Київський столичний університет імені Бориса Грінченка
Факультет інформаційних технологій та математики
Кафедра комп'ютерних наук

Допущено до захисту
Завідувач кафедри
комп'ютерних наук, кандидат
технічних наук, доцент,
_____ Ірина МАШКІНА

(підпис)

« ___ » _____ 2025 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня «бакалавр»

Спеціальність 122 Комп'ютерні науки

Освітня програма 122.00.01 Інформатика

Тема:

СТВОРЕННЯ ІНСТРУМЕНТУ ДЛЯ УПРАВЛІННЯ ВЗАЄМОВІДНОСИНАМИ З КЛІЄНТАМИ В ОНЛАЙН-КУРСАХ

Виконав
студент групи ІНб-2-21-4.0д

Акопян Максим Манвелович

_____

Науковий керівник
кандидат технічних наук, доцент

Мельник Ірина Юріївна

_____

Київ – 2025

Київський столичний університет імені Бориса Грінченка

Факультет інформаційних технологій та математики
Кафедра комп'ютерних наук

«Затверджую»
Завідувач кафедри
комп'ютерних наук, кандидат
технічних наук, доцент,
_____ Ірина МАШКІНА

(підпис)

« ___ » _____ 2025 р.

ЗАВДАННЯ НА КВАЛІФІКАЦІНУ РОБОТУ БАКАЛАВРА

«Створення інструменту для управління взаємовідносинами з клієнтами в онлайн-курсах»

Виконавець – студент групи ІНб-1-хх-4.0д, спеціальності 122 Комп'ютерні наук, освітньої програми 122.00.01 Інформатика **Акопян Максим Манвелович**

1. Вихідні дані: *інструмент для управління взаємовідносинами з клієнтами в онлайн-курсах.*

2. Основні завдання: *Провести аналіз сучасних CRM-систем для онлайн-курсів. Здійснити огляд вебтехнологій для створення CRM-систем. Обґрунтувати вибір засобів і методів для розробки CRM. Спроєктувати архітектуру інструменту. Реалізувати серверну частину CRM-системи на основі Laravel. Створити адміністративну панель за допомогою Filament. Організувати базу даних MySQL для зберігання даних про клієнтів, курси, викладачів і оплати.*

3. Пояснювальна записка: *Обсяг – до 60 стор. формату А4 комп'ютерного набору з дотриманням вимог стандарту і методичних рекомендацій кафедри.*

4. Графічні матеріали: *не передбачено.*

5. Додатки: *не передбачено*.

6. Строк подання роботи на кафедру: *«___» _____2025 р.*

Науковий керівник Виконавець: к.т.н., доцент

_____ Мельник І.Ю. _____ Акопян М.М.

дата

дата

Календарний план

№	Назва етапу дипломної роботи	Строк виконання етапу роботи	Примітка
1	Формування теми дипломної роботи бакалавра	10.10.2024	Виконано
2	Ознайомлення з методичними рекомендаціями до дипломної роботи	15.10.2024	Виконано
3	Складання змісту та календарного плану роботи	15.10.2024	Виконано
4	Підбір літератури, складання завдання	18.10.2024	Виконано
5	Написання реферату та вступу	19.11.2024	Виконано
6	Написання першого розділу	20.12.2024	Виконано
7	Написання другого розділу	25.02.2025	Виконано
8	Написання третього розділу	29.03.2025	Виконано
9	Написання висновків	29.03.2025	Виконано
10	Виправлення зауважень	5.04.2025	Виконано
11	Створення презентації	22.04.2025	Виконано
12	Захист матеріалів дипломної роботи на засіданні кафедри	15.05.2025	Виконано
13	Офіційний захист матеріалів дипломної роботи на засіданні екзаменаційної комісії	18.06.2025	Виконано

Здобувач_____

Керівник роботи _____

РЕФЕРАТ

Кваліфікаційна робота: с. 60, рис. 4, табл. 1, 27 посилань.

Актуальність:

У сучасних умовах стрімкого розвитку онлайн-освіти, організація ефективного управління взаємовідносинами з клієнтами, зокрема студентами онлайн-курсів, є важливим чинником успішної діяльності освітніх платформ. Більшість існуючих CRM-систем мають універсальний характер і не враховують специфіку взаємодії у сфері онлайн-навчання, що обумовлює актуальність створення спеціалізованих інструментів для управління навчальним процесом, контролю оплати, реєстрації та супроводу клієнтів.

Об’єкт дослідження:

Процес управління взаємовідносинами з клієнтами у сфері онлайн-курсів.

Предмет дослідження:

Методи, технології та архітектурні рішення створення вебінструменту для управління клієнтами онлайн-курсів на основі сучасних вебтехнологій.

Мета роботи:

Створення спеціалізованого інструменту для управління взаємовідносинами з клієнтами в онлайн-курсах з використанням сучасних технологій Laravel, Filament та MySQL.

Завдання:

- провести аналіз сучасних CRM-систем для онлайн-курсів;
- здійснити огляд вебтехнологій для створення CRM-систем;
- обґрунтувати вибір засобів і методів для розробки CRM;
- спроєктувати архітектуру інструменту;
- реалізувати серверну частину CRM-системи на основі Laravel;
- створити адміністративну панель за допомогою Filament;

- організувати базу даних MySQL для зберігання даних про клієнтів, курси, викладачів і оплати.

Основні результати:

У ході роботи було розроблено спеціалізований модуль до CRM-системи для онлайн-курсів, що включає серверну частину на базі Laravel, адміністративну панель на основі Filament та інтегровану базу даних MySQL. Отримано новий програмний продукт, адаптований до потреб управління освітніми проєктами в онлайн-форматі, з можливістю реєстрації курсів, ведення клієнтської бази, контролю оплати, відстеження навчального прогресу та формування звітності. Новизна роботи полягає у створенні гнучкого, масштабованого та адаптивного рішення, спеціалізованого для онлайн-курсів, з урахуванням особливостей їх організації та взаємодії з клієнтами.

Практичне значення дослідження:

Полягатиме у впровадженні розробленого модуля до CRM-систем онлайн-курсів для автоматизації роботи з клієнтами, оптимізації управління навчальними проєктами, підвищення якості обслуговування студентів та полегшення адміністративних процесів у сфері онлайн-освіти.

Ключові слова:

CRM-СИСТЕМА, ОНЛАЙН-КУРСИ, ВЗАЄМОВІДНОСИНИ З КЛІЄНТАМИ, LARAVEL, FILAMENT, MYSQL, АДМІНІСТРУВАННЯ, УПРАВЛІННЯ КЛІЄНТАМИ, ОСВІТНІЙ ПРОЦЕС, ОНЛАЙН-ОСВІТА.

ЗМІСТ

ВСТУП

У сучасному світі управління взаємовідносинами з клієнтами є важливою складовою успіху бізнесу, особливо в таких сферах, як онлайн-освіта. Онлайн-курси, як один з головних інструментів сучасного навчання, вимагають ефективних механізмів для управління контактами з учасниками, що забезпечує не тільки підтримку навчального процесу, але й стимулює зростання платформи через належну взаємодію з користувачами. Сучасні CRM-системи (Customer Relationship Management) відіграють ключову роль у цій взаємодії, дозволяючи автоматизувати процеси управління студентами, курсами та комунікацією.

Аналіз вітчизняної та зарубіжної науково-технічної літератури з проблеми створення CRM-систем для онлайн-курсів показує наявність широкого спектру інструментів та рішень. Зарубіжні компанії, такі як Salesforce, HubSpot, Zoho, вже давно розробили та інтегрували CRM-системи для різних напрямків бізнесу, включаючи освітні платформи. Вони пропонують розв’язання основних задач, таких як автоматизація маркетингу, управління взаємодією з клієнтами, відстеження їхнього прогресу та забезпечення персоналізованої підтримки. У той же час на вітчизняному ринку є потреба у розробці спеціалізованих CRM-систем для онлайн-курсів, що забезпечить врахування специфічних вимог до інтеграції з навчальними платформами та обробки даних студентів [1].

Проте в наукових дослідженнях можна помітити прогалини щодо розробки та впровадження інструментів, які дозволяють ефективно інтегрувати CRM-системи із навчальними платформами, зокрема в умовах вітчизняних освітніх установ. Існуючі рішення зазвичай не враховують усі нюанси навчальних процесів та специфіки онлайн-освіти, що створює потребу в розробці нових, адаптованих інструментів для управління взаємовідносинами з клієнтами в цій галузі.

Актуальність цієї роботи обумовлена необхідністю створення інструменту для управління взаємовідносинами з клієнтами в онлайн-курсах, який забезпечить комплексний підхід до обробки даних студентів, управління курсами та автоматизації комунікації між учасниками навчального процесу. Крім того, розробка такого інструменту дозволить значно підвищити ефективність роботи онлайн-платформ та покращити якість обслуговування студентів.

Метою даної роботи є розробка CRM-системи для управління взаємовідносинами з клієнтами в онлайн-курсах, яка дозволить автоматизувати основні процеси взаємодії між студентами та платформою.

Для досягнення поставленої мети необхідно вирішити такі завдання:

1. Оцінити сучасний стан CRM-систем для онлайн-курсів, проаналізувати їхні переваги та недоліки.
2. Обґрунтувати вибір технологій для розробки CRM-системи, таких як Laravel, Filament, MySQL.
3. Розробити архітектуру CRM-системи для онлайн-курсів.
4. Реалізувати бекенд-частину системи на основі технології Laravel.
5. Розробити адміністративну панель для керування CRM-системою з використанням Filament.
6. Інтегрувати базу даних MySQL для зберігання інформації про студентів та курси.

Об’єктом дослідження є процеси управління взаємовідносинами з клієнтами в онлайн-курсах, а предметом — інструменти та методи, які застосовуються для автоматизації та оптимізації цих процесів [2].

Методи дослідження включають аналіз наукової та технічної літератури, дослідження існуючих CRM-рішень, проектування архітектури програмного забезпечення, використання методів об’єктно-орієнтованого програмування та розробки веб-додатків, а також тестування розробленого програмного продукту.

Практичне значення одержаних результатів полягає в можливості використання розробленої CRM-системи для управління онлайн-курсами, що дозволить значно покращити ефективність взаємодії з клієнтами та автоматизувати основні процеси на освітніх платформах. Результати дослідження можуть бути використані в проектах створення та впровадження CRM-систем для освітніх організацій та онлайн-платформ [3].

Галузь застосування результатів роботи охоплює розробку програмних засобів для онлайн-освіти, автоматизацію взаємовідносин з клієнтами та адміністрування навчальних платформ. Апробація результатів роботи здійснюватиметься через тестування розробленої системи в реальних умовах онлайн-курсів та публікацію матеріалів на наукових семінарах та конференціях.

РОЗДІЛ І
ТЕОРЕТИЧНІ АСПЕКТИ УПРАВЛІННЯ ВЗАЄМОВІДНОСИНАМИ З КЛІЄНТАМИ В ОНЛАЙН-КУРСАХ

1.1 Аналіз сучасних інструментів CRM для онлайн-курсів

У сучасному світі онлайн-освіти ефективне управління взаємовідносинами з клієнтами є важливою складовою успіху будь-якої освітньої платформи. CRM-системи (Customer Relationship Management) дозволяють не лише зберігати та організовувати інформацію про клієнтів, а й автоматизувати процеси взаємодії з ними, покращуючи якість обслуговування та збільшуючи лояльність. Для онлайн-курсів важливою функцією CRM є ефективне управління студентами, курсами, оплатами та зворотним зв'язком [4].

