МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

КВАЛІФІКАЦІЙНА МАГІСТЕРСЬКА РОБОТА

на тему:

«Інформаційна технологія інтеграції API СумДУ до сайтів на платформі Laravel»

Завідувач випускаючої кафедри — **Довбиш А.С.**

Керівник роботи — **Ободяк В.К.**

Студента групи ІН.м-91н — **Фесенко О.І.**

СУМИ 2021

Сумський державний університет
(назва вузу)

Факультет *ЕлІТ* Кафедра *Комп'ютерних наук*

Спеціальність *«Інформатика»*

Затверджую:

зав.кафедрою _____

_____

"_____" _____ 20___р.

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТОВІ

Фесенку Олександру Ігоровичу

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) *Інформаційна технологія інтеграції API СумДУ до сайтів на платформі Laravel*

затверджую наказом по інституту від "____" _____ 20 ___ р. № _____

2. Термін здачі студентом закінченого проєкту (роботи) _____

3. Вхідні данні до проєкту (роботи) _____
_____
_____
_____
_____

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) *1) Аналіз проблеми та постановка задачі 2) Вибір інструментів для розробки 3) Інформаційна та програмна реалізація ситеми*

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) _____
_____

6. Консультанти до проєкту (роботи), із значенням розділів проєкту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання _____

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів дипломного проєкту (роботи)	Термін виконання проєкту (роботи)	Примітка
1.	*Аналіз проблеми та постановка задачі*		
2.	*Вибір інструментів для розробки*		
3.	*Інформаційна та програмна реалізація ситеми*		
4.	*Оформлення пояснювальної записки до дипломної роботи*		

Студент – дипломник _____
(підпис)

Керівник проєкту _____
(підпис)

РЕФЕРАТ

Записка: 55 стор., 38 рис., 1 додаток, 21 джерело.

Об’єкт дослідження — процес розробки інформаційної технології інтеграції API СумДУ до сайтів на платформі Laravel.

Мета роботи — перевірка можливості інтеграції API СумДУ з сайтом на платформі Laravel.

Методи дослідження — методи інтеграції і аналізу інформаційних систем, методи розробки сайтів за допомогою фреймворків.

Результати — створений сайт типу «дошка оголошень» на платформі Laravel для вирішення різноманітних проблем у межах СумДУ. В якості інтегрованих API був задіяний список кафедр, факультетів та інститутів, а також можливість авторизації через особистий кабінет.

LARAVEL, ФРЕЙМВОРК, MYSQL, JAVASCRIPT, BOOSTRAP,
API

ЗМІСТ

Перелік умовних позначень

ПЗ – Програмне забезпечення

JS – JavaScript

API – Application Programming Interface

MVC – Model-View-Controller

MVVM – Model – View – View – Model

SQL – Structured query language

БД – База даних

DOM – Document Object Model

ВСТУП

Сумський державний університет постійно слідкує за стрімким розвитком інформаційних технологій. Відповідно, сайти підрозділів університету своєчасно оновлюються та модернізуються, а особистий кабінет постійно поповнюється новими функціями. Можуть також використовуватись сторонні наробки (сайти або програми), які спочатку могли бути розроблені для інших організацій.

З часом, більшу частину сторонніх розробок також необхідно буде оновити або модернізувати, адже необхідно слідкувати за стандартами якості. Але при цьому, можуть виникнути проблеми інтеграції вже існуючих API, що розширюють можливості сайтів та програм СумДУ, з цими сторонніми наробками.

Для розгляду можливості такої інтеграції, а також огляду можливих проблем було вирішено використати інформаційну технологію інтеграції API СумДУ до сайтів на платформі Laravel [1], а саме створити сайт типу «дошка оголошень».

Кінцевим результатом роботи стане повнофункціональний сайт з розподілом на ролі замовника та виконавця, системою додавання замовлень та подальшою інтеграцією з API СумДУ.

В кваліфікаційній магістерській роботі буде досліджена інтеграції API СумДУ зі стороннім сайтом, а також розглянуті можливі проблеми та способи їх вирішення.

1 АНАЛІЗ ІСНУЮЧИХ ПЛАТФОРМ

1.1 Огляд існуючих платформ

В інтернеті можна знайти безліч різноманітних типів сайтів та додатків і кожен з них побудований за допомогою якої-небудь платформи. У кожної з них одне й те саме призначення: допомогти з реалізацією майбутнього сайту чи додатку, але в той-же час кожна надає свій, унікальний набір функцій для роботи над проєктом.

При виборі майбутньої платформи, перш за все, слід звернути увагу призначення майбутнього продукту: сайт-візитка, інформаційний сайт, інтернет-магазин або щось інше. Також слід звернути увагу на мову програмування, що лежить в основі платформи, на наявність детальної документації, а також ознайомитися зі спектром її можливостей. Звертаючи увагу на ці критерії можна значно скоротити витрати часу на пошук та ознайомлення з майбутньою платформою. Розглянемо деякі популярні в даний момент платформи для роботи з сайтами та додатками, їх головні особливості, переваги та недоліки.

1.1.1 Symfony

На рисунку 1.1 наведено логотип платформи Symfony.

Рисунок 1.1 – Логотип Symfony

Symfony - PHP фреймворк, що в основному, використовують для фундаменту в великих проєктах, наприклад, для написання власного фреймворка або для невелеких завданнь веб-програмістів [2]. Розробка і підтримка фреймворку спонсорується французькою компанією Sensio. Symfony складається з набору не пов'язаних між собою компонентів, які можна використовувати повторно в проєктах.

