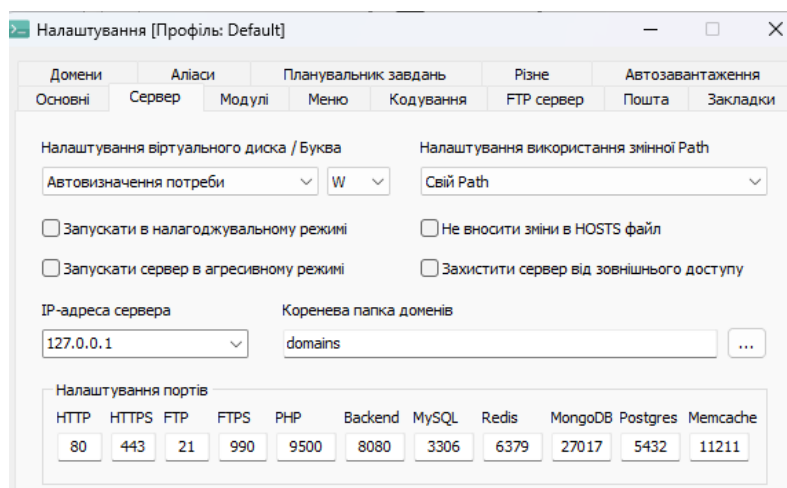


Рис 3.2. – Налаштування IP-адреси сервера та коренева папка доменів

Для локального розгортання системи спершу необхідно клонувати репозиторій проекту з SCM GitLab, виконавши наступні команди у командному рядку:

```
git clone https://link
```

cd repo

Після клонування репозиторію з'явиться каталог проєкту з наступною файловою структурою:

```
C:\Windows\System32\cmd.exe
Directory of D:\OSPanel\domains\diplom

18.06.2024  00:20    <DIR>          .
18.06.2024  00:29    <DIR>          ..
04.03.2023  15:14                258 .editorconfig
16.06.2024  18:17                1 160 .env
04.03.2023  15:14                179 .gitattributes
04.03.2023  15:14                227 .gitignore
18.06.2024  00:55    <DIR>          .idea
18.06.2024  00:20    <DIR>          app
14.06.2024  16:33                1 686 artisan
18.06.2024  00:20    <DIR>          bootstrap
04.03.2023  15:14                1 850 composer.json
04.03.2023  15:14          300 567 composer.lock
18.06.2024  00:20    <DIR>          config
18.06.2024  00:20    <DIR>          database
04.03.2023  15:14                164 deploy.txt
18.06.2024  00:20    <DIR>          lang
04.03.2023  15:14                286 package.json
04.03.2023  15:14                1 175 phpunit.xml
18.06.2024  00:31    <DIR>          public
04.03.2023  15:14                3 028 q
04.03.2023  15:14                4 158 README.md
18.06.2024  00:20    <DIR>          resources
18.06.2024  00:20    <DIR>          routes
18.06.2024  00:20    <DIR>          storage
18.06.2024  00:20    <DIR>          tests
18.06.2024  00:21    <DIR>          vendor
04.03.2023  15:14                263 vite.config.js
               13 File(s)          315 001 bytes
               14 Dir(s)      183 781 974 016 bytes free
```

Рис 3.3. – Файли та каталоги проєкту

Після клонування репозиторію, необхідно встановити всі залежності проєкту за допомогою менеджера пакетів Composer, щоб завантажити усі бібліотеки та пакети, необхідні для роботи системи.

composer install

Налаштування параметрів системи містяться у файлі .env.example. Можна перейменувати файл на .env, або створити новий і налаштувати список змінних:

1. APP_NAME: назва додатку.
2. APP_ENV: середовище роботи (static, production, staging).
3. APP_KEY: ключ додатка для шифрування.
4. APP_DEBUG: включення або виключення режиму налагодження.
5. APP_URL: URL додатку.
6. DB_CONNECTION: тип з'єднання з базою даних.
7. DB_HOST: хост сервера БД.

8. DB_PORT: порт сервера БД.
9. DB_USERNAME: ім'я користувача для підключення до БД.
10. DB_PASSWORD: пароль для підключення до БД.
11. MAIL_MAILER: тип сервісу для відправки пошти.
12. MAIL_HOST: хост SMTP сервера.
13. MAIL_PORT: порт SMTP сервера.
14. MAIL_USERNAME: ім'я для SMTP.
15. MAIL_PASSWORD: пароль SMTP сервера.
16. MAIL_ENCRYPTION: тип шифрування.
17. MAIL_FROM_ADRESS: пошта відправника.
18. MAIL_FROM_NAME: ім'я відправника.

Ключ додатка генерується викликом наступної команди:

```
php artisan key:generate
```

Тепер потрібно виконати міграції:

```
php artisan migrate
```

Наступна команда створить символічне посилання на сховище storage:

```
php artisan storage:link
```

Нарешті, запуск сервера за адресою `http://localhost:8000`

```
php artisan serve
```

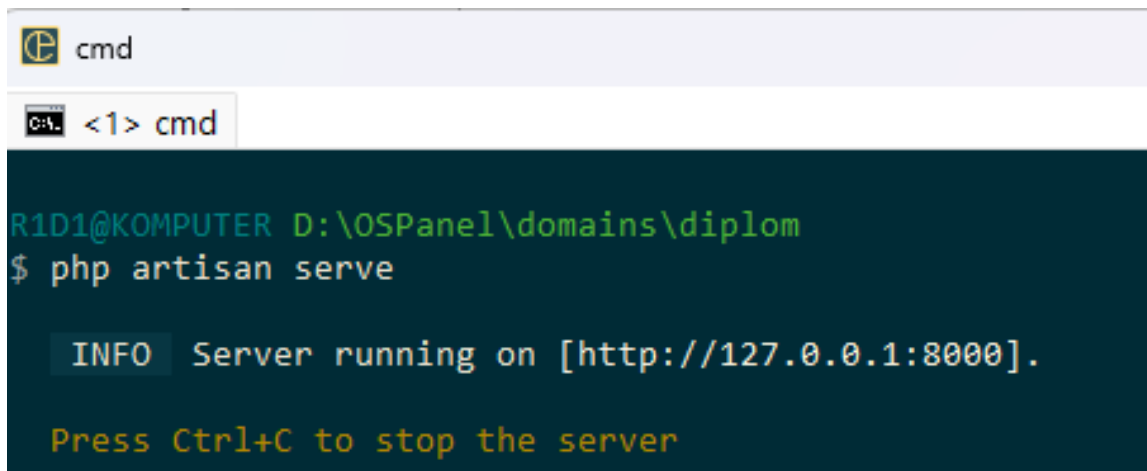


Рис 3.4. – Запуск локального сервера через консоль OSPanel.

3.2 Інтерфейс платформи

Розглянемо основні елементи інтерфейсу платформи занять музикою. Базовою частиною будь-якої веб-системи є форма для авторизації:

The image shows a login form titled "Вхід" (Login) on a yellow background. It contains two input fields: "Пошта" (Email) and "Пароль" (Password). Below these fields are three buttons: "Увійти" (Login), "Реєстрація" (Registration), and "Забули пароль?" (Forgot password?).

Рис 3.5. – Форма входу

Звідси гість може перейти на форму реєстрації, якщо в нього немає облікового запису, або відновити пароль.

The image shows a registration form titled "Зареєструватися" (Register) on a yellow background. It contains several input fields: "Ім'я" (Name), "Прізвище" (Surname), "Пошта" (Email), "Телефон" (Phone), "Оберіть соц. мережу" (Select social network) with a dropdown arrow and a social media icon, "Телефон соц. мережі" (Social network phone), "Пароль" (Password), and "Повторіть пароль" (Repeat password). A "Зареєструватися" (Register) button is at the bottom.

Рис 3.6. – Форма реєстрації

Забули пароль?

На вказану вами пошту прийде лист для скидання паролю.
Якщо ви не побачите листа, то перевірте будь-ласка спам.

ollbutchersmail@gmail.com

Відправити

Рис 3.7. – Форма скидання паролю

Після вказання електронної пошти, на яку акаунт зареєстрований, надішлеться лист з посиланням для відновлення.

Скидання паролю Входящие x

Vadym <ollbutchersmail@gmail.com>
кому: мне ▾

Forget Password Email

You can reset password from bellow link: [Reset Password](#)

Рис 3.8. – Лист на електронну пошту

Скинути пароль

Пошта

Пароль

Повторіть пароль

Reset Password

Рис 3.9. – Форма скидання паролю за посиланням

Авторизований користувач має розширений доступ до контенту платформи та більше доступних дій з нею. Розділи платформи розміщені на футері сторінок у вигляді бургер-меню. Розділи майстер-класів, напрямків діяльності та додаткового в свою чергу поділяються на підрозділи. Ці розділи недоступні неавторизованим користувачам.