Сьогодні на ринку існує багато CRM-систем, спеціалізованих на управлінні взаємовідносинами з клієнтами для освітніх платформ. Одним з найбільш відомих і універсальних рішень є HubSpot CRM (рис. 1.1).

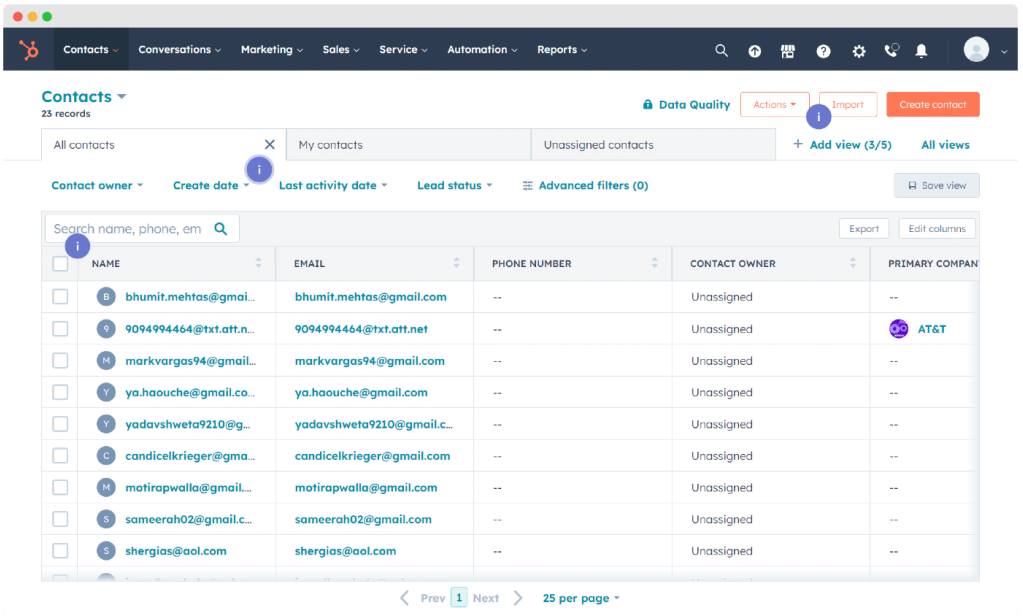

Рисунок 1.1 - HubSpot CRM

Ця система дозволяє зберігати всі дані про клієнтів, а також взаємодіяти з ними через різні канали комунікації, включаючи електронну

пошту, соціальні мережі та інші канали. Вона пропонує потужні інструменти для маркетингу, аналізу та відстеження активності студентів. Для онлайн-курсів це може бути корисно для автоматизації процесів реєстрації студентів, управління курсами, контролю за оплатами та надання персоналізованого зворотного зв'язку [5].

Іншим популярним інструментом є Salesforce, який часто використовують великі освітні організації та платформи. Salesforce надає можливості для глибокої персоналізації взаємодії з клієнтами, дозволяючи створювати кастомізовані ланцюжки для кожного студента та надаючи потужні аналітичні інструменти (рис. 1.2). Ця система особливо корисна для платформ, які працюють з великою кількістю курсів та користувачів, оскільки вона дозволяє ефективно управляти взаємодією з клієнтами та автоматизувати численні процеси.

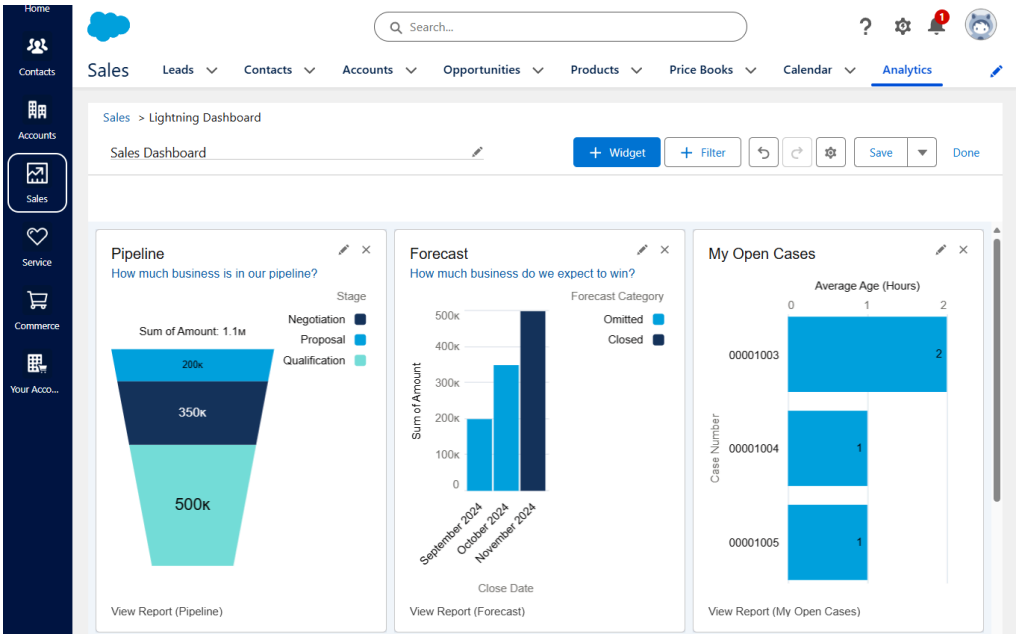

Рисунок 1.2 - Salesforce

Система Zoho CRM є ще однією популярною платформою, яка використовується для управління клієнтами в онлайн-курсах. Вона дозволяє створювати автоматизовані процеси для реєстрації студентів, організації онлайн-навчання, обробки запитів студентів та управління їх оплатами [6].

Zoho також підтримує інтеграцію з іншими сервісами для електронного навчання та управління контентом, що є великою перевагою для онлайн-освітніх платформ (рис. 1.3).

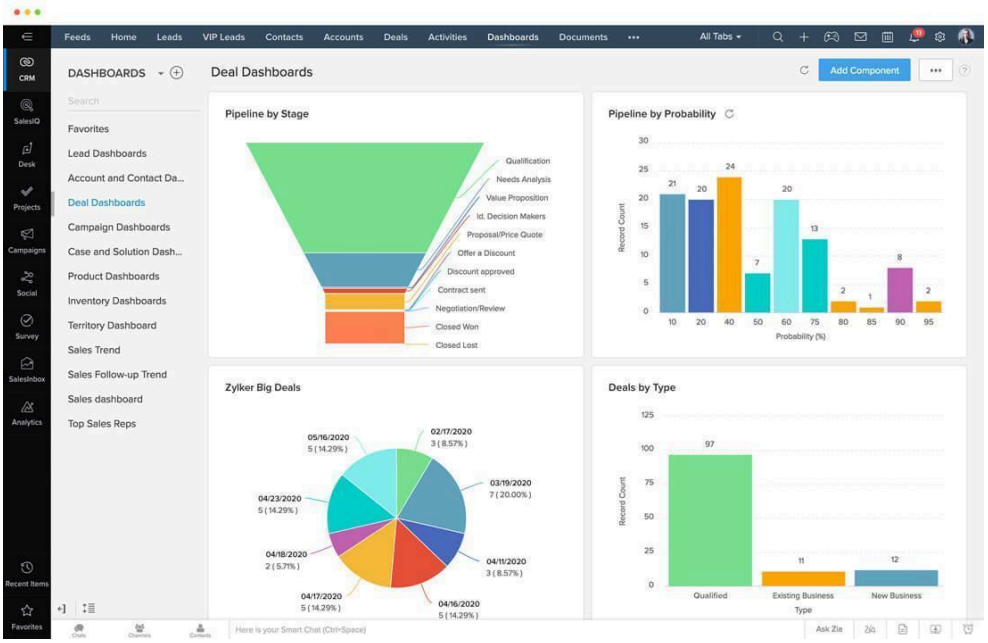

Рисунок 1.3 - Zoho CRM

Окрім таких великих і відомих CRM-систем, існують і спеціалізовані рішення, призначені саме для онлайн-освіти, такі як Teachery, Thinkific, Teachable. Ці платформи дозволяють зосередитися саме на управлінні курсами та студентами, зокрема вони надають функціонал для створення курсів, реєстрації студентів, моніторингу їхнього прогресу та управління платежами. Вони мають вбудовані CRM-функції для ефективного відстеження клієнтської активності, збору даних про студентів та полегшення процесу навчання [7].

Також після аналізу можемо створити порівняльну таблицю цих трьох CRM (табл. 1.1).

Характеристика	**Salesforce**	**Zoho CRM**	**HubSpot CRM**
Цільова аудиторія	Великі та середні компанії, корпоративний сектор	Малий, середній та великий бізнес	Малий та середній бізнес, стартапи

Основні функції	Повний спектр CRM-функцій, автоматизація продажів, аналітика, маркетинг, підтримка клієнтів, кастомізація	Управління лідами, продажами, маркетингом, аналітика, інтеграції	Управління контактами, маркетинг, email-розсилки, звіти, чат, інтеграції
Вартість	Висока, від $25 до $300+ за користувача/місяць	Доступна, від $14 до $52 за користувача/місяць	Безкоштовний базовий тариф, платні — від $20
Інтерфейс	Потужний, складний для новачків, гнучкий	Зручний, простий у використанні	Інтуїтивно зрозумілий, простий
Можливість інтеграції	Багато інтеграцій з зовнішніми сервісами та API	Широкий вибір інтеграцій	Легкі інтеграції з популярними сервісами (Gmail, Outlook, Slack, Zoom)
Автоматизація бізнес-процесів	Розвинена система автоматизації та сценаріїв	Вбудовані шаблони автоматизації	Базова автоматизація у безкоштовній версії, розширена — у платній
Аналітика та звітність	Потужна аналітика, гнучкі звіти, AI-аналітика	Стандартна та розширена аналітика	Базова аналітика, розширена — у платних тарифах
Мобільний додаток	Так	Так	Так
Підходить для онлайн-курсів	Так, для великих освітніх проєктів з гнучкою кастомізацією	Так, для малого та середнього EdTech бізнесу	Так, для невеликих освітніх платформ та стартапів

Таблиця 1.1 – Порівнвняня CRM

Загалом, вибір CRM-системи для онлайн-курсів залежить від кількості курсів, студентів, необхідності в інтеграціях з іншими інструментами та вимог до персоналізації. Важливо обрати платформу, яка не лише автоматизує процеси, але й дозволяє забезпечити високий рівень підтримки студентів та оптимізувати роботу з даними [8].

В освітній сфері клієнтами є не лише ті, хто купує чи проходить навчання, але й потенційні слухачі, корпоративні партнери, викладачі та представники інших організацій. Саме тому ефективне управління взаємовідносинами з клієнтами стає ключовим фактором успішного функціонування та розвитку онлайн-курсів.

Управління взаємовідносинами з клієнтами (CRM — Customer Relationship Management) у сфері онлайн-освіти передбачає систематичне ведення обліку клієнтів, контроль за взаємодією з ними, організацію комунікацій, персоналізацію навчального процесу, а також аналіз ефективності взаємодії. Це дозволяє не лише залучати нових користувачів, а й утримувати постійних клієнтів, підвищувати їхню лояльність і задоволення від освітніх послуг.