Даний фреймворк дозволяє гнучко налаштовувати практично будь-яку систему під конкретні потреби, але в ньому невелика кількість того, що можна використовувати в якості готових рішень для дрібних типових задач. В интернеті можна знайти велику кількість доповнень та бібліотек розробелих для Laravel та Yii2 написаних за допомогою Symfony.

Переваги:

- постійно оновлювана та широка документація;
- безліч різних непов'язаних компонентів для повторного використання;
- механізм функціональних і модульних тестів для знаходження помилок в веб-додатку;
- прийнятність для використання у великих проєктах.

Symfony дозволяє встановлювати сторонні пакети, бібліотеки, компоненти і налаштовувати їх за допомогою конфігурації в форматах YAML, XML, PHP, а також .env файлах. Symfony пропонує систему логування помилок додатків, а також підключення бібліотеки логування Monolog. Даний фреймворк часто використовують для довгострокових складних проєктів. Він вимагає більш високих фінансових витрат, але дає перевагу при персоналізації проєкту. Нові версії Symfony пропонують тільки необхідне і дозволяють додавати тільки ті елементи, які дійсно потрібні.

Недоліки:

- високий поріг входу;
- повільніше завантаження деяких додатків;
- для тестування необхідно більше часу.

Навіть за наявності обширної документації, програмісту-початківцю, знадобиться багато часу на ознайомлення з виликою кількістю компонентів та здобуття необхідних навичок для роботи з ними. В той же час велика кількість непов'язаних компонентів може негативно сказатися на швидкодії деяких додатків, а через необхідність попередньо скомпілювати код для багаторазового використання, потрібно більше часу для тестування.

1.1.2 Django

На рисунку 1.2 наведено логотип платформи Django.

Рисунок 1.2 – Логотип Django

Django — високорівневий відкритий Python-фреймворк для розробки веб-систем [3]. Сайт на Django будується з однієї або декількох частин, які рекомендується робити модульними. Це одна з істотних архітектурних відмінностей цього фреймворку від деяких інших (наприклад Ruby on Rails).

Переваги:

- принцип «Все включено» («Batteries included»);
- стандартизована структура;
- додаток Django;
- вбудовані фреймворки для створення API.

Фраза «все включено» означає, що більшість інструментів для створення програми - частина фреймворка, а не поставляються у вигляді

окремих бібліотек. Django має широкий функціонал (ORM, міграції баз даних, аутентифікація користувача, панель адміністратора, форми) для вирішення більшості завдань веб-розробки. Слід також зазначити, що даний фреймворк був розроблений для того, щоб допомогти розробникам створювати додатки якомога швидше.

Django як фреймворк задає структуру проєкту. Вона допомагає розробникам розуміти, де і як додавати нову функціональність. Завдяки однаковою для всіх проєктів структурі набагато простіше знайти вже готові рішення або отримати допомогу від спільноти.

Додатки в Django дозволяють розділити проєкт на декілька частин. Додатки встановлюються шляхом додавання в settings.INSTALLED_APPS. Цей підхід дозволяє легко інтегрувати готові рішення. Також цей фреймворк включає в себе механізми запобігання поширених атак на зразок SQL-ін'єкцій (XSS) і підробки міжсайтових запитів (CSRF).

Django REST Framework (DRF) є бібліотекою для побудови API. Він має модульну архітектуру, яка досить легко налаштовується. Ця архітектура використовується для створення як простих, так і складних API. Великі REST API часто вимагають великої кількості запитів для отримання всіх необхідних даних. GraphQL - це мова запитів, який дозволяє обмінюватися пов'язаними даними набагато простіше. Graphene-Django дозволяє легко додавати моделі, форми, аутентифікацію та інші функціональні можливості в проєкт.

Недоліки:

- ORM Django;
- повільний розвиток;
- необхідність високого рівня володіння платформою;
- компоненти розгортуються сумісно.

Django ORM значно поступається останньої SQLAlchemy.Django ORM заснований на шаблоні Active Record, який гірше, ніж шаблон Unit of Work, який використовується в SQLAlchemy. На практиці це виражається в тому,

що в Django моделі можуть «зберігати» себе за бажанням, а транзакції відключені за замовчуванням.

Django є великим і монолітним фреймворком. Це дозволяє спільноті розробляти сотні універсальних модулів і додатків, але знижує швидкість розробки самого Django. Крім того, фреймворк повинен підтримувати зворотну сумісність, тому він розвивається відносно повільно.

1.1.3 AngularJS

На рисунку 1.3 наведено логотип платформи AngularJS.

Рисунок 1.3 – Логотип AngularJS

AngularJS — JavaScript-фреймворк з відкритим програмним кодом, який розробляє Google [4]. Він призначений для розробки односторінкових додатків, що складаються з одної HTML-сторінки з CSS і JavaScript. Його мета — розширення браузерних додатків на основі шаблонів MVC, а також спрощення їх тестування та розробки.

Переваги:

- двостороння прив’язка;
- універсальність;
- компонентна архітектура;
- модулі.

Спрощена двостороння прив’язка даних. AngularJS дозволяє прив'язувати дані до HTML за допомогою виразів, а директиви AngularJS дозволяють розробникам розширювати функціонал HTML і створювати нові конструкції. Маніпуляції з DOM і код прив'язки даних «загорнуті» в прості елементи, які можна швидко і просто вставити в HTML шаблони. Також, кожен раз, коли