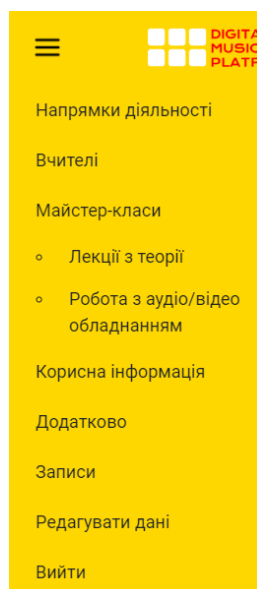


Рис 3.10. – Розділи

Сторінка «Вчителі» надає список вчителів, про яких можна дізнатися більше та записатися до них на навчання:

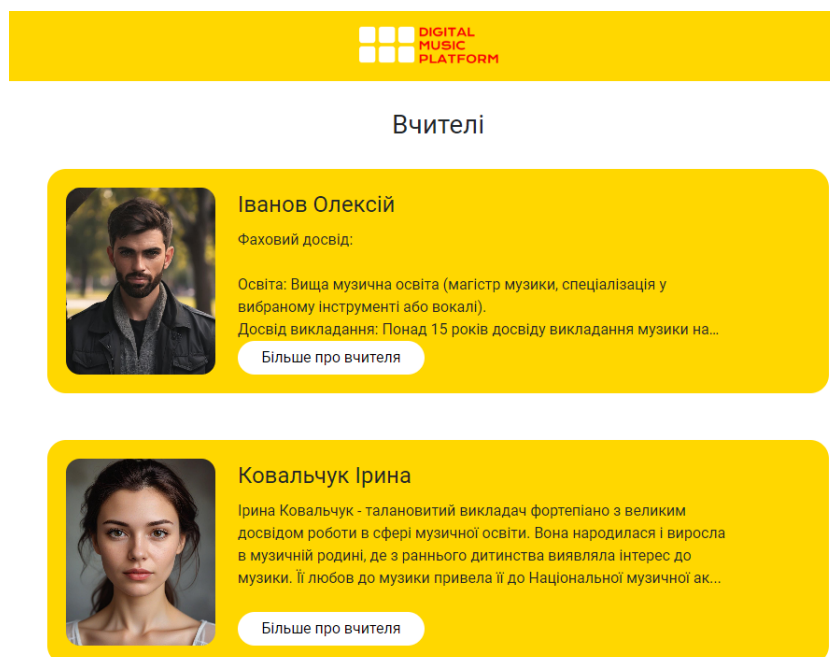
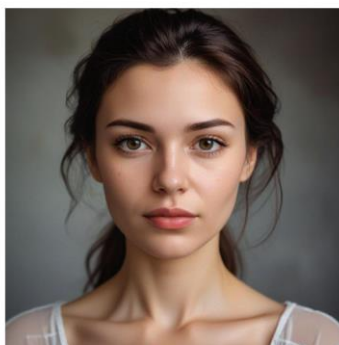


Рис 3.11. – Вчителі

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						56
Змн.	Арк.	№ докум.	Підпис	Дата		

Ковальчук Ірина



Ірина Ковальчук - талановитий викладач фортепіано з великим досвідом роботи в сфері музичної освіти. Вона народилася і виросла в музичній родині, де з раннього дитинства виявляла інтерес до музики. Її любов до музики привела її до Національної музичної академії України ім. П.І. Чайковського, де вона здобула вищу освіту з відзнакою. Після закінчення академії Ірина працювала у кількох музичних школах, де вона зарекомендувала себе як відповідальний і натхненний педагог.

Протягом своєї кар'єри Ірина працювала з учнями різного віку та рівня підготовки. Вона розробила власну методику викладання, яка дозволяє швидко та ефективно навчати гри на фортепіано, а також розвивати музичний слух і теоретичні знання. Її уроки завжди проходять у дружній та підтримуючій атмосфері, що сприяє максимально продуктивному навчанню. Ірина впевнена, що кожен учень має унікальні здібності, і її завдання як вчителя - допомогти кожному учню розкрити свій потенціал.

Ірина також активно займається науковою діяльністю, публікуючи статті та методичні розробки у галузі музичної педагогіки. Вона є постійним учасником міжнародних конференцій та семінарів, де ділиться своїм досвідом та вивчає нові методики навчання. Її робота отримала визнання серед колег, а її учні регулярно займають призові

Рис 3.12. – Сторінка вчителя

До вчителя, на його сторінці, можна записатись, натиснувши відповідну кнопку. З'явиться елемент рорир, схожий на форми авторизації, де можна підтвердити запис. На усіх формах платформи, за потреби, з'являються повідомлення про помилки чи успіх.

Запис до Ковальчук Ірина

Запис на ім'я: Вадим Нелеп
Відслідкуйте статус запису на сторінці "Мої записи", щоб потрапити на неї натисніть на меню та оберіть її

Коли адміністратор розгляне ваш запис, то статус зміниться та вам зателефонують

Ви вже відправили запит на запис до вчителя!
Зачекайте, поки адміністратор зателефонує Вам
для узгодження часу і місця!

Записатися

Рис 3.13. – Запис до вчителя з помилкою

На тій же сторінці можна залишити відгук про вчителя:

Відгуки

Тест

Надіслати

Вадим Нелеп
Класні уроки, дякую!

Рис 3.14. – Залишення відгуку

Користувач має можливість записатися до керівника за допомогою календаря. Дні з доступними годинами для запису виділені кольором. Якщо час не зайнятий, можливо записатися.

Оберіть дату та час
Доступні дати виділені жовтим, після натискання будуть
відображені години для запису.

Calendar for June 2024. The 24th and 28th are highlighted in yellow, indicating available dates.

Доступні години запису:

Час: 10:10 [Записатись](#)
Час: 10:20 [Записатись](#)
Час: 10:30 [Записатись](#)

Рис 3.15. – Запис до керівника

Коли адміністратор погодить запис, календар зникне:

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						58
Змн.	Арк.	№ докум.	Підпис	Дата		

Ви вже записані на 2024-06-28 10:10:00

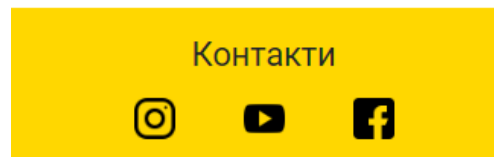


Рис 3.16. – Погоджений запис до керівника

На сторінках майстер-класів користувач може переглянути список цікавих йому майстер-класів та записатися на них.