Основними перевагами впровадження CRM-систем в освітній сфері є:

- Підвищення якості обслуговування клієнтів. Завдяки централізованому обліку контактів, історії взаємодій та персональних даних кожного слухача стає можливим забезпечення оперативного та індивідуального підходу до вирішення питань клієнтів.
- Побудова довгострокових взаємин. CRM-система дозволяє фіксувати історію замовлень, проходження курсів, зворотний зв’язок та інші дані, що сприяє формуванню довірчих відносин і повторному залученню клієнтів.
- Аналіз і прогнозування потреб клієнтів. Зібрані дані дають змогу аналізувати поведінку користувачів, їх уподобання та інтереси, що

дозволяє пропонувати персоналізований контент, додаткові курси або акційні пропозиції.

- Автоматизація бізнес-процесів. За допомогою CRM можна автоматизувати реєстрацію на курси, надсилання повідомлень, нагадувань, облік оплати та формування звітності.
- Оптимізація роботи персоналу. Використання єдиної системи обліку зменшує ризик втрати інформації, дублювання даних та спрощує доступ до необхідної інформації для менеджерів і викладачів.

Особливо актуальним є впровадження CRM-систем для онлайн-курсів, оскільки освітній процес у дистанційному форматі потребує більшої уваги до організації комунікацій та контролю за взаємодією з клієнтами. Без ефективного інструменту для управління взаємовідносинами складно забезпечити належний рівень сервісу, оперативно реагувати на запити користувачів та підтримувати їхню залученість у процес навчання.

Основні функції CRM-системи для онлайн-курсів охоплюють кілька ключових аспектів, що забезпечують ефективне управління взаємодією з клієнтами. Однією з основних функцій є збір та аналіз даних про учасників. CRM-система виступає як єдине сховище для всієї інформації про клієнтів: від персональних даних (ім’я, електронна пошта, номер телефону) до історії взаємодій з платформою. Завдяки цьому можна відслідковувати, які курси проходять слухачі, їхні результати, а також будь-які запити або проблеми, які можуть виникати. Такий підхід дозволяє створити детальну картину кожного користувача і адаптувати сервіс до їхніх індивідуальних потреб.

Іншою важливою функцією є автоматизація маркетингових кампаній та комунікацій. CRM-система дає змогу автоматично надсилати користувачам персоналізовані повідомлення, нагадування про курс чи нові пропозиції, оновлення щодо розкладу та акцій. Це дозволяє ефективно залучати нових учасників, підтримувати їхній інтерес до платформ, а також стимулювати повторні покупки або участь у нових курсах. Завдяки сегментації клієнтської

бази та аналізу попередніх взаємодій, можна налаштувати ці кампанії так, щоб вони були максимально релевантними та ефективними.

Крім того, CRM-система значно покращує підтримку клієнтів та сервісну підтримку. Всі запити, проблеми або питання користувачів зберігаються в єдиній базі даних, що дозволяє менеджерам швидко знаходити інформацію і оперативно вирішувати проблеми. Це також забезпечує високий рівень персоналізації обслуговування, оскільки кожен клієнт має історію взаємодій, що допомагає менеджерам реагувати на запити вчасно та точно. Важливою складовою є також моніторинг задоволення клієнтів, що дозволяє виявляти слабкі місця в обслуговуванні та покращувати якість надання послуг.

Онлайн-курси мають свою специфіку роботи з аудиторією — це постійна комунікація з клієнтами, автоматизація розсилок, облік записів на курси, управління сертифікатами, збереження історії взаємодій та аналітика активності слухачів. Тому при виборі CRM важливо враховувати не лише загальний функціонал, а й можливість його адаптації до потреб навчального сервісу.

Насамперед, одним із головних критеріїв є функціональність. CRM має забезпечувати зручний облік клієнтів, їх контактів, історії замовлень, записів на курси та результатів навчання. Важливо, щоб система дозволяла автоматизувати розсилки нагадувань, інформаційних листів, маркетингових пропозицій та анкет для зворотного зв’язку. Додатковою перевагою є наявність модуля для аналітики, який дозволяє оцінювати популярність курсів, активність користувачів і рівень задоволеності.

Другим визначальним критерієм є можливість інтеграції з іншими системами, які використовуються в рамках онлайн-курсу. Це можуть бути системи електронної комерції, email-маркетингу, платіжні сервіси або освітні платформи. Наявність відкритого API або готових інтеграцій значно спрощує об’єднання всіх сервісів в єдину екосистему.

Не менш важливим є зручність у користуванні та адмініструванні. CRM має мати простий і зрозумілий інтерфейс як для адміністраторів, так і для менеджерів, які працюють із клієнтами. Можливість швидко додавати нові поля, створювати фільтри для відбору клієнтів, налаштовувати типи розсилок — усе це має бути доступним без необхідності залучення програмістів.

Ще одним критерієм є вартість впровадження та обслуговування CRM. Оскільки проєкт створюється для освітньої сфери, де бюджети часто обмежені, важливо обирати рішення, яке не потребує значних фінансових витрат на ліцензування, додаткові модулі або підтримку. Саме тому перевагу варто надавати безкоштовним або умовно безкоштовним системам, або ж реалізовувати власне рішення на основі відкритих технологій.

Окрему увагу приділяється гнучкості налаштування та можливості розширення функціоналу. Онлайн-курси — це динамічна сфера, яка постійно розвивається. Тому CRM має дозволяти вносити зміни до структури бази даних, додавати нові типи полів, модифікувати шаблони повідомлень та автоматизованих дій без повної переробки системи.

Процес впровадження CRM-системи в освітній процес онлайн-курсів має бути комплексним, поетапним і добре продуманим, оскільки від цього залежить не лише ефективність взаємодії з клієнтами, а й загальний рівень сервісу, задоволеність слухачів та конкурентоспроможність навчального проєкту. Передусім варто розпочати з підготовчого етапу, на якому необхідно визначити основні цілі впровадження CRM: систематизація клієнтської бази, автоматизація комунікацій, поліпшення сервісу підтримки, аналітика ефективності курсів та маркетингових кампаній. На цьому ж етапі відбувається аналіз існуючих бізнес-процесів навчального проєкту, виявлення слабких місць і визначення тих елементів взаємодії з клієнтами, які потребують автоматизації.

Наступним кроком є безпосереднє налаштування та адаптація CRM до потреб онлайн-курсів. Йдеться про структурування бази даних, створення розділів для зберігання інформації про клієнтів, курси, записи, сертифікати, платежі та зворотній зв’язок. Важливо одразу впровадити автоматизовані процеси: надсилання листів-підтверджень після реєстрації, нагадувань про початок курсів, анкетування після завершення навчання. Доцільно також передбачити інтеграцію з платформами розсилок, платіжними системами та системами відеозанять, аби всі етапи взаємодії з клієнтом відбувалися в єдиному середовищі.

Стратегія оптимізації використання CRM передбачає регулярний аналіз її роботи та постійне вдосконалення процесів. Серед основних напрямів — налаштування аналітичних звітів для відстеження кількості нових клієнтів, активності слухачів, ефективності рекламних кампаній і рівня задоволеності навчанням. Ці дані дозволяють швидко реагувати на зміни в поведінці аудиторії, коригувати стратегію комунікацій та вдосконалювати навчальні програми. Для досягнення максимальної ефективності важливо навчити менеджерів і адміністраторів використовувати CRM на практиці — проводити навчальні сесії, створювати інструкції та впроваджувати стандарти ведення клієнтської бази.

Окрім того, однією з ефективних стратегій є впровадження персоналізованих сценаріїв комунікації на основі даних з CRM. Це можуть бути індивідуальні пропозиції повторного навчання, знижки для постійних клієнтів, нагадування про закінчення доступу до курсів, запрошення на безкоштовні вебінари чи тестування нових програм. Також важливо передбачити регулярне оновлення та актуалізацію бази даних, щоб уникнути дублювання записів, неактуальних контактів і застарілих даних.

1.2 Огляд вебтехнологій для створення інструменту управління взаємовідносинами з клієнтами

Для створення інструменту управління взаємовідносинами з клієнтами (CRM-системи) важливим етапом є вибір відповідних вебтехнологій, які дозволяють забезпечити ефективну взаємодію між користувачами та системою, а також зручне адміністрування даних, управління курсами та комунікацію з клієнтами. У процесі розробки CRM-системи для онлайн-курсів необхідно враховувати кілька основних аспектів веброзробки, таких як вибір серверних технологій, баз даних, інтерфейсів і фреймворків для створення динамічних вебдодатків [9].

Однією з найважливіших складових є вибір фреймворку, який забезпечить організацію взаємодії з користувачем та ефективну роботу з даними. Laravel є одним з найбільш популярних фреймворків для створення вебдодатків. Цей фреймворк на PHP має потужний набір інструментів для розробки динамічних вебсайтів і додатків. Laravel пропонує високу продуктивність, зручну роботу з базами даних завдяки ORM (Object-Relational Mapping), систему маршрутизації, а також потужний механізм аутентифікації та авторизації користувачів. Laravel значно спрощує створення масштабованих додатків завдяки вбудованим бібліотекам для роботи з чергами повідомлень, сповіщеннями та механізмами тестування.

Іншим важливим інструментом для розробки адміністративних панелей є Filament — фреймворк, який інтегрується з Laravel і дозволяє створювати інтуїтивно зрозумілі інтерфейси для адміністраторів. Filament значно полегшує процес створення панелей для управління даними, пропонуючи готові рішення для створення інтерфейсів CRUD (Create, Read, Update, Delete). Це дозволяє швидко реалізувати необхідні функції для управління даними користувачів, курсів та платежів у CRM-системі [10].

Що стосується роботи з базами даних, одним із найбільш поширених рішень є використання MySQL або MariaDB, які є реляційними базами даних і добре інтегруються з PHP-фреймворками. Вони дозволяють зберігати дані про клієнтів, курси, зарахування, платежі тощо, забезпечуючи високу надійність і швидкість роботи з великими обсягами інформації. MySQL має потужну підтримку транзакцій та індексації, що дозволяє ефективно працювати з великими даними та виконувати складні запити.

Для створення інтерактивних і зручних інтерфейсів для користувачів важливо використовувати сучасні фронтенд-технології. Найбільш популярними інструментами для розробки таких інтерфейсів є фреймворки та бібліотеки JavaScript, як React, Vue.js або Angular. Вони дозволяють створювати динамічні та інтерактивні веб-сторінки, на яких користувач може зручно працювати з курсами, здійснювати покупки, переглядати інформацію про доступні ресурси або звертатися до служби підтримки [11].

React є однією з найпопулярніших бібліотек для створення інтерфейсів завдяки своїй простоті, ефективності та можливості роботи з компонентами. Завдяки великій екосистемі React дозволяє інтегрувати різноманітні бібліотеки та фреймворки для покращення функціональності додатка, а також легко взаємодіяти з бекенд-серверами через API-запити. Vue.js також є популярним вибором серед розробників завдяки своїй легкості у використанні та гнучкості. Цей фреймворк дозволяє швидко створювати інтерактивні компоненти та підтримує двосторонню прив’язку даних, що робить його зручним для створення динамічних інтерфейсів.

Що стосується безпеки даних, то важливо використовувати сучасні механізми шифрування та захисту особистої інформації користувачів. Для цього в Laravel вже є вбудовані можливості для хешування паролів, а також інтеграція з бібліотеками для реалізації двофакторної аутентифікації. Важливо також використовувати HTTPS для захисту даних, що передаються

між клієнтом і сервером, що дозволяє забезпечити конфіденційність та захист від атак [12].

Завдяки використанню сучасних вебтехнологій можна створити потужний і зручний інструмент для управління взаємовідносинами з клієнтами в онлайн-курсах, що дозволяє не лише ефективно управляти курсами та клієнтами, але й надавати користувачам зручний інтерфейс для взаємодії з системою, підтримку персональних даних та забезпечення високого рівня безпеки.

1.3 Архітектурні підходи до розробки CRM-системи для онлайн-курсів

Архітектурні підходи до розробки CRM-системи для онлайн-курсів є важливим етапом у процесі створення такої системи, оскільки вони визначають, як будуть організовані всі компоненти системи, як вони взаємодіятимуть між собою та з користувачами, а також як буде забезпечена масштабованість, надійність і безпека системи. Вибір архітектури має значний вплив на продуктивність системи, її здатність підтримувати великий обсяг даних і користувачів, а також на зручність її подальшої розширюваності [13].

Одним з найбільш популярних архітектурних підходів для розробки CRM-систем є модель “клієнт-сервер”. Вона передбачає чітке розділення функцій між клієнтською і серверною частинами. Клієнтська частина взаємодіє з користувачем і відправляє запити до серверної частини, яка, у свою чергу, обробляє ці запити, виконує необхідні операції з базою даних і надсилає результати назад клієнту. Така архітектура дозволяє зручно масштабувати систему і робить її незалежною від конкретних пристроїв користувачів [14].

Серверна частина CRM-системи повинна бути спроектована таким чином, щоб забезпечити високу продуктивність і швидкий доступ до даних. У цьому контексті важливими є застосування принципів розподілених систем і використання мікросервісної архітектури. Мікросервіси дозволяють розділити систему на невеликі, автономні модулі, кожен з яких відповідає за конкретну частину функціональності. Наприклад, один мікросервіс може бути відповідальний за управління користувачами, інший — за управління курсами, третій — за обробку платежів, і так далі. Така архітектура дозволяє легше підтримувати систему, оновлювати її окремі компоненти без необхідності переробляти всю систему, а також забезпечує можливість вертикального масштабування [15].

Однією з важливих складових архітектури CRM-системи є вибір бази даних. Для зберігання інформації про клієнтів, курси, зарахування, платежі та інші дані може бути використана реляційна база даних, така як MySQL або PostgreSQL. Вибір реляційної бази даних обумовлений її здатністю ефективно зберігати структуровані дані, забезпечувати транзакційність та підтримку складних запитів, необхідних для аналізу інформації. Для великих систем може бути доцільно впровадити шардінг або реплікацію бази даних для забезпечення високої доступності та масштабованості.

Важливою частиною архітектури є система авторизації та аутентифікації користувачів. Вона повинна бути побудована з урахуванням сучасних вимог до безпеки, таких як двофакторна аутентифікація, захист від атак типу SQL Injection, шифрування даних за допомогою сучасних алгоритмів. У цьому контексті використовуються фреймворки для безпеки, що дозволяють легко інтегрувати такі механізми, як OAuth або JWT для забезпечення безпечного доступу до системи [16].

Для управління інтерфейсами та взаємодією з користувачами важливим елементом є розробка адміністративної панелі, яка дозволить адміністратору керувати курсами, користувачами та іншими аспектами системи. Для цього можуть бути використані готові рішення, такі як Filament для Laravel, що дозволяють швидко створювати зручні інтерфейси для управління даними. Фронтенд може бути реалізований за допомогою сучасних JavaScript-фреймворків, таких як React або Vue.js, що забезпечує високу інтерактивність та зручність використання [17].

Ще одним важливим аспектом архітектури CRM-системи є забезпечення інтеграції з іншими системами, такими як платіжні системи, системи для проведення онлайн-курсів, платформи для відеоконференцій тощо. Всі ці інтеграції можуть бути реалізовані через API, що дозволяє забезпечити зручний доступ до зовнішніх сервісів і забезпечує гнучкість у налаштуванні взаємодії з іншими платформами [18].

Усі ці архітектурні рішення разом забезпечують створення потужної, надійної та масштабованої CRM-системи для онлайн-курсів, яка здатна забезпечити ефективне управління взаємовідносинами з клієнтами, підтримку високої продуктивності та безпеку даних.

РОЗДІЛ ІІ
ОБҐРУНТУВАННЯ І ВИБІР МЕТОДІВ ДЛЯ СТВОРЕННЯ CRM-СИСТЕМИ ДЛЯ ОНЛАЙН-КУРСІВ

2.1 Постановка завдань, огляд та вибір технологій (Laravel, Filament, MySQL) для створення CRM

Створення інструменту управління взаємовідносинами з клієнтами для онлайн-курсів потребує використання сучасних вебтехнологій, які забезпечують ефективність, масштабованість та зручність у користуванні. Основні технології, що застосовуються для розробки таких CRM-систем, поділяються на три рівні: фронтенд (клієнтська частина), бекенд (серверна частина) та база даних. Додатково використовуються API та засоби безпеки для інтеграції та захисту даних [19].

Для фронтенду найчастіше використовують HTML, CSS і JavaScript, оскільки вони є основою веброзробки. Щоб забезпечити сучасний користувацький інтерфейс та зручний UX/UI, застосовуються бібліотеки та фреймворки, такі як React.js, Vue.js або Angular. React.js є одним із найпопулярніших виборів завдяки своїй компонентній архітектурі, що дозволяє створювати повторно використовувані елементи інтерфейсу та ефективно оновлювати сторінку без перезавантаження. Vue.js відзначається простотою у використанні та гарною продуктивністю, що робить його чудовим вибором для невеликих і середніх проектів. Angular, у свою чергу, є потужним фреймворком із вбудованими інструментами для роботи з формами, маршрутизацією та залежностями, що полегшує розробку великих корпоративних додатків [20].

Для бекенду найбільш поширеним вибором є PHP з фреймворком Laravel, оскільки він надає зручні засоби для роботи з базами даних, безпекою та аутентифікацією користувачів. Laravel пропонує інструменти, такі як Eloquent ORM для зручного управління записами у базі даних, Blade для створення динамічних шаблонів та Middleware для обробки запитів.

Альтернативними варіантами можуть бути Node.js (з використанням Express.js) або Django (на Python), що також добре підходять для розробки CRM-систем [21].

Для зберігання даних зазвичай використовується реляційна база даних MySQL або PostgreSQL, оскільки вони надають гнучкі можливості для роботи з транзакціями, масштабуванням та забезпеченням цілісності даних. MySQL є одним із найпопулярніших варіантів завдяки високій продуктивності та сумісності з Laravel, тоді як PostgreSQL забезпечує розширені можливості для роботи зі складними запитами та аналітикою.

Для інтеграції та комунікації між фронтендом і бекендом використовується REST API або GraphQL. REST API дозволяє бекенду передавати дані у форматі JSON, що спрощує взаємодію з клієнтською частиною. GraphQL, на відміну від REST, дозволяє клієнту запитувати лише ті дані, які йому потрібні, що зменшує навантаження на сервер та оптимізує швидкість роботи додатка.

Забезпечення безпеки є критично важливим аспектом розробки CRM-системи. Основними технологіями для цього є JWT (JSON Web Token) для аутентифікації користувачів, OAuth 2.0 для авторизації та SSL/TLS для шифрування даних під час передачі. Laravel надає вбудовані засоби для аутентифікації через Laravel Sanctum або Laravel Passport, що полегшує впровадження безпечної авторизації [22].

Таким чином, розробка CRM-системи для онлайн-курсів ґрунтується на поєднанні сучасних фронтенд- та бекенд-технологій, використанні надійних баз даних та API для інтеграції, а також впровадженні заходів безпеки для захисту даних користувачів. Використання Laravel у поєднанні з MySQL та Vue.js дозволяє створити ефективну, зручну та масштабовану систему для управління взаємовідносинами з клієнтами у сфері онлайн-освіти.

2.2 Опис архітектури CRM-системи на основі Laravel та Filament

Для розробки CRM-системи для онлайн-курсів у межах цієї роботи було обрано зв’язку Laravel та Filament. Це рішення продиктоване рядом переваг, які безпосередньо впливають на швидкість розробки, гнучкість архітектури, безпеку та зручність подальшої підтримки системи.

Laravel є одним із найпопулярніших PHP-фреймворків сучасності, що відзначається зрозумілим синтаксисом, розвиненою екосистемою та активною спільнотою. Однією з ключових переваг Laravel є його модульна структура та підтримка шаблону проєктування MVC (Model-View-Controller), що дозволяє розділяти бізнес-логіку, дані та представлення, забезпечуючи чистоту та порядок у проєкті. Завдяки потужному маршрутизатору, системі міграцій для роботи з базою даних, вбудованим засобам автентифікації, захисту та кешування, Laravel дозволяє швидко створювати складні багатофункціональні системи із високим рівнем безпеки.

Окремо варто відзначити зручність використання ORM Eloquent, яка спрощує взаємодію з базами даних, дозволяючи працювати з таблицями як з об’єктами, що прискорює розробку та зменшує кількість помилок. Ще однією перевагою є наявність великої кількості готових пакетів, що дозволяє розширювати функціонал без необхідності створення всього з нуля. Це особливо актуально для CRM-систем, де часто потрібні рішення для розсилок, звітів, інтеграцій із зовнішніми сервісами.

Для реалізації адміністративної панелі обрано Filament — сучасний фреймворк для створення бек-офісів на основі Laravel. Його головною перевагою є надзвичайно швидка інтеграція з Laravel-проєктами, інтуїтивно зрозумілий інтерфейс і можливість створювати адміністраторські панелі без складного фронтенду. Filament дозволяє за лічені хвилини налаштувати зручні CRUD-операції для роботи з клієнтами, курсами, платежами та іншими даними. Також цей фреймворк підтримує роботу з формами, таблицями, фільтрами, аналітичними віджетами, що дозволяє сформувати повноцінну CRM-панель без значного залучення додаткових бібліотек.

Ще однією важливою перевагою є те, що Filament має сучасний дизайн, адаптивний під різні пристрої, що робить адміністративну панель зручною для роботи як на ПК, так і на планшетах чи смартфонах. Завдяки відкритому коду та активній спільноті розробників, фреймворк постійно оновлюється та розширюється новими модулями, що дозволяє без проблем адаптувати систему до зростаючих потреб освітнього проєкту [23].

У межах розробки CRM-системи для онлайн-курсів було застосовано класичну багаторівневу архітектуру, яка забезпечує розділення відповідальності між різними складовими системи (рис. 2.1). Це дозволяє досягти зручності в розробці, супроводі та масштабуванні системи. Загальна архітектура рішення складається з трьох основних рівнів: Presentation (представлення), Business Logic (бізнес-логіка) та Data Access (доступ до даних).