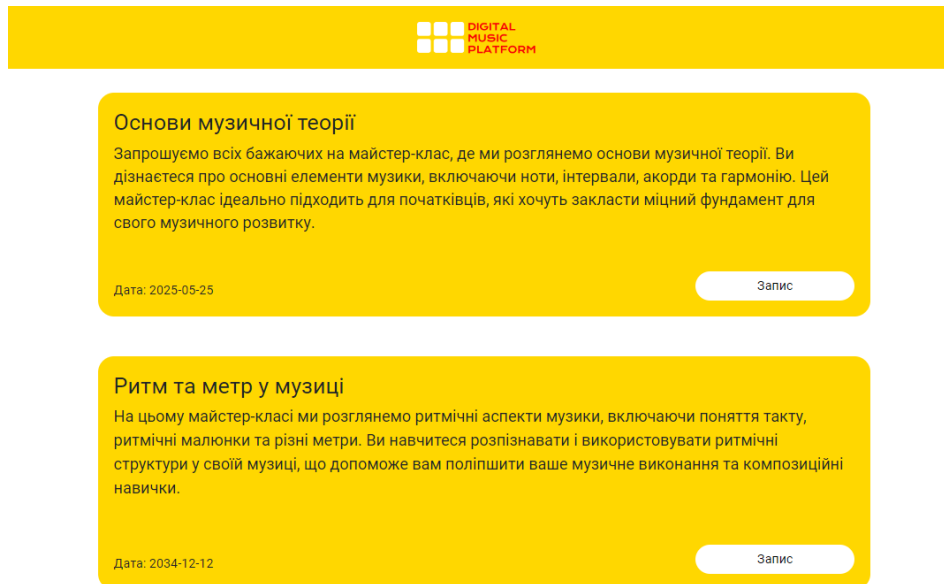


Рис 3.17. – Майстер-класи за категорією

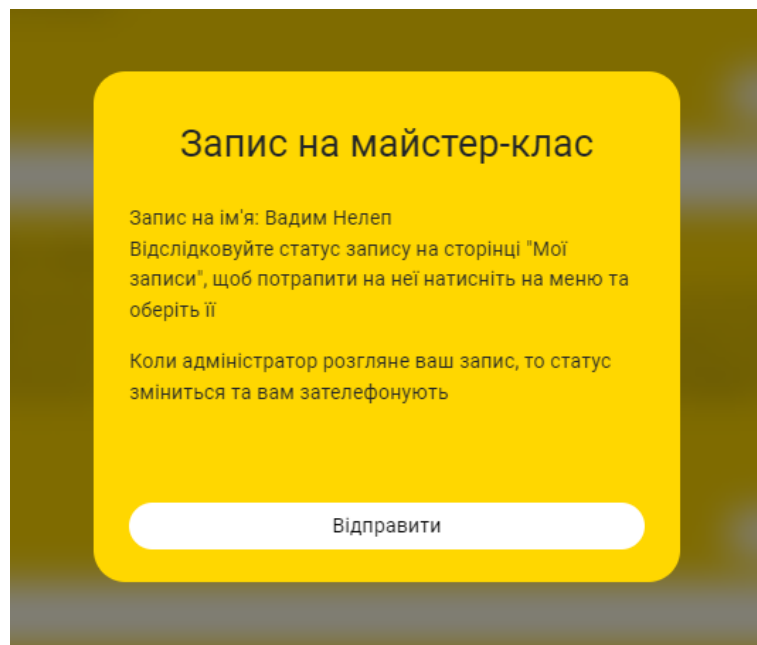


Рис 3.18. – Форма запису на майстер-клас

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						59
Змн.	Арк.	№ докум.	Підпис	Дата		

Користувач має можливість відслідковувати статус усіх своїх записів в розділі «Мої записи»:

DIGITAL MUSIC PLATFORM		
Записи		
До керівникаДо вчителівНа майстер-класи		
#	Заголовок	СтатусДата відправлення
7	Основи музичної теорії	wait_accept2024-06-18 14:43:52

Рис 3.19 – Розділ «Мої записи»

Структура розділів «Корисна інформація», «Додатково» та «Напрямки діяльності» являє собою систему дописів, з прикріпленими медіа (зображення, відеоролик, аудіофайл чи YouTube-відео).

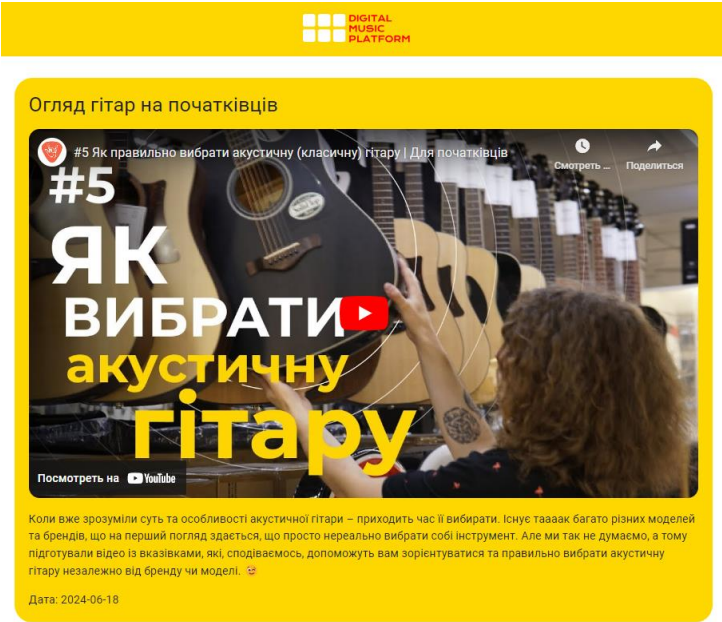


Рис 3.20. – Пост з YouTube-відео (Додатково/Огляди)



Рис 3.21. – Пост з зображенням (Корисна інформація/Учням)

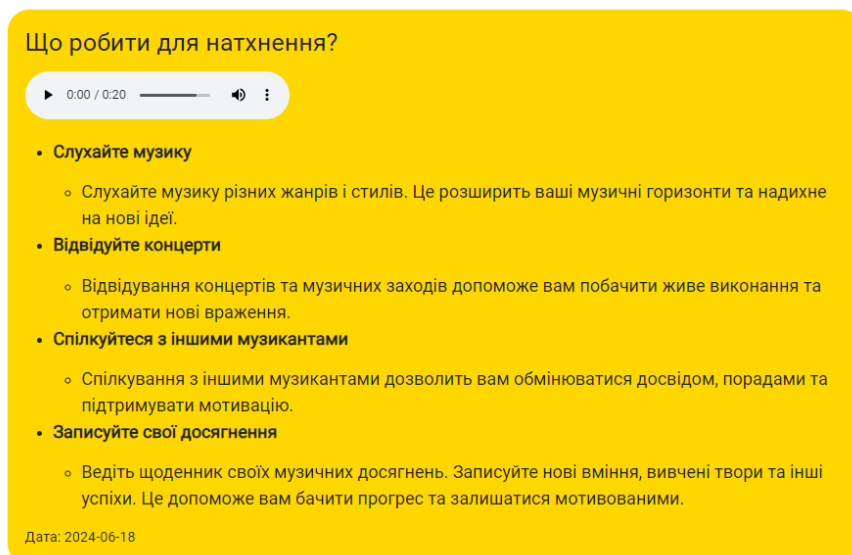


Рис 3.22. – Пост з аудіо (Корисна інформація/Учням)

Оброблювати запити на записи, а також редагувати контент розділів платформи має право адміністратор. У адміністратора є власна сторінка з панеллю доступу до розділів.



Рис 3.23. – Панель адміністратора

На головній сторінці панелі адміністратора знаходиться невелика кількісна статистична інформація:



Рис 3.24. – Статистика

На сторінці «Календар» адміністратор може додавати на видаляти години на календарі, погоджувати чи відхиляти записи:

June 2024

Sun	Mon	Tue	Wed	Thu	Fri	Sat
26	27	28	29	30	31	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	1	2	3	4	5	6

Новий час

--:--

Записи

Новий час

--:--

Час: 10:10
Клієнт: Вадим Нелен
Пошта: ollbutchersmail@gmail.com
Телефон: +380 (98) 178-00-05
Статус: accepted

Час: 10:20

Час: 10:30

Рис 3.25. – Редагування календаря

Увесь контент платформи адміністратор може створювати, видаляти та приховувати.