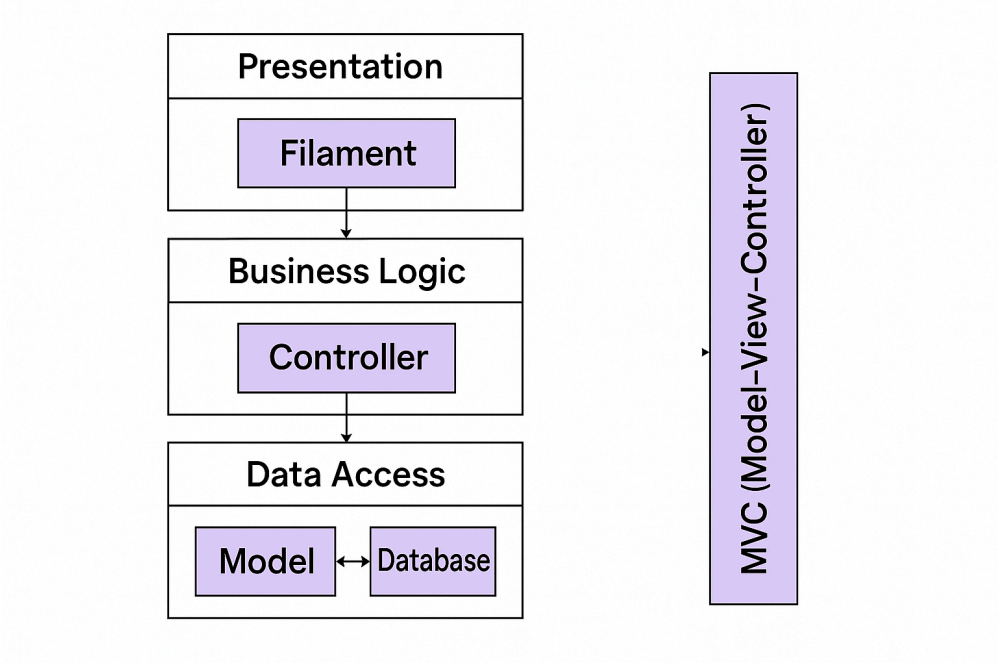

Рисунок 2.1 – схема багаторівневої архітектури

На рівні Presentation реалізовано взаємодію користувача з системою через веб-інтерфейс адміністративної панелі, побудованої на основі Filament. Цей рівень відповідає за відображення інформації та отримання даних від

користувача, надаючи доступ до функцій CRM — керування клієнтами, курсами, заявками, оплатами, статистикою. Використання Filament дозволяє реалізувати сучасний, адаптивний та функціональний інтерфейс без потреби у складній фронтенд-розробці [24].

Рівень Business Logic містить основну бізнес-логіку CRM-системи. Саме тут відбувається обробка запитів, валідація даних, виконання операцій над інформацією про клієнтів, курси, розсилки та інші сутності системи. У Laravel бізнес-логіка реалізована через контролери, сервіси та middleware, які отримують дані з інтерфейсу, обробляють їх і передають для збереження або виведення. Завдяки гнучкій структурі Laravel, бізнес-логіка легко розподіляється між контролерами, окремими сервісами чи класами, що забезпечує зручність у підтримці та розширенні функціоналу системи.

На рівні Data Access реалізовано взаємодію з базою даних MySQL. У Laravel для цього використовується ORM Eloquent, яка дозволяє працювати з базою даних у вигляді об’єктів. Моделі Eloquent відображають структуру таблиць бази даних і забезпечують доступ до даних, їхнє збереження, оновлення та видалення. Завдяки системі міграцій у Laravel, структура бази даних задається у вигляді коду, що спрощує її супровід, внесення змін і забезпечення цілісності даних.

В основі всієї архітектури лежить шаблон проєктування MVC (Model-View-Controller), який є стандартом для побудови вебзастосунків на Laravel. У цій структурі:

- Model відповідає за роботу з базою даних, зберігаючи інформацію про клієнтів, курси, замовлення тощо.
- View відповідає за відображення інформації користувачеві у вигляді адміністративної панелі, яку в цьому проєкті формує Filament.
- Controller приймає запити від користувача, викликає відповідні моделі для обробки даних та повертає результат до View або виконує перенаправлення.

Така багаторівнева архітектура дозволяє ефективно розділити систему на незалежні складові, що спрощує її розробку, тестування, підтримку та розширення. Завдяки застосуванню шаблону MVC у зв’язці з Laravel та Filament CRM-система для онлайн-курсів залишається структурованою, логічно організованою та легко масштабованою [25].

РОЗДІЛ ІІІ
РОЗРОБКА CRM-СИСТЕМИ ДЛЯ ОНЛАЙН-КУРСІВ

3.1 Реалізація бекенд-частини CRM-системи на основі Laravel

Реалізація бекенд-частини CRM-системи на основі Laravel розпочинається зі створення нового проєкту, налаштування середовища та визначення основних компонентів системи. Laravel надає зручний інструментарій для створення масштабованих веб-додатків, тому його використання дозволяє значно спростити розробку.

Laravel використовує командний інтерфейс Artisan для виконання команд. Ось приклад скрипту artisan:

```
#!/usr/bin/env php
<?php
use Illuminate\Foundation\Application;
use Symfony\Component\Console\Input\ArgvInput;
define('LARAVEL_START', microtime(true));
// Register the Composer autoloader...
require __DIR__.'/vendor/autoload.php';
// Bootstrap Laravel and handle the command...
/** @var Application $app */
$app = require_once __DIR__.'/bootstrap/app.php';
$status = $app->handleCommand(new ArgvInput);
exit($status);
```

Цей скрипт є головним виконуваним файлом Artisan – командного інтерфейсу Laravel. Він використовується для запуску Artisan-команд у терміналі.

Перша строка #!/usr/bin/env php вказує, що цей файл слід виконувати за допомогою PHP. Це дозволяє запускати його як звичайний скрипт у терміналі без явного виклику php. Далі підключаються необхідні класи через use Illuminate\Foundation\Application; і use

Symfony\Component\Console\Input\ArgvInput;. Це означає, що скрипт буде працювати з ядром Laravel та отримувати аргументи з командного рядка.

Визначається константа LARAVEL_START, яка фіксує час початку виконання скрипта за допомогою microtime(true). Це може бути корисним для вимірювання продуктивності команд Artisan. Далі виконується підключення Composer autoloader через require __DIR__.'/vendor/autoload.php';, що забезпечує автоматичне завантаження всіх залежностей Laravel.

Потім ініціалізується ядро Laravel через $app = require_once __DIR__.'/bootstrap/app.php';. Це створює екземпляр Application, який є основою всієї Laravel-програми. Далі викликається метод $app->handleCommand(new ArgvInput);, який передає введені користувачем аргументи в Artisan і запускає відповідну команду. В кінці результат команди зберігається у змінній $status, і виконання завершується командою exit($status);, передаючи отриманий код завершення назад у командний рядок.

Цей скрипт є важливою частиною Laravel, оскільки дозволяє керувати міграціями, сидерами, кешем, чергами, створювати контролери, моделі, та виконувати інші дії в рамках проєкту без необхідності безпосередньої взаємодії з кодом через браузер.

У файлі routes/console.php визначаються консольні команди:

```
<?php

use Illuminate\Foundation\Inspiring;
use Illuminate\Support\Facades\Artisan;

Artisan::command('inspire', function () {
    $this->comment(Inspiring::quote());
})->purpose('Display an inspiring quote');
```

Тут спочатку підключаються необхідні класи: Illuminate\Foundation\Inspiring використовується для отримання випадкової мотивуючої цитати, а Illuminate\Support\Facades\Artisan надає доступ до Artisan-команд. Далі за допомогою Artisan::command('inspire', function () { ... }) реєструється нова консольна команда inspire, яку можна викликати через термінал командою php artisan inspire.

В середині анонімної функції викликається метод $this->comment(Inspiring::quote());, який виводить у консоль випадкову цитату. Це робиться через клас Inspiring, який містить колекцію мотивуючих висловів.

В кінці до команди додається метод ->purpose('Display an inspiring quote');, який описує призначення команди. Це допомагає, коли користувач викликає php artisan list для перегляду доступних команд, оскільки опис допомагає зрозуміти, що саме робить команда inspire.

Файл routes/web.php містить маршрути для веб-інтерфейсу:

```
<?php

use Illuminate\Support\Facades\Route;

Route::get('/', function () {
    return view('welcome');
});
```

Спочатку підключається клас Illuminate\Support\Facades\Route, який надає методи для роботи з маршрутами. Далі викликається Route::get('/', function () { ... });, що створює маршрут для HTTP-запиту GET на головну сторінку (/). Коли користувач відкриває головну сторінку сайту, виконується передана анонімна функція.

У цій функції викликається return view('welcome');, що повертає представлення welcome.blade.php, яке знаходиться у папці resources/views/. Це стандартна стартова сторінка Laravel.

3.2 Використання Filament для створення адміністративної панелі

Filament — це потужний інструмент для створення адміністративних панелей у Laravel, який надає простий спосіб керування ресурсами бази даних через UI. У цьому розділі розглянемо, як реалізовано ресури: CourseResource, EnrollmentResource, PaymentResource, StudentResource, TeacherResource.

Почнемо з CourseResource. Це клас є ресурсом Filament, який відповідає за CRUD-операції для моделі Course. Він знаходиться у просторі імен App\Filament\Resources і розширює базовий клас Filament\Resources\Resource.

Визначення моделі та налаштування навігації:

```
protected static ?string $model = Course::class;

protected static ?string $navigationIcon = 'heroicon-o-book-open';

protected static ?string $navigationGroup = 'Course Management';
```

Тут вказується, що цей ресурс відповідає моделі Course. Також задається значок (navigationIcon) і група навігації (navigationGroup), щоб організувати адміністративну панель.

Налаштування форми для створення та редагування курсів:

```
Forms\Components\TextInput::make('title')
    ->required()
    ->maxLength(255);
```

Метод form(Form $form): Form визначає поля форми, які будуть використовуватися в інтерфейсі адміністративної панелі. Цей елемент додає

текстове поле для назви курсу, роблячи його обов’язковим (required()) та обмежуючи довжину (maxLength(255)).

```
Forms\Components\Select::make('teacher_id')
    ->label('Teacher')
    ->options(Teacher::where('status', 'active')->pluck('name', 'id'))
    ->required()
    ->searchable();
```

Поле вибору Select дозволяє вибрати викладача з активних у базі даних (where('status', 'active')). Поле є обов’язковим і підтримує пошук.

Налаштування таблиці для перегляду курсів:

```
Tables\Columns\TextColumn::make('title')
    ->searchable();
```

Метод table(Table $table): Table визначає, які стовпці будуть відображатися у списку курсів. Додається колонка з назвою курсу, яка підтримує пошук (searchable()).

```
Tables\Columns\TextColumn::make('teacher.name')
    ->searchable()
    ->sortable();
```

Додається колонка для викладача, яка дозволяє шукати курси за викладачем і сортувати їх.

Додавання фільтрів:

```
Tables\Filters\SelectFilter::make('status')
    ->options([
        'upcoming' => 'Upcoming',
        'active' => 'Active',
        'completed' => 'Completed',
        'cancelled' => 'Cancelled',
    ]);
```

Фільтр SelectFilter дозволяє фільтрувати курси за статусом (upcoming, active, completed, cancelled).

Додавання дій над записами:

```
Tables\Actions\EditAction::make();
```

Додається кнопка редагування курсу.

Додається масова дія, яка дозволяє експортувати вибрані курси у форматі Excel. Аналогічно у headerActions додається кнопка для експорту всіх курсів:

```
ExportBulkAction::make()
    ->label('Export Selected')
    ->exports([
        ExcelExport::make()
            ->fromTable()
            ->withFilename('courses-' . now()->format('Y-m-d'))
            ->withWriterType(\Maatwebsite\Excel\Excel::XLSX)
    ]),
```

Налаштування сторінок ресурсу:

```
public static function getPages(): array
{
    return [
        'index' => Pages\ListCourses::route('/'),
        'create' => Pages\CreateCourse::route('/create'),
        'edit' => Pages\EditCourse::route('/{record}/edit'),
    ];
}
```

Тут визначаються маршрути для сторінок списку курсів (ListCourses), створення (CreateCourse) та редагування (EditCourse).

Далі перейдемо до EnrollmentResource. Він є ресурсом для управління зарахуваннями студентів на курси у Laravel за допомогою Filament. Він дозволяє адміністратору створювати, редагувати та переглядати записи про зарахування студентів на конкретні курси. Тепер давайте розглянемо кожну частину цього ресурсу детальніше.

Модель і навігація:

```
protected static ?string $model = Enrollment::class;
protected static ?string $navigationIcon = 'heroicon-o-clipboard-document-list';
protected static ?string $navigationGroup = 'Course Management';
```

Перші кілька рядків визначають, яка модель буде використовуватись для цього ресурсу. У цьому випадку це модель Enrollment, що відповідає за збереження даних про зарахування. Визначено також іконку для навігації у адмін-панелі (це іконка для списку документів) та групу навігації, до якої буде належати цей ресурс. Вона належить групі “Course Management” (Управління курсами), що допомагає організувати ресурси в адмінці.

Форма для створення та редагування записів:

```
public static function form(Form $form): Form
{
    return $form->schema([
        Forms\Components\Section::make('Enrollment Information')
            ->schema([
                Forms\Components\Select::make('student_id')
                    ->label('Student')
                    ...
                    ->default('pending')
                    ->required(),
            ])->columns(2),
```

Перша частина форми описує поля, необхідні для введення основної інформації про зарахування. Адміністратор має можливість вибрати студента та курс з відповідних списків. Це поля student_id та course_id, де вибір студентів обмежений активними студентами, а курси мають статус “upcoming” або “active”. Обидва ці поля є обов’язковими для заповнення.

Також є поле для вибору дати зарахування, яке за замовчуванням встановлюється на поточну дату. Для статусу зарахування пропонується кілька варіантів: “pending”, “active”, “completed” або “cancelled”. Це дозволяє відстежувати поточний статус кожного запису зарахування.

Другий розділ форми зосереджений на інформації про платіж. Тут адміністратор може ввести суму, яку студент вже сплатив за курс, а також додаткові нотатки, які можуть містити деталі платежу чи іншої корисної інформації. Код наведено нижче:

```
Forms\Components\Section::make('Payment Information')
    ->schema([
        Forms\Components\TextInput::make('amount_paid')
            ->label('Amount Paid')
            ...
        Forms\Components\Textarea::make('notes')
            ->maxLength(65535)
            ->columnSpanFull(),
    ]),
]);
```

Таблиця для перегляду даних:

```
public static function table(Table $table): Table
{
    return $table->columns([
        Tables\Columns\TextColumn::make('student.name')
            ...
        Tables\Columns\TextColumn::make('course.title')
            ...
        Tables\Columns\TextColumn::make('enrollment_date')
            ->date()
            ->sortable(),
```

У таблиці відображаються основні дані про зарахування: ім’я студента, назва курсу та дата зарахування. Всі ці колонки можна сортувати, що дає адміністратору можливість зручно переглядати записи, відсортовані за певними критеріями.

Окрім базових даних, таблиця також відображає статус зарахування, при цьому статус відображається у вигляді кольорових міток (badge), що дозволяє швидко визначити стан кожного зарахування. Є також інформація

про суму, сплачену студентом, ціну курсу, а також кількість платежів, які було здійснено за цей запис. Код наведено нижче.

```
Tables\Columns\TextColumn::make('status')
        ->badge()
        ->color(fn (string $state): string => match ($state) {
            'pending' => 'warning',
            ...
        }),
    Tables\Columns\TextColumn::make('amount_paid')
                                        ...
```

Далі налаштовуються фільтри для швидкого пошуку та сортування даних. Адміністратор може фільтрувати записи за студентом, курсом або статусом зарахування. Це дозволяє звузити пошук і знайти необхідні записи, навіть якщо база даних містить багато записів.

```
->filters([
    Tables\Filters\SelectFilter::make('student')
        ->relationship('student', 'name'),
    Tables\Filters\SelectFilter::make('course')
        ->relationship('course', 'title'),
    ...
  ])
```

Таблиця також підтримує операції редагування для окремих записів і масові операції, такі як видалення кількох записів одночасно. Це значно спрощує управління великими обсягами даних:

```
->actions([
    Tables\Actions\EditAction::make(),
])
->bulkActions([
    Tables\Actions\BulkActionGroup::make([
        Tables\Actions\DeleteBulkAction::make(),
    ]),
]);
```

Далі перейдемо до EnrollmentResource. це ресурс для керування платежами в адміністративній панелі Filament. Він дозволяє здійснювати операції з додавання, редагування та перегляду платежів, пов’язаних з певними реєстраціями студентів на курси. Ресурс включає форми для введення даних про платіж, таблицю для відображення списку платежів, а також фільтри та можливості для масових дій. Використовується в рамках системи фінансового управління, щоб забезпечити ефективний моніторинг і управління транзакціями.

Назва і налаштування:

```
protected static ?string $model = Payment::class;
protected static ?string $navigationIcon = 'heroicon-o-currency-dollar';
protected static ?string $navigationGroup = 'Financial Management';
```

Ці налаштування визначають основні характеристики ресурсу PaymentResource. $model вказує, що цей ресурс працює з моделлю Payment. Іконка навігації та група навігації визначають зовнішній вигляд цього ресурсу в панелі керування Filament. Це дозволяє зручно знаходити ресурс у відповідній секції фінансового управління.

Метод form:

```
public static function form(Form $form): Form
{
    return $form
        ->schema([
            Forms\Components\Section::make('Payment Information')
                ->schema([
                    Forms\Components\Select::make('enrollment_id')
                        ->label('Enrollment')
                        ->options(function () {
                            return Enrollment::with(['student', 'course'])
```

У цьому методі описується форма для введення даних платежу. Вибір студента та курсу здійснюється через компонент Select, що дозволяє вибирати зі списку зареєстрованих учнів та курсів. Поле для суми платежу обов’язкове й має числовий формат із префіксом долара. Дата платежу автоматично встановлюється на поточну, а метод оплати може бути одним із варіантів (готівка, кредитна картка, банківський переказ або онлайн).

Додаткові дані у формі:

```
Forms\Components\Section::make('Additional Information')
    ->schema([
        Forms\Components\TextInput::make('transaction_id')
            ->maxLength(255),
        Forms\Components\Select::make('status')
            ->options([
                'pending' => 'Pending',
                'completed' => 'Completed',
                'failed' => 'Failed',
                'refunded' => 'Refunded',
            ])
```

Цей блок додає додаткові поля до форми, такі як ID транзакції, статус платежу (очікується, завершено, не вдалося, повернено) та примітки, що дають можливість надати додаткову інформацію про платіж.

Метод table:

```
public static function table(Table $table): Table
{
    return $table
        ->columns([
            Tables\Columns\TextColumn::make('enrollment.student.name')
                ->label('Student')
                ->searchable()
                ->sortable(),
```

Цей код описує таблицю для відображення інформації про платежі. Для кожного поля в таблиці задані опції, такі як сортування, пошук і форматування. Статус платежу відображається з використанням кольорових беджів для кожного статусу, що дозволяє легко відрізняти різні стани платежу.

Фільтри таблиці дозволяють користувачам відфільтровувати платежі за методом оплати, статусом та діапазоном дат. Це дає змогу зручніше знаходити потрібні записи, особливо при великих обсягах даних. Код наведено нижче:

```
->filters([
    Tables\Filters\SelectFilter::make('payment_method')
        ->options([
            'cash' => 'Cash',
            'credit_card' => 'Credit Card',
            'bank_transfer' => 'Bank Transfer',
            'online' => 'Online Payment',
        ]),
```

Дії та групові дії:

```
->bulkActions([
    Tables\Actions\BulkActionGroup::make([
        Tables\Actions\DeleteBulkAction::make()->after(function () {
            Enrollment::all()->each(function ($enrollment) {
                $totalPaid = $enrollment->payments()->where('status', 'completed')->sum('amount');
                $enrollment->update(['amount_paid' => $totalPaid]);
            });
        }),
    ]),
]);
```

Цей блок додає можливість редагувати платежі після їх внесення та здійснювати групові операції, такі як видалення кількох записів одночасно. Після кожної операції, наприклад, редагування чи видалення, автоматично оновлюється інформація про загальну суму оплаченої вартості в моделі Enrollment.

Метод getPages:

```
public static function getPages(): array
{
    return [
        'index' => Pages\ListPayments::route('/'),
        'create' => Pages\CreatePayment::route('/create'),
        'edit' => Pages\EditPayment::route('/{record}/edit'),
    ];
}
```

Метод getPages визначає сторінки, доступні для цього ресурсу. Вони включають сторінку списку платежів, сторінку створення нового платежу та сторінку редагування існуючого платежу. Це дозволяє створити зручний інтерфейс для роботи з платежами в адмінпанелі.

Наступний на черзі йде StudentResource. Це ресурс для керування студентами в адміністративній панелі Filament. Він дозволяє адміністраторам додавати, редагувати та переглядати інформацію про студентів, включаючи контактні дані, статус та пов’язані з ними курси. Також ресурс включає можливості для масового експорту даних про студентів у форматі Excel та фільтрації студентів за їх статусом. Цей ресурс використовує компоненти Filament для створення форм, таблиць і фільтрів для зручного управління студентами.

Опис моделі та навігації:

```
protected static ?string $model = Student::class;
protected static ?string $navigationIcon = 'heroicon-o-user-group';
protected static ?string $navigationGroup = 'User Management';
```

Тут вказується модель, яка використовується для цього ресурсу — це Student, що означає, що цей ресурс працює з таблицею студентів. Також вказано іконку для навігаційного меню і групу, до якої цей ресурс належить, що дає адміністраторам змогу легко знаходити його в панелі управління.

Форма для додавання та редагування студентів:

```
public static function form(Form $form): Form
{
    return $form
        ->schema([
            Forms\Components\Section::make('Student Information')
                ->schema([
                    Forms\Components\TextInput::make('name')
                        ->required()
                        ->maxLength(255),
```

Цей код створює форму для введення інформації про студента, включаючи ім’я, email, телефон та дату народження. Всі поля мають відповідні правила валідації: деякі є обов’язковими, інші мають обмеження на довжину або повинні бути унікальними. Це дає змогу ввести повну інформацію про студента.

Таблиця для відображення студентів:

```
public static function table(Table $table): Table
{
    return $table
        ->columns([
            Tables\Columns\TextColumn::make('name')->searchable(),
            Tables\Columns\TextColumn::make('email')->searchable(),
            Tables\Columns\TextColumn::make('phone')->searchable(),
        ]);
}
```

Цей код описує таблицю для відображення студентів, що містить стовпці з ім’ям, email та телефоном студента. Ці стовпці є пошуковими, що дозволяє швидко знаходити студентів за певними даними. Відображення студентів у таблиці дозволяє адміністратору ефективно керувати записами.

Додавання фільтрів та масових дій:

```
->filters([
    Tables\Filters\SelectFilter::make('status')
        ->options([
            'active' => 'Active',
            'inactive' => 'Inactive',
            'on_hold' => 'On Hold',
        ]),
])
```

Цей код додає фільтр для вибору статусу студентів (активний, неактивний, на паузі), а також масові дії. Адміністратор може вибрати декілька студентів для виконання дій, таких як видалення або експортування вибраних студентів у форматі Excel. Масові дії підвищують ефективність роботи з великою кількістю студентів.