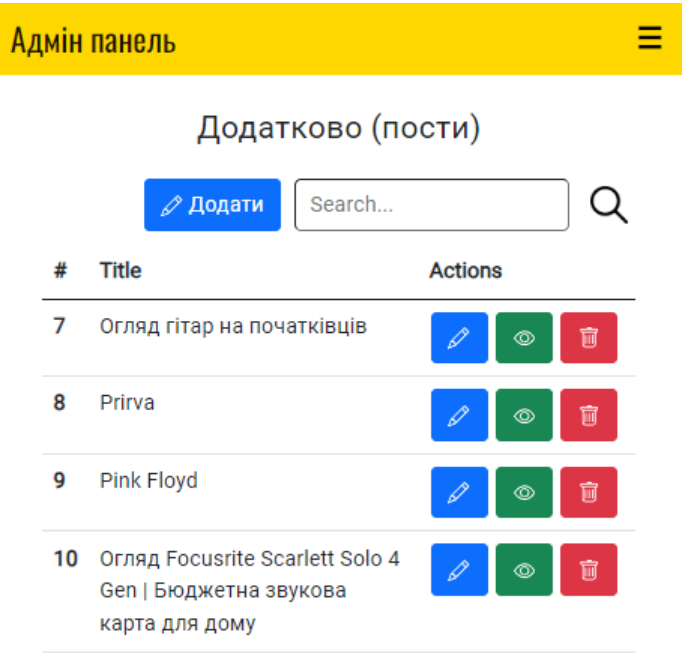


Рис 3.26. – Повний доступ до додаткових дописів

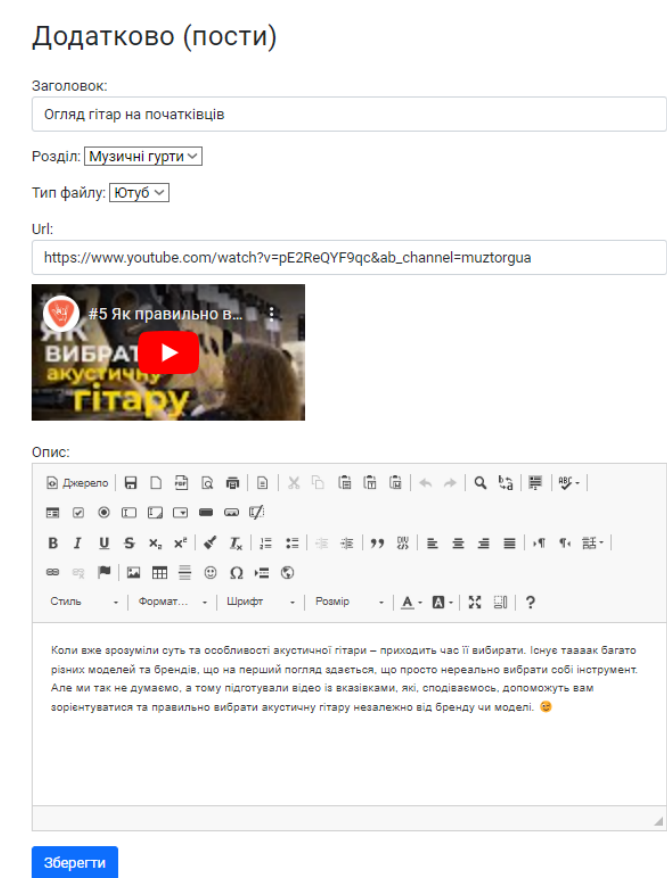


Рис 3.27. – Створення/редагування допису

Адміністратор бачить усіх користувачів та коментарі. Коментарі може видаляти, а користувачам призначати роль адміністратора, чи навпаки.

Коментарі				
Search...				
#	Master	Author	Body	Actions
15	Ірина Ковальчук	Вадим Нелен	Класні уроки, дякую!	

Користувачі				
Search...				
#	Email	First name	Last name	Actions
1	jwa67m8ui5@gmail.com	admin	admin	
28	ollbutchersmail@gmail.com	Вадим	Нелен	
29	123@123	123	123	
30	123@321	123	123	

Рис 3.28. – Користувачі, коментарі

Як і з календарем, адміністратор може погодити чи відхилити запит на майстер-клас чи до вчителя. Він бачить ці записи теж у вигляді таблиці:

Записи до вчителів				
Search...				
#	Study	User	User phone	User email
14	Ірина Ковальчук	Вадим Нелен	+380 (98) 178-00-05	ollbutchersmail@gmail.com

Рис 3.29. – Записи до вчителів

Усі прийняття чи відхилення записів супроводжуються підтверджувальними вікнами:

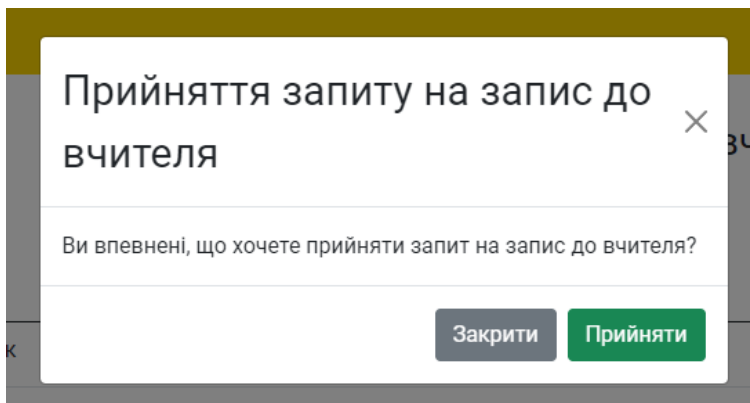


Рис 3.30. – Вікно про прийняття запису

Окремо, адміністратор може редагувати і головну сторінку:

Головна сторінка

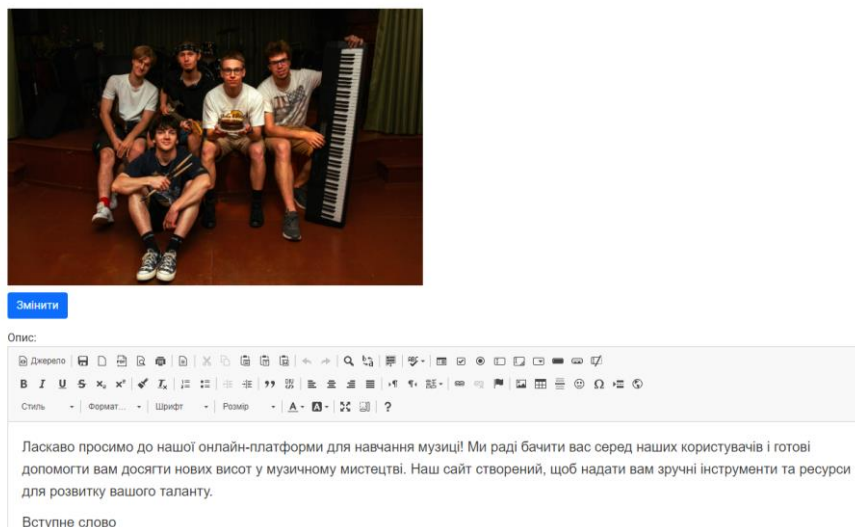


Рис 3.31. – Редагування головної сторінки

Дизайн інтерфейсу платформи розроблений з урахуванням сучасних стандартів та вимог до веб-дизайну. Основною метою є забезпечення простоти, зручності та доступності для користувачів з різним рівнем технічної підготовки. Усі елементи інтерфейсу розташовані логічно та послідовно, що дозволяє користувачам швидко орієнтуватися на платформі та знаходити необхідну інформацію.

Особливу увагу приділено адаптивності дизайну, що дозволяє платформі коректно відображатися на різних пристроях, включаючи настільні комп'ютери, планшети та смартфони.

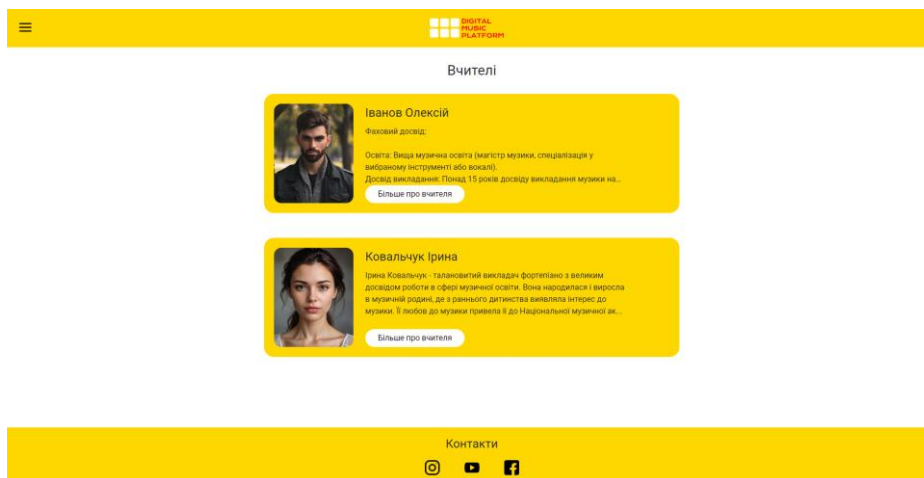


Рис 3.32. – Сторінка платформи з ПК

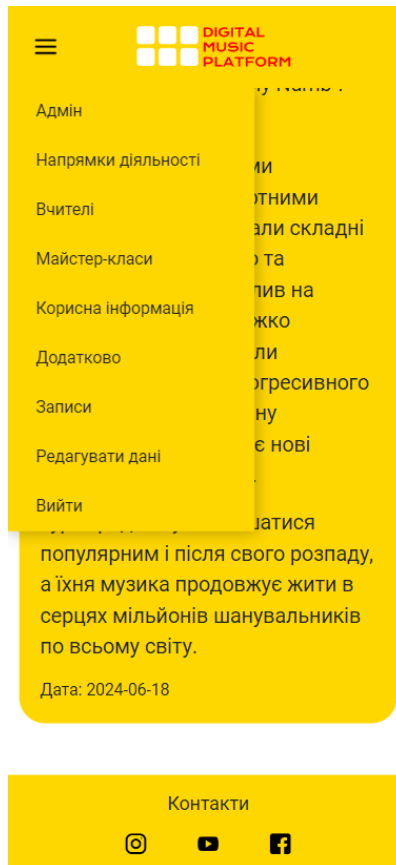


Рис 3.33 – Вигляд сторінки платформи з iPhone 12 Pro

3.3 Тестування платформи

Тестування відіграє важливу роль у розробці програмного забезпечення, оскільки дозволяє забезпечити високу якість, надійність та безпеку додатка. Воно допомагає виявити помилки на ранніх етапах розробки, що значно знижує витрати на їх виправлення в майбутньому. Тестування також гарантує, що додаток відповідає вимогам користувачів і бізнесу, підвищуючи конкурентоспроможність продукту та задоволеність клієнтів.

Процес тестування платформи для занять музикою складався з двох основних етапів: ручного тестування та автоматизованих unit-тестів. Такий підхід дозволив всебічно перевірити функціональність системи та забезпечити високу якість кінцевого продукту.

Ручне тестування використовувалося для перевірки функціональності додатка на всіх етапах розробки. Після кожного значного оновлення коду

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						66
Змн.	Арк.	№ докум.	Підпис	Дата		

здійснювалася ретельна перевірка всіх компонентів системи, щоб переконатися у відсутності помилок. Це дозволило виявляти та усувати дефекти ще до того, як вони могли вплинути на кінцевих користувачів.

Unit-тестування було використано у ході розробки моделей Laravel. Тести дозволяли перевіряти, як окремі моделі працюють у ізольованому середовищі, що сприяло швидкому виявленню помилок, яких було багато, а також перевірці коректності виконання логіки додатка. Завдяки використанню юніт-тестів ми могли впевнитися, що кожна зміна в коді не призведе до небажаних побічних ефектів у інших частинах системи.

Фреймворк Laravel поставляє модуль PHPUnit, завдяки якому можна створити Unit-тести, які зберігаються у каталозі tests/Unit.

Розглянемо тест, який перевірає роботоздатність моделі User коректно реагувати на CRUD-операції (створення, оновлення, видалення, пошук за електронною поштою, перевірка пароля). Його представляє клас UserTest, який містить публічні методи відповідних кейсів.

```
public function it_can_create_a_user()
{
    $user = User::create([
        'name' => 'Test User',
        'email' => 'test@example.com',
        'password' => 'password',
        'first_name' => 'Test',
        'last_name' => 'User',
        'phone' => '123456789',
        'social_type' => 'facebook',
        'social_phone' => '987654321',
        'admin' => false,
    ]);

    $this->assertDatabaseHas('users', ['email' => 'test@example.com']);
    $this->assertEquals('Test User', $user->name);
}
```

Метод it_can_create_a_user() створює тестовий екземпляр користувача, перевіряє, чи з'являється у базі користувач за електронною поштою, додатково порівнюючи ім'я об'єкта створеної моделі з ім'ям, за яким модель створювалась. Таким чином, метод повертає bool результату тесту.

Наступний метод перевіряє оновлення даних:

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						67
Змн.	Арк.	№ докум.	Підпис	Дата		

```

public function it_can_update_a_user()
{
    $user = User::factory()->create();
    $user->name = 'Updated Name';
    $user->save();

    $this->assertDatabaseHas('users', ['name' => 'Updated Name']);
}

```

Метод `it_can_update_a_user()` створює новий об'єкт `User` та змінює значення поля ім'я, після чого зберігає зміни і перевіряє наявність об'єкта у базі за новим іменем.

Решта методів за свою специфікою аналогічні:

```

public function it_can_delete_a_user()
{
    $user = User::factory()->create();
    $user->delete();

    $this->assertDeleted($user);
}

/** @test */
public function it_can_find_a_user_by_email()
{
    $user = User::factory()->create([
        'email' => 'findme@example.com',
    ]);

    $foundUser = User::where('email', 'findme@example.com')->first();
    $this->assertEquals($user->id, $foundUser->id);
}

/** @test */
public function it_can_check_user_password()
{
    $user = User::factory()->create([
        'password' => 'password', // The password will be automatically hashed by
the model
    ]);

    $this->assertTrue(\Hash::check('password', $user->password));
}

```

Такі Unit-тести доречно виконувати після виконання міграцій, щоб перевірити їх коректність, в перспективі перед застосуванням на віддалених серверах. Автоматизовані тести безпомилково повертають звіти про невдачу, завдяки чому можна швидко віднайти критичні фрагменти коду і виправити їх.

Результат виконання тестів для моделі `User` зображено на рис. 3.34.

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						68
Змн.	Арк.	№ докум.	Підпис	Дата		

```

R1D1@KOMPUTER D:\OSPanel\domains\diplom
$ php artisan test
Warning: TTY mode is not supported on Windows platform.

PASS Tests\Unit\ExampleTest
✓ that true is true

PASS Tests\Unit\UserTest
✓ that it can create a user
✓ that it can update a user
✓ that it delete a user
✓ that it can find a user by email
✓ that iit can check user password

PASS Tests\Feature\ExampleTest
✓ the application returns a successful response

Tests: 7 passed
Time: 0.22s

```

Рис 3.34 – Unit-тестування User

Висновки до розділу 3

Було детально розглянуто процес встановлення та налаштування системи, а також представлено інтерфейс платформи. Було визначено ключові етапи розгортання системи у програмному середовищі, включаючи налаштування веб-сервера та бази даних, клонування репозиторію проєкту, встановлення залежностей та налаштування конфігураційних файлів.

Дизайн інтерфейсу розроблений з урахуванням сучасних стандартів та вимог до веб-дизайну, забезпечуючи зручність, простоту та доступність для користувачів з різним рівнем технічної підготовки. Особливу увагу було приділено адаптивності дизайну інтерфейсу, що дозволяє платформі коректно відображатися на різних пристроях, комп'ютери, планшети та смартфони.

Було продемонстровано основні елементи інтерфейсу, включаючи форми для авторизації, реєстрації та скидання паролю, а також розділи платформи, які надають користувачам доступ до майстер-класів, списків вчителів та іншого контенту. Адміністраторська панель дозволяє

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						69
Змн.	Арк.	№ докум.	Підпис	Дата		

адміністратору керувати контентом, переглядати статистику, редагувати календар та користувачів.

Окрім цього, було розглянуто процес тестування платформи, який складався з ручного тестування та автоматизованих юніт-тестів. Ручне тестування забезпечило перевірку функціональності системи на всіх етапах розробки, тоді як юніт-тести допомогли виявити помилки у логіці додатка та забезпечити коректність виконання ключових функцій. Завдяки всебічному тестуванню вдалося досягти високої якості кінцевого продукту, забезпечивши його надійність, продуктивність та безпеку.

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						70
Змн.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

Розробка web-орієнтованої платформи для занять музикою включала всебічний аналіз потреб користувачів і ринку, що дозволило сформулювати основні вимоги до платформи. Спочатку було проведено дослідження існуючих аналогів для визначення їх сильних і слабких сторін. В результаті цього аналізу було прийнято рішення використовувати клієнт-серверну архітектуру з застосуванням PHP та Laravel для серверної частини, HTML, CSS, JavaScript і Bootstrap для клієнтської частини, а також MySQL для збереження даних.

Наступним кроком стало визначення функціональних можливостей платформи та способів взаємодії користувачів з нею. Було описано основні сценарії використання, такі як реєстрація та авторизація користувачів, запис на заняття, перегляд освітніх матеріалів та залишення відгуків. Особлива увага приділялася забезпеченню зручності використання платформи для всіх категорій користувачів.

Після цього було розроблено структуру платформи, включаючи створення діаграм бази даних і визначення взаємодії між компонентами системи. Створено базу даних, що містить таблиці для зберігання інформації про користувачів, заняття, відгуки та інші важливі дані. Після чого, у ході програмування, було забезпечено ефективну взаємодію між клієнтською та серверною частинами платформи.

Дизайн інтерфейсу було створено з урахуванням сучасних стандартів веб-дизайну, що забезпечує зручність та доступність для користувачів з різним рівнем технічної підготовки.

Завершальним етапом стало тестування платформи, яке складалося з ручного тестування та автоматизованих юніт-тестів. Це дало змогу виявити і виправити помилки, забезпечивши стабільну та надійну роботу платформи. Ручне тестування охоплювало різні сценарії використання, тоді як юніт-тести перевіряли коректність виконання основних функцій та методів.

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						71
Змн.	Арк.	№ докум.	Підпис	Дата		

В кінцевому результаті, було створено ефективну та надійну онлайн платформу для занять музикою, яка відповідає сучасним вимогам ринку та готова до впровадження.