Експорт всіх студентів у Excel:

```
->headerActions([
    ExcelExportAction::make()
        ->label('Export All')
        ->exports([
            ExcelExport::make()
                ->fromTable()
                ->withFilename('students-' . now()->format('Y-m-d'))
                ->withWriterType(\Maatwebsite\Excel\Excel::XLSX)
        ]),
]);
```

Цей код додає кнопку для експорту всіх записів про студентів у Excel у заголовок таблиці. Це дає змогу адміністратору завантажити всю інформацію

про студентів у вигляді Excel-файлу, що може бути корисним для подальшого аналізу чи архівування.

Реляційні менеджери:

```
public static function getRelations(): array
{
    return [
        RelationManagers\EnrollmentsRelationManager::class,
    ];
}
```

Цей метод повертає реляційні менеджери, які визначають зв’язки з іншими моделями. У даному випадку, це менеджер для керування записами про зарахування студентів, що дозволяє переглядати курси, на які зараховані студенти.

І останнім розглянемо TeacherResource — це ресурс для керування інформацією про викладачів в адміністративній панелі Filament. Цей ресурс дозволяє адміністраторам додавати, редагувати та переглядати деталі викладачів, такі як ім’я, email, телефон, спеціальність, ставка за годину, статус і біографія. Він також включає можливість сортування та редагування даних у таблиці.

Опис моделі та навігації:

```
protected static ?string $model = Teacher::class;
protected static ?string $navigationIcon = 'heroicon-o-rectangle-stack';
```

Тут вказується модель, яка використовується для цього ресурсу — це Teacher, що означає, що ресурс працює з таблицею викладачів. Іконка heroicon-o-rectangle-stack визначає вигляд значка для цього ресурсу в навігації, що дозволяє адміністраторам швидко знаходити ресурс.

Форма для додавання та редагування викладачів:

```
public static function form(Form $form): Form
{
    return $form
        ->schema([
            Forms\Components\TextInput::make('name')
                ->required()
                ->maxLength(255),
            Forms\Components\TextInput::make('email')
                ->email()
```

Цей код створює форму для введення інформації про викладача. Поля, такі як ім’я, email, телефон, спеціальність, ставка за годину та статус є обов’язковими. Біографія викладача вводиться через текстову область, яка займає всю ширину колонки. Всі ці поля мають відповідні правила валідації та обмеження на довжину.

Таблиця для відображення викладачів:

```
public static function table(Table $table): Table
{
    return $table
        ->columns([
            Tables\Columns\TextColumn::make('name')
                ->searchable(),
            Tables\Columns\TextColumn::make('email')
                ->searchable(),
```

Цей код створює таблицю для відображення викладачів у адміністративній панелі. Вона містить стовпці для імені, email, телефону, спеціальності, ставки за годину та статусу. Для деяких стовпців доступні функції пошуку та сортування, а також можливість редагування за допомогою кнопки

редагування. Масові дії дозволяють виконувати операції, такі як видалення кількох записів одночасно.

Сторінки ресурсу:

```
public static function getPages(): array
{
    return [
        'index' => Pages\ListTeachers::route('/'),
        'create' => Pages\CreateTeacher::route('/create'),
        'edit' => Pages\EditTeacher::route('/{record}/edit'),
    ];
}
```

Цей метод визначає сторінки для ресурсу. Вони дозволяють адміністратору переглядати список викладачів, створювати нових викладачів або редагувати існуючих.

3.3 Інтеграція бази даних MySQL для зберігання інформації про клієнтів та курси

Інтеграція бази даних є важливим етапом у розробці системи, оскільки вона забезпечує збереження, обробку та доступ до даних у структурованому вигляді. Для ефективного управління інформацією про клієнтів та курси необхідно налаштувати базу даних, яка буде містити таблиці для зберігання всіх необхідних даних. У нашому випадку ми використовуватимемо MySQL — одну з найбільш популярних і надійних систем управління базами даних, що дозволяє виконувати складні запити та операції з великими обсягами даних.

Основними об’єктами, які ми будемо зберігати в базі даних, є клієнти та курси. Для кожного клієнта потрібно зберігати такі дані, як ім’я, контактна інформація, статус та історія відвідувань курсів. Кожен курс, в свою чергу, має містити такі параметри, як назва, опис, тривалість, ціна та інші

характеристики, які дозволяють ефективно управляти навчальними програмами.

Для реалізації цього плану необхідно створити відповідні моделі в нашій програмі, які відповідатимуть за взаємодію з базою даних. Кожна модель буде відображати відповідну таблицю в базі даних і надаватиме зручний інтерфейс для роботи з даними. У цьому пункті буде розглянуто процес створення моделей для клієнтів і курсів, що дозволить нам організувати структуру бази даних та забезпечити ефективну роботу з даними через ORM (Object-Relational Mapping) у Laravel.

Модель Course у Laravel відповідає за збереження та обробку даних про курси у базі даних. Вона використовує функціональність Laravel Eloquent ORM для взаємодії з таблицею курсів. У простір імен App\Models вноситься відповідна модель, а також підключаються необхідні фасади для роботи з базою даних, фабриками та відносинами між моделями. Важливими частинами цієї моделі є властивості $fillable і $casts.

Властивість $fillable визначає атрибути, які можуть бути масово заповнені, тобто дозволяє масово присвоювати значення для певних полів у базі даних. Це захищає від масового присвоєння небажаних значень для полів, які не повинні бути змінені користувачем. У випадку моделі Course до масового заповнення доступні такі поля як: title, description, teacher_id, price, duration_weeks, start_date, end_date, status, та max_students. Код наведено нижче:

```
protected $fillable = [
    'title',
    'description',
    'teacher_id',
    'price',
    'duration_weeks',
    'start_date',
    'end_date',
    'status',
    'max_students',
  ];
```

Властивість $casts використовується для визначення типів даних для полів, які зберігаються у базі даних. Наприклад, для полів start_date та end_date вказано тип date, що означає, що ці атрибути будуть автоматично перетворюватися у об’єкти типу Carbon, що дозволяє зручніше працювати з датами.

```
protected $casts = [
    'start_date' => 'date',
    'end_date' => 'date',
  ];
```

Модель також визначає методи для роботи з відносинами між таблицями. Метод teacher() описує зв’язок “один до багатьох” між курсом та викладачем. Це означає, що кожен курс має одного викладача, але викладач може мати кілька курсів. Зв’язок реалізується за допомогою методу belongsTo(), який вказує, що курс належить певному викладачу.

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

Метод enrollments() описує відносини між курсом та записами про зарахування студентів. Один курс може мати багато записів про зарахування, що дозволяє відслідковувати, які студенти записалися на конкретний курс. Зв'язок реалізується через метод hasMany(), який вказує на множинність відносин.

```
public function enrollments(): HasMany
    {
        return $this->hasMany(Enrollment::class);
    }
```

Метод students() використовує зв'язок "багато до багатьох" між курсом та студентами через посередницьку таблицю enrollments. Це означає, що один курс може мати багато студентів, і один студент може бути зарахований на кілька курсів. Цей зв'язок реалізується через метод belongsToMany().

```
public function students()
    {
        return $this->belongsToMany(Student::class, 'enrollments');
    }
```

Наступною йде модель Enrollment. Вона представляє запис про зарахування студента на курс. Вона використовується для керування інформацією про студентів, які записалися на певні курси, та їхні платежі.

У моделі використовується простір імен App\Models, а також підключаються необхідні трейти та класи, зокрема HasFactory для роботи з фабриками та Model для взаємодії з базою даних через Eloquent ORM.

Властивість $fillable містить список полів, які можуть бути масово заповнені. Це захищає від несанкціонованого масового присвоєння значень. У цій моделі дозволено заповнювати такі поля: student_id (ідентифікатор студента), course_id (ідентифікатор курсу), enrollment_date (дата запису), status (статус зарахування), amount_paid (сума сплачених коштів) і notes (примітки).

```
protected $fillable = [
    'student_id',
    'course_id',
    'enrollment_date',
    'status',
    'amount_paid',
    'notes',
];
```

Властивість $casts використовується для автоматичного приведення певних полів до необхідних типів. У даному випадку поле enrollment_date визначене як date, що означає автоматичне перетворення його у об’єкт Carbon, що полегшує роботу з датами.

```
protected $casts = [
    'enrollment_date' => 'date',
];
```

Метод student() описує зв’язок “багато до одного” між записом про зарахування та студентом. Це означає, що кожен запис про зарахування належить одному студенту. Для реалізації цього використовується метод belongsTo(), який встановлює відношення до моделі Student.

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

Метод course() визначає аналогічний зв’язок “багато до одного” між записом про зарахування та курсом. Це означає, що кожен запис про зарахування належить одному конкретному курсу.

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

Метод payments() описує зв’язок “один до багатьох” між записом про зарахування та платежами. Оскільки студент може здійснювати кілька платежів у межах одного курсу, використовується метод hasMany(), який встановлює зв’язок із моделлю Payment.

```
public function payments(): HasMany
{
    return $this->hasMany(Payment::class);
}
```

Таким чином, модель Enrollment є центральною ланкою між студентами, курсами та платежами. Вона дозволяє керувати процесом запису на курси, зберігати інформацію про статус зарахування та контроль оплати, а також пов’язує студентів із курсами у системі Laravel.

Далі поговоримо про модель Payment. Вона використовується для збереження інформації про платежі, здійснені студентами за курси. Вона належить до простору імен App\Models і використовує стандартний набір класів для роботи з базою даних через Eloquent ORM. У моделі підключено

трейт HasFactory, що дозволяє зручно створювати екземпляри цієї моделі у фабриках під час тестування або заповнення бази даних.

Список $fillable визначає поля, які дозволено масово заповнювати. До них належать enrollment_id (ідентифікатор запису про зарахування), amount (сума платежу), payment_date (дата здійснення платежу), payment_method (метод оплати, наприклад, картка, банківський переказ тощо), transaction_id (унікальний ідентифікатор транзакції), status (статус платежу, наприклад, “підтверджено” або “очікується”), а також notes (примітки щодо платежу).

```
protected $fillable = [
    'enrollment_id',
    'amount',
    'payment_date',
    'payment_method',
    'transaction_id',
    'status',
    'notes',
];
```

Властивість $casts використовується для автоматичного приведення значень полів до відповідних типів. У цьому випадку поле payment_date приводиться до типу date, що дозволяє зручно працювати з датами за допомогою класу Carbon.

```
protected $casts = [
    'payment_date' => 'date',
];
```

Метод enrollment() визначає зв’язок “багато до одного” між платежами та записами про зарахування (Enrollment). Це означає, що кожен платіж прив’язаний до конкретного запису про зарахування студента на курс. Для

цього використовується метод belongsTo(), який встановлює зв’язок із моделлю Enrollment.

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

Таким чином, модель Payment відіграє важливу роль у системі управління платежами за курси, забезпечуючи збереження інформації про суми, дати, методи оплати та статуси транзакцій. Це дозволяє ефективно керувати фінансовими операціями, пов’язаними з навчальним процесом.

Також присутня модель Student. Вона представляє студентів у системі управління навчальними курсами. Вона розташована у просторі імен App\Models та використовує HasFactory для спрощення створення екземплярів моделі у фабриках під час тестування або наповнення бази даних.