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						72
Змн.	Арк.	№ докум.	Підпис	Дата		

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Документація Laravel [Електронний ресурс]. – 2024. – Режим доступу до ресурсу: <https://laravel.com/docs/11.x/readme>
2. Документація PHP [Електронний ресурс]. – 2024. – Режим доступу до ресурсу: <https://www.php.net/manual/en/>
3. Веб-сервер Apache [Електронний ресурс]. – 2024. – Режим доступу до ресурсу: <https://httpd.apache.org/docs/2.4/>.
4. Документація Composer [Електронний ресурс]. – 2024. – Режим доступу до ресурсу: <https://getcomposer.org/doc/>
5. Документація PHPUnit. The PHP Testing Framework [Електронний ресурс]. – 2024. – Режим доступу до ресурсу: <https://phpunit.de/documentation.html>.
6. Blade Templating Engine. Laravel 10.x Blade Templates [Електронний ресурс]. – 2024. – Режим доступу до ресурсу: <https://laravel.com/docs/10.x/blade>.
7. Документація Bootstrap [Електронний ресурс]. – 2024. – Режим доступу до ресурсу: <https://getbootstrap.com/docs/5.3/>.
8. MVC Framework - Introduction [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: https://www.tutorialspoint.com/mvc_framework/mvc_framework_introduction.htm
9. Документація artisan [Електронний ресурс]. – 2024. – Режим доступу до ресурсу: <https://laravel.com/docs/11.x/artisan>
10. Database: Migrations [Електронний ресурс]. – 2024. – Режим доступу до ресурсу: <https://laravel.com/docs/11.x/migrations>
11. Документація jQuery [Електронний ресурс]. – 2024. – Режим доступу до ресурсу: <https://api.jquery.com>
12. Документація Sanctum [Електронний ресурс]. – 2024. – Режим доступу до ресурсу: <https://laravel.com/docs/10.x/sanctum>

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						73
Змн.	Арк.	№ докум.	Підпис	Дата		

13. Документація MySQL [Електронний ресурс] . – 2024. – Режим доступу до ресурсу: <https://dev.mysql.com/doc/refman/8.0/en/>
14. Responsive Design vs. Adaptive Design [Електронний ресурс] . – 2022. – Режим доступу до ресурсу: <https://www.uxpin.com/studio/blog/responsive-vs-adaptive-design-whats-best-choice-designers/>
15. Crockford D. JavaScript: The Good Parts: Основні принципи Javascript / Douglas Crockford., 2008. – 176 с.
16. Welling L. PHP and MySQL Web Development: Використання PHP і MySQL для розробки веб-додатків / L. Welling, T. Laura., 2008. – 1008 с.
17. Josh L. Modern PHP: New Features and Good Practices: Сучасні практики розробки на PHP / Lockhart Josh., 2015. – 270 с.
18. Nathan W. Learning Laravel: The Easiest Way: Основні принципи роботи з Laravel / W. Nathan., 2020. – 325 с.
19. Craig L. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development: Використання UML і шаблонів / Larman Craig., 2004. – 736 с.
20. Jesse J. The Elements of User Experience: User-Centered Design for the Web and Beyond: Основи UX-дизайну / James Jesse., 2010. – 272 с.
21. Erich G. Design Patterns: Elements of Reusable Object-Oriented Software: Основи шаблонів проєктування / Gamma Erich., 1994. – 395 с.
22. Fowler M. Patterns of Enterprise Application Architecture: Шаблони архітектури корпоративних додатків / Martin Fowler., 2002. – 560 с.
23. Plaisant C. Designing the User Interface: Strategies for Effective Human-Computer Interaction: Стратегії взаємодії людини і комп'ютера / Catherine Plaisant., 2010. – 624 с.

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						74
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТКИ

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						75
Змн.	Арк.	№ докум.	Підпис	Дата		

MastersController.php – контролер, відповідає за логіку обробки запитів на сторінки, пов'язані зі вчителями.

```
<?php

namespace App\Http\Controllers;

use App\Models\Master;
use App\Models\User;
use App\Models\MasterComment;
use Illuminate\Http\Request;

class MastersController extends Controller
{
    public function create()
    {
        if (!auth()->check() || !auth()->user()->admin) return abort(404);
        return $this->view('masters.edit');
    }

    public function edit(Request $request, $id)
    {
        if (!auth()->check() || !auth()->user()->admin) return abort(404);
        $master = Master::where('id', '=', $id)->first();

        if (!$master) return abort(404);

        return $this->view('masters.edit', [
            'first_name' => $master->first_name,
            'last_name' => $master->last_name,
            'id' => $master->id,
            'description' => $master->description,
            'img_src' => $master->img_src,
        ]);
    }

    public function save(Request $request)
    {
        if (!auth()->check() || !auth()->user()->admin) return abort(404);

        $data = $request->input();

        if (!array_key_exists('id', $data)) {
            $data['id'] = '-1';
        }

        if (!array_key_exists('first_name', $data) ||
            !array_key_exists('last_name', $data) ||
            !array_key_exists('description', $data) ||
            !array_key_exists('img_src', $data) ||
            (!$data['first_name'] || strlen($data['first_name']) < 1) ||
            (!$data['last_name'] || strlen($data['last_name']) < 1) ||
            (!$data['description'] || strlen($data['description']) < 1) ||
            (!$data['img_src'] || strlen($data['img_src']) < 1)
        ) return back();
    }
}
```

```

>withInput(\Illuminate\Support\Facades\Request::except(''))->withErrors("Жодне
поле не може бути порожнім!");

    try {
        $findedByData = Master::where([[ "id", "!=" , $data['id']],
"first_name" => $data['first_name'], "last_name" => $data['last_name']])->
>first();
        if ($findedByData)
            return back()-
>withInput(\Illuminate\Support\Facades\Request::except(''))->withErrors([
            'error' => 'Вчитель з такими даними уже існує: <a href="' .
url("/admin/edit-master/" . $findedByData->id) . "'>' . $findedByData->last_name
. ' ' . $findedByData->first_name . '</a>'
            ]);

        $findedByData = Master::where(["first_name" => $data['first_name'],
"last_name" => $data['last_name']])->first();
        if ($findedByData && $findedByData['id'] == $data['id'] &&
$data['img_src'] == $findedByData['img_src'] && $data['description'] ==
$findedByData['description'] && $data['first_name'] ==
$findedByData['first_name'] && $data['last_name'] == $findedByData['last_name'])
            return back()-
>withInput(\Illuminate\Support\Facades\Request::except(''));

        $master = Master::firstOrNew(['id' => $data['id']]);
        $master->img_src = $data['img_src'];
        $master->description = $data['description'];
        $master->first_name = $data['first_name'];
        $master->last_name = $data['last_name'];
        $master->save();

        return redirect()->to('/admin/edit-master/' . $master->id)-
>with('success', 'Майстра збережено успішно!');
    } catch (\Exception $e) {
        file_put_contents("log.txt", $e->getMessage());
        return back()-
>withInput(\Illuminate\Support\Facades\Request::except(''))->withErrors([
            'message' => $e->getMessage()
        ]);
    }
}

public function masters()
{
    $masters = Master::where('hidden', false)->get();
    return $this->view('masters.all', ['masters' => $masters]);
}

public function delete(Request $request)
{
    if (!auth()->check() || !auth()->user()->admin) return abort(404);
    $data = json_decode($request->getContent(), true);
    if (!array_key_exists('id', $data)) return json_encode(["error" => true,
"message" => 'Вчителя не знайдено!']);

    try {

```

					КН.КР.Б – 122 – 24 – ПЗ	Арк.
						77
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        Master::where("id", $data['id'])->delete();
        return json_encode(["error" => false, "message" => 'Вчителя видалено
успішно!']);
    } catch (\Exception $e) {
        return json_encode(["error" => true, "message" => $e->getMessage()]);
    }
}

public function changeVisible(Request $request)
{
    if (!auth()->check() || !auth()->user()->admin) return abort(404);
    $data = json_decode($request->getContent(), true);
    if (!array_key_exists('id', $data)) return json_encode(["error" => true,
"message" => 'Вчителя не знайдено!']);

    try {
        $master = Master::where("id", $data["id"])->first();
        $master->hidden = !$master->hidden;
        $master->save();

        $full_name = $master->first_name . " " . $master->last_name;

        $message = 'Вчитель більше не відображається у користувачів!';
        if (!$master->hidden) $message = 'Вчитель знову відображається у
користувачів!';

        return json_encode(["error" => false, "message" => $message, 'hidden'
=> $master->hidden]);
    } catch (\Exception $e) {
        return json_encode(["error" => true, "message" => $e->getMessage()]);
    }
}

public function master(Request $request, $id)
{
    $master = Master::where('id', "=", $id)->where('hidden', false)->first();
    if (!$master) return abort(404);
    $full_comments = [];
    if (auth()->user()) {
        $comments = MasterComment::where("master_id", $id)-
>where("author_id", "=", auth()->user()->id)->get();
        foreach ($comments as $comment) {
            $author = User::where("id", '=', $comment['author_id'])->first();
            array_push($full_comments, ['id' => $comment['id'], 'body' =>
$comment['body'], 'created_at' => $comment['created_at'], 'first_name' =>
$author['first_name'], 'last_name' => $author['last_name']]);
        }
    }

    return $this->view('masters.index', ['master' => $master, 'comments'
=> $full_comments]);
}
}

```

routes/web.php – список веб-маршрутів додатку.

<?php