Поле $fillable визначає список атрибутів, які можна масово заповнювати при створенні або оновленні запису. Сюди входять name (ім’я студента), email (адреса електронної пошти), phone (номер телефону), address (адреса проживання), date_of_birth (дата народження), status (статус студента, наприклад, “активний” або “відрахований”), а також notes (додаткові примітки).

```
protected $fillable = [
    'name',
    'email',
    'phone',
    'address',
    'date_of_birth',
    'status',
    'notes',
];
```

Метод enrollments() визначає зв’язок “один до багатьох” між студентом та записами про зарахування (Enrollment). Це означає, що кожен студент може мати багато записів про зарахування на різні курси. Встановлення цього зв’язку виконується через метод hasMany(), який пов’язує студента із записами у таблиці enrollments.

```
public function enrollments(): HasMany
{
    return $this->hasMany(Enrollment::class);
}
```

Метод courses() реалізує зв’язок “багато до багатьох” між студентами та курсами через проміжну таблицю enrollments. Оскільки студент може записатися на кілька курсів, а кожен курс може мати багато студентів, використовується метод belongsToMany(), що дозволяє отримувати всі курси, на які записаний студент.

```
public function courses()
{
    return $this->belongsToMany(Course::class, 'enrollments');
}
```

Таким чином, модель Student відіграє ключову роль у збереженні інформації про студентів та їхні взаємозв’язки з навчальними курсами. Вона дозволяє легко отримувати список курсів, на які записаний студент, а також відстежувати всі його зарахування та статус у навчальній системі.

Модель Teacher у Laravel представляє викладачів у системі управління курсами. Вона розташована у просторі імен App\Models і використовує трейт HasFactory для спрощення створення екземплярів цієї моделі в фабриках, що корисно під час тестування або заповнення бази даних.

Поле $fillable визначає список атрибутів, які можна масово заповнювати при створенні або оновленні запису. У цьому випадку до списку входять: name (ім’я викладача), email (електронна пошта викладача), phone (номер телефону), specialty (спеціалізація викладача), hourly_rate (погодинна ставка), status (статус викладача, наприклад, “активний” або “неактивний”) і bio (біографія викладача).

```
protected $fillable = [
    'name',
    'email',
    'phone',
    'specialty',
    'hourly_rate',
    'status',
    'bio',
];
```

Метод courses() визначає зв’язок “один до багатьох” між викладачем і курсами, які він викладає. Це означає, що кожен викладач може викладати кілька курсів. Встановлення цього зв’язку відбувається через метод hasMany(), який дозволяє отримати всі курси, пов’язані з конкретним викладачем.

```
public function courses(): HasMany
{
    return $this->hasMany(Course::class);
}
```

Таким чином, модель Teacher є важливою частиною системи, оскільки дозволяє зберігати та обробляти інформацію про викладачів, їхні курси та інші важливі атрибути, такі як спеціалізація, ставка та біографія. Вона також дозволяє легко взаємодіяти з курсами, що асоційовані з конкретним викладачем.

І останньою є модель User. Вона представляє користувачів системи. Вона розташована у просторі імен App\Models та використовує класи HasFactory і Notifiable. Трейт HasFactory дозволяє легко створювати екземпляри цієї моделі для тестування або наповнення бази даних через фабрики, а трейт Notifiable додає можливість надсилання повідомлень користувачам, наприклад, для підтвердження реєстрації або відновлення паролю.

Поле $fillable визначає атрибути, які можуть бути масово заповнені при створенні або оновленні запису. Для моделі User це: name (ім’я користувача), email (електронна пошта користувача) та password (пароль користувача).

```
protected $fillable = [
    'name',
    'email',
    'password',
];
```

Поле $hidden визначає атрибути, які будуть приховані при серіалізації моделі, тобто, коли її дані передаються у вигляді масиву чи JSON. У цьому випадку, для безпеки, password та remember_token не будуть відображатися при серіалізації, щоб уникнути витоку чутливої інформації.

```
protected $hidden = [
    'password',
    'remember_token',
];
```

Метод casts() вказує, які атрибути повинні бути приведені до певних типів під час доступу до них. Для моделі User атрибут email_verified_at буде автоматично приведений до типу datetime, що дозволяє зручно працювати з датами, а атрибут password буде збережений у вигляді хешованого значення для забезпечення безпеки.

```
protected function casts(): array
{
    return [
        'email_verified_at' => 'datetime',
        'password' => 'hashed',
    ];
}
```

Модель User є основною для реалізації користувацької автентифікації в Laravel. Вона містить усі необхідні атрибути та методи для взаємодії з базою даних, а також підтримує функціональність для обробки паролів, підтвердження електронної пошти та інших аспектів користувацького обліку.

ВИСНОВКИ

У ході виконання даної дипломної роботи було розроблено модуль CRM-системи для управління взаємовідносинами з клієнтами в онлайн-курсах, використовуючи сучасні вебтехнології, зокрема PHP, Laravel та Filament. Важливим етапом дослідження стало вивчення існуючих CRM-рішень, що застосовуються у сфері онлайн-освіти, аналіз їхніх можливостей та обмежень. Було визначено, що більшість готових рішень мають високий рівень складності, надлишкову функціональність або недостатню адаптованість до специфічних потреб онлайн-курсів, що зумовило необхідність розробки власного інструменту з урахуванням конкретних вимог.

Проведений аналіз технологій дозволив обґрунтувати вибір стеку розробки. Laravel було обрано як основний фреймворк для серверної частини завдяки його гнучкості, високому рівню безпеки, розвиненій екосистемі та підтримці сучасних підходів до розробки вебзастосунків. Використання цього фреймворку забезпечило ефективну обробку запитів, зручну організацію бізнес-логіки та можливість масштабування системи. Filament було обрано як інструмент для створення адміністративної панелі, що значно спростило процес розробки завдяки своїй простоті у використанні, інтеграції з Laravel та підтримці широкого спектра готових компонентів для керування базою даних. MySQL використовувався для зберігання структурованої інформації про користувачів, курси, оплати та інші ключові елементи системи, що дозволило організувати швидку обробку даних та їхню надійну збереженість.

Адміністративна панель на основі Filament забезпечує зручний доступ до управління курсами, студентами, викладачами, оплатами та розсилками. Використання цього інструменту дозволило швидко реалізувати гнучку систему фільтрації, сортування та редагування записів без необхідності ручного створення додаткових CRUD-операцій. Це значно скоротило час розробки та дозволило сфокусуватися на реалізації специфічної бізнес-логіки системи.

Функціонал модулю CRM-системи включає можливість реєстрації нових курсів, додавання матеріалів, управління групами студентів та відстеження їхнього прогресу. Реалізована система також підтримує автоматичне формування звітів про успішність студентів та фінансові надходження, що допомагає викладачам та адміністраторам ефективніше планувати навчальний процес та фінансові операції.

Практичне значення виконаної роботи полягає у створенні ефективного, зручного та адаптивного інструменту для управління взаємовідносинами з клієнтами у сфері онлайн-освіти. Розроблена система може бути легко розширена та налаштована відповідно до потреб конкретного навчального закладу або платформи. Її інтеграція з зовнішніми сервісами, можливість масштабування та використання сучасних технологій роблять цей модуль перспективним рішенням для автоматизації процесів управління курсами.

Таким чином, у ході виконання дипломної роботи було успішно досягнуто поставленої мети – розроблено модуль CRM-систему для онлайн-курсів, яка відповідає сучасним вимогам ефективного управління навчальним процесом. Виконані завдання дозволили створити стабільну, безпечну та масштабовану платформу, яка сприяє покращенню організації навчання, спрощує адміністрування курсів та підвищує рівень взаємодії між студентами та викладачами.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Laravel Documentation. URL: https://laravel.com/docs (дата звернення: 29.03.2025).
2. Filament Documentation. URL: https://filamentphp.com/docs (дата звернення: 29.03.2025).
3. PHP Manual. URL: https://www.php.net/manual/en/ (дата звернення: 29.03.2025).
4. MySQL Documentation. URL: https://dev.mysql.com/doc/ (дата звернення: 29.03.2025).
5. W3Schools PHP Tutorial. URL: https://www.w3schools.com/php/ (дата звернення: 29.03.2025).
6. MDN Web Docs: JavaScript. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript (дата звернення: 29.03.2025).
7. Stack Overflow - Laravel Questions. URL: https://stackoverflow.com/questions/tagged/laravel (дата звернення: 29.03.2025).
8. GitHub Laravel Repository. URL: https://github.com/laravel/laravel (дата звернення: 29.03.2025).
9. Dev.to Laravel Blog. URL: https://dev.to/t/laravel (дата звернення: 29.03.2025).
10. SitePoint PHP Tutorials. URL: https://www.sitepoint.com/php/ (дата звернення: 29.03.2025).
11. FreeCodeCamp PHP Guide. URL: https://www.freecodecamp.org/news/tag/php/ (дата звернення: 29.03.2025).
12. Smashing Magazine - Web Development. URL: https://www.smashingmagazine.com/category/coding/ (дата звернення: 29.03.2025).

13. How to create First Laravel Project using composer!!!. URL: https://medium.com/nerd-for-tech/how-to-create-first-laravel-project-using-composer-c753d1096e32 (дата звернення: 29.03.2025).
14. Laracasts - Laravel Video Tutorials. URL: https://laracasts.com/ (дата звернення: 29.03.2025).
15. CSS-Tricks PHP Articles. URL: https://css-tricks.com/?s=php (дата звернення: 29.03.2025).
16. Make your first Laravel project. URL: https://medium.com/@iqbal.ramadhani55/make-your-first-laravel-project-7e87f9ad672f (дата звернення: 29.03.2025).
17. Laravel Developers: Stop Writing SQL or Eloquent. URL: https://medium.com/software-grognard/laravel-developers-stop-writing-sql-or-eloquent-859779701a35 (дата звернення: 29.03.2025).
18. Symfony vs Laravel Comparison. URL: https://adevait.com/laravel/laravel-vs-symfony-comparison (дата звернення: 29.03.2025).
19. CodeCanyon PHP Scripts Marketplace. URL: https://codecanyon.net/category/php-scripts (дата звернення: 29.03.2025).
20. Middleware in Laravel 12: A Step-by-Step Guide. URL: https://medium.com/@sandeeppant/middleware-in-laravel-12-a-step-by-step-guide-4d70a2395019 (дата звернення: 29.03.2025).
21. Молчанов В. П., Пандорін О. К. Технології розробки WEB-ресурсів: навчальний посібник. – Харків: ХНЕУ ім. С. Кузнеця, 2019. – 130 с.
22. В Машкіна, ВО Абрамов, ТІ Носенко, ЄВ Іваніченко.Навчально-методичний посібник до виконання і захисту кваліфікаційної роботи бакалавра спеціальностей 122 Комп'ютерні науки для здобувачів вищої освіти денної форми навчання, Київський столичний університет імені Бориса Грінченка. Київ, 2024.- 43 с.

23. Теоретичні та практичні аспекти використання математичних методів та інформаційних технологій в освіті й науці:моногр. / за заг. ред. О. Литвин. — К.: Київ. ун-т ім. Б. Грінченка,2021. — 332 с.
24. Двірничук К. В., Вацек Д. О. Веб-програмування та веб-дизайн: навчальний посібник. – Чернівці: Чернівецький національний університет ім. Ю. Федьковича, 2022. – 472 с.
25. Практичний курс Laravel: Розробка чистих MVC веб-застосунків / Daniel Correa, Paola Vallejo. – 2022. – 177 с. (Примітка: книга англійською мовою, але може бути корисною для глибшого вивчення Laravel.)
26. Серверні WEB-технології: навчальний посібник / за ред. проф. О. М. Гусак. – Київ: КПІ ім. Ігоря Сікорського, 2023. – 150 с.
27. Laravel Up & Running: A Framework for Building Modern PHP Apps. - O’Reilly Media, 2019. – 422 с.