```

use App\Http\Controllers\AreasActivityController;
use App\Http\Controllers\CalendarController;
use App\Http\Controllers\InfosController;
use App\Http\Controllers\MastersAppointmentsController;
use App\Http\Controllers\MastersController;
use App\Http\Controllers\MediaController;
use App\Http\Controllers\StudiesAppointmentsController;
use App\Http\Controllers\StudiesController;
use App\Http\Controllers\UserController;
use Illuminate\Support\Facades\Route;
use App\Http\Controllers\RegistrationController;
use App\Http\Controllers\SessionsController;
use App\Http\Controllers>PasswordController;
use App\Http\Controllers\ExtrasController;
use App\Http\Controllers>MainController;
use App\Http\Controllers\MastersCommentsController;
use App\Http\Controllers\ChiefAppointmentsController;

Route::get('/', [MainController::class, 'home']);
Route::get('/admin', [MainController::class, 'admin']);
Route::get('/admin/masters', [MainController::class, 'masters']);
Route::get("/admin/activities", [MainController::class, 'activities']);
Route::get("/admin/infos", [MainController::class, 'infos']);
Route::get("/admin/studies-topics", [MainController::class, 'studiesTopics']);
Route::get("/admin/extras-topics", [MainController::class, 'extrasTopics']);
Route::get("/admin/users", [MainController::class, 'users']);
Route::get("/admin/extras", [MainController::class, 'extras']);
Route::get("/admin/infos-posts", [MainController::class, 'infosPosts']);
Route::get("/admin/studies", [MainController::class, 'studies']);
Route::get("/admin/main-settings", [MainController::class, 'mainPageSettings']);
Route::get("/admin/comments", [MainController::class, 'comments']);

Route::post("/admin/save-main", [MainController::class, 'saveMain']);
Route::get("/admin/comments", [MainController::class, 'comments']);
Route::get("/admin/get-comments", [MainController::class, 'getComments']);
Route::post("/admin/master-comment-delete", [MastersCommentsController::class, 'delete']);

Route::get("/admin/calendar", [MainController::class, 'calendar']);

Route::get('/admin/get-infos-posts', [MainController::class, 'getInfosPosts']);
Route::get('/admin/get-extras', [MainController::class, 'getExtras']);
Route::get('/admin/get-studies', [MainController::class, 'getStudies']);

Route::get("/admin/get-users", [UserController::class, 'asyncUsers']);
Route::get("/admin/change-admin-users", [UserController::class, 'changeUsers']);

Route::post('/file-save', [MediaController::class, 'save']);

Route::get('/registration', [RegistrationController::class, 'create']);
Route::post('/registration', [RegistrationController::class, 'store']);

```

| | | | | | | |
|------|------|----------|--------|------|-------------------------|------|
| | | | | | КН.КР.Б – 122 – 24 – ПЗ | Арк. |
| | | | | | | 79 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

```

Route::get('/edit-profile', [UserController::class, 'editProfile']);
Route::post('/edit-profile', [UserController::class, 'saveProfile']);
Route::get('/users-admins', [UserController::class, 'usersForAdmin']);

Route::get('/login', [SessionsController::class, 'create']);
Route::post('/login', [SessionsController::class, 'store']);
Route::get('/logout', [SessionsController::class, 'destroy']);

Route::get('/forget-password', [PasswordController::class,
'showForgetPasswordForm']);
Route::post('/forget-password', [PasswordController::class,
'submitForgetPasswordForm']);
Route::get('/reset-password/{token}', [PasswordController::class,
'showResetPasswordForm']);
Route::post('/reset-password', [PasswordController::class,
'submitResetPasswordForm']);

Route::get("/admin/create-activity", [AreasActivityController::class, 'create']);
Route::get("/admin/edit-activity/{id}", [AreasActivityController::class,
'edit']);
Route::post("/admin/save-activity", [AreasActivityController::class, 'save']);
Route::post("/admin/activity-change-visible", [AreasActivityController::class,
'changeVisible']);
Route::post("/admin/activity-delete", [AreasActivityController::class,
'delete']);
Route::get("/area-activity/{id}", [AreasActivityController::class, 'topic']);

Route::get("/masters", [MastersController::class, 'masters']);
Route::get("/master/{id}", [MastersController::class, 'master']);
Route::post("/get-master-comments", [MastersCommentsController::class,
'getForMaster']);
Route::post("/create-master-comments", [MastersCommentsController::class,
'create']);
Route::get("/admin/create-master", [MastersController::class, 'create']);
Route::get("/admin/edit-master/{id}", [MastersController::class, 'edit']);
Route::post("/admin/save-master", [MastersController::class, 'save']);
Route::post("/admin/master-change-visible", [MastersController::class,
'changeVisible']);
Route::post("/admin/master-delete", [MastersController::class, 'delete']);

Route::get("/admin/create-study-topic", [StudiesController::class,
'createTopic']);
Route::get("/admin/edit-study-topic/{id}", [StudiesController::class,
'editTopic']);
Route::post("/admin/save-study-topic", [StudiesController::class, 'saveTopic']);
Route::post("/admin/study-topic-change-visible", [StudiesController::class,
'changeVisibleTopic']);
Route::post("/admin/study-topic-delete", [StudiesController::class,
'deleteTopic']);

Route::get("/admin/create-study", [StudiesController::class, 'createStudy']);
Route::get("/admin/edit-study/{id}", [StudiesController::class, 'editStudy']);
Route::post("/admin/edit-study", [StudiesController::class, 'saveStudy']);
Route::post("/admin/study-change-visible", [StudiesController::class,
'changeVisibleStudy']);
Route::post("/admin/study-delete", [StudiesController::class, 'deleteStudy']);

```

| | | | | | | |
|------|------|----------|--------|------|-------------------------|------|
| | | | | | КН.КР.Б – 122 – 24 – ПЗ | Арк. |
| | | | | | | 80 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

```

Route::get('/get-studies', [StudiesController::class, 'getStudies']);
Route::get("/studies/{id}", [StudiesController::class, 'studies']);

Route::get("/calendar-times/{date}", [CalendarController::class, 'getTimes']);
Route::get("/admin/calendar-times/{date}", [MainController::class, 'getTimes']);
Route::post("/admin/calendar-time-edit", [CalendarController::class,
'saveTimeCalendar']);
Route::post("/admin/calendar-time-delete", [CalendarController::class,
'delete']);

Route::get("/admin/create-info", [InfosController::class, 'createInfo']);
Route::get("/admin/edit-info/{id}", [InfosController::class, 'editInfo']);
Route::post("/admin/save-info", [InfosController::class, 'saveInfo']);
Route::post("/admin/info-change-visible", [InfosController::class,
'changeVisible']);
Route::post("/admin/info-delete", [InfosController::class, 'delete']);

Route::get("/admin/create-info-post", [InfosController::class, 'createPost']);
Route::get("/admin/edit-info-post/{id}", [InfosController::class, 'editPost']);
Route::post("/admin/save-info-post", [InfosController::class, 'savePost']);
Route::post("/admin/info-post-change-visible", [InfosController::class,
'changeVisiblePost']);
Route::post("/admin/info-post-delete", [InfosController::class, 'deletePost']);
Route::get('/get-infos-posts', [InfosController::class, 'getPosts']);
Route::get('/infos-posts/{id}', [InfosController::class, 'infosPosts']);

Route::get('/admin/create-topic-extra', [ExtrasController::class,
'createTopic']);
Route::get('/admin/edit-topic-extra/{id}', [ExtrasController::class,
'editTopic']);
Route::post('/admin/save-topic-extra', [ExtrasController::class, 'saveTopic']);
Route::post("/admin/topic-extra-change-visible", [ExtrasController::class,
'changeVisibleTopic']);
Route::post("/admin/topic-extra-delete", [ExtrasController::class,
'deleteTopic']);

Route::get('/admin/create-extra', [ExtrasController::class, 'createPost']);
Route::get('/admin/edit-extra/{id}', [ExtrasController::class, 'editPost']);
Route::post('/admin/save-extra', [ExtrasController::class, 'savePost']);
Route::post('/admin/extra-post-delete', [ExtrasController::class, 'deletePost']);
Route::post('/admin/extra-post-change-visible', [ExtrasController::class,
'changeVisiblePost']);
Route::get('/extra-posts/{id}', [ExtrasController::class, 'extras']);
Route::get('/get-extras', [ExtrasController::class, 'getPosts']);

Route::post('/note-to-boss', [ChiefAppointmentsController::class, 'save']);
Route::post('/admin/note-to-boss-reject', [ChiefAppointmentsController::class,
'reject']);
Route::post('/admin/note-to-boss-accept', [ChiefAppointmentsController::class,
'accept']);

Route::post('/note-to-master', [MastersAppointmentsController::class, 'save']);
Route::get("/admin/notes-to-masters", [MastersAppointmentsController::class,
'allNotes']);
Route::post("/admin/get-notes-to-masters", [MastersAppointmentsController::class,
'getNotes']);
Route::post('/admin/note-to-master-reject',

```

| | | | | | | |
|------|------|----------|--------|------|-------------------------|------|
| | | | | | КН.КР.Б – 122 – 24 – ПЗ | Арк. |
| | | | | | | 81 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

```

[MastersAppointmentsController::class, 'reject']);
Route::post('/admin/note-to-master-accept',
[MastersAppointmentsController::class, 'accept']);

Route::post('/note-to-study', [StudiesAppointmentsController::class, 'save']);
Route::get("/admin/notes-to-studies", [StudiesAppointmentsController::class,
'allNotes']);
Route::post("/admin/get-notes-to-studies", [StudiesAppointmentsController::class,
'getNotes']);
Route::post('/admin/note-to-study-reject', [StudiesAppointmentsController::class,
'reject']);
Route::post('/admin/note-to-study-accept', [StudiesAppointmentsController::class,
'accept']);

Route::get("/user/notes-to-studies", [StudiesAppointmentsController::class,
'userNotes']);
Route::get("/user/notes-to-masters", [MastersAppointmentsController::class,
'userNotes']);
Route::get("/user/notes-to-chief", [ChiefAppointmentsController::class,
'userNotes']);

```

Додаток В

SessionController.php – контролер, відповідальний за авторизацію.

```

<?php

namespace App\Http\Controllers;

use Illuminate\Support\Facades\Request;

class SessionsController extends Controller
{
    public function create()
    {
        if (auth()->check()) return abort(404);
        return $this->view('auth.login');
    }

    public function store()
    {
        if (auth()->check()) return abort(404);
        if (auth()->attempt(request(['email', 'password'])) == false) {
            return back()->withInput(Request::except('password'))->withErrors([
                'message' => 'Електронна пошта чи пароль неправильні.'
            ]->with('login', true));
        }
        return redirect()->to('/');
    }

    public function destroy()
    {
        if (!auth()->check()) return redirect()->to('/');
        auth()->logout();

        return redirect()->to('/');
    }
}

```

| | | | | | | |
|------|------|----------|--------|------|-------------------------|------|
| | | | | | КН.КР.Б – 122 – 24 – ПЗ | Арк. |
| | | | | | | 82 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

```
}
}
```

Додаток Г

MediaController.php – контролер, відповідальний за завантаження медіа

В СХОВИЩЕ.

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\Storage;

class MediaController
{
    public function save(Request $request)
    {
        try {
            $file = $request->file("file");
            if (is_null($file)) return json_encode(["filename" => 'null']);
            $filename = time() . "_" . $file->getClientOriginalName();
            Storage::putFileAs('public/' . $request->input('type') . 's/', $file,
$filename);
            return json_encode(["filename" => $filename]);
        } catch (\Exception $e) {
            return json_encode(["filename" => 'null']);
        }
    }
}
```

Додаток Д

User.php – модель User

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;
use Laravel\Sanctum\HasApiTokens;

class User extends Authenticatable
{
    use HasApiTokens, HasFactory, Notifiable;
    protected $fillable = [
        'name',
        'email',
        'password',
        'first_name',
        'last_name',
    ];
}
```

| | | | | | | |
|------|------|----------|--------|------|-------------------------|------|
| | | | | | КН.КР.Б – 122 – 24 – ПЗ | Арк. |
| | | | | | | 83 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

```

        'phone',
        'social_type',
        'social_phone',
        'admin'
    ];
    protected $hidden = [
        'password',
        'remember_token'
    ];
    protected $casts = [
        'email_verified_at' => 'datetime',
    ];
    public function setPasswordAttribute($password)
    {
        $this->attributes['password'] = bcrypt($password);
    }
}

```

Додаток Е

.env – конфігураційний файл.

```

APP_NAME=Laravel
APP_ENV=local
APP_KEY=base64:LZ+FY207Vk0Bb8vAg4p9gdGcL/zWo62f4x0ES7r1J0A=
APP_DEBUG=true
APP_URL=http://localhost

```

```

LOG_CHANNEL=stack
LOG_DEPRECATIONS_CHANNEL=null
LOG_LEVEL=debug

```

```

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=laravel
DB_USERNAME=root
DB_PASSWORD=

```

```

BROADCAST_DRIVER=log
CACHE_DRIVER=file
FILESYSTEM_DISK=local
QUEUE_CONNECTION=sync
SESSION_DRIVER=file
SESSION_LIFETIME=120

```

```

MEMCACHED_HOST=127.0.0.1

```

```

REDIS_HOST=127.0.0.1
REDIS_PASSWORD=null
REDIS_PORT=6379

```

```

MAIL_DRIVER=sendmail
MAIL_HOST=smtp.gmail.com
MAIL_PORT=587
MAIL_USERNAME=ollbutchersmail@gmail.com

```

| | | | | | | |
|------|------|----------|--------|------|-------------------------|------|
| | | | | | КН.КР.Б – 122 – 24 – ПЗ | Арк. |
| | | | | | | 84 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

MAIL_PASSWORD=enyuvoqevdqnxkx
MAIL_FROM_ADDRESS=ollbutchersmail@gmail.com
MAIL_FROM_NAME=Vadym
MAIL_ENCRYPTION=tls

AWS_ACCESS_KEY_ID=
AWS_SECRET_ACCESS_KEY=
AWS_DEFAULT_REGION=us-east-1
AWS_BUCKET=
AWS_USE_PATH_STYLE_ENDPOINT=false

PUSHER_APP_ID=
PUSHER_APP_KEY=
PUSHER_APP_SECRET=
PUSHER_HOST=
PUSHER_PORT=443
PUSHER_SCHEME=https
PUSHER_APP_CLUSTER=mt1

VITE_PUSHER_APP_KEY="\${PUSHER_APP_KEY}"
VITE_PUSHER_HOST="\${PUSHER_HOST}"
VITE_PUSHER_PORT="\${PUSHER_PORT}"
VITE_PUSHER_SCHEME="\${PUSHER_SCHEME}"
VITE_PUSHER_APP_CLUSTER="\${PUSHER_APP_CLUSTER}"

Додаток Є

UserTest.php – модуль Unit-тестування моделі User

```
namespace Tests\Unit;

use Tests\TestCase;
use Illuminate\Foundation\Testing\RefreshDatabase;
use App\Models\User;

class UserTest extends TestCase
{
    use RefreshDatabase;

    /** @test */
    public function it_can_create_a_user()
    {
        $user = User::create([
            'name' => 'Test User',
            'email' => 'test@example.com',
            'password' => 'password', // The password will be automatically
hashed by the model
            'first_name' => 'Test',
            'last_name' => 'User',
            'phone' => '123456789',
            'social_type' => 'facebook',
            'social_phone' => '987654321',
            'admin' => false,
        ]);

        $this->assertDatabaseHas('users', ['email' => 'test@example.com']);
    }
}
```

| | | | | | | |
|------|------|----------|--------|------|-------------------------|------|
| | | | | | КН.КР.Б – 122 – 24 – ПЗ | Арк. |
| | | | | | | 85 |
| Змн. | Арк. | № докум. | Підпис | Дата | | |

```

        $this->assertEquals('Test User', $user->name);
    }

    /** @test */
    public function it_can_update_a_user()
    {
        $user = User::factory()->create();
        $user->name = 'Updated Name';
        $user->save();

        $this->assertDatabaseHas('users', ['name' => 'Updated Name']);
    }

    /** @test */
    public function it_can_delete_a_user()
    {
        $user = User::factory()->create();
        $user->delete();

        $this->assertDeleted($user);
    }

    /** @test */
    public function it_can_find_a_user_by_email()
    {
        $user = User::factory()->create([
            'email' => 'findme@example.com',
        ]);

        $foundUser = User::where('email', 'findme@example.com')->first();
        $this->assertEquals($user->id, $foundUser->id);
    }

    /** @test */
    public function it_can_check_user_password()
    {
        $user = User::factory()->create([
            'password' => 'password', // The password will be automatically
hashed by the model
        ]);

        $this->assertTrue(\Hash::check('password', $user->password));
    }
}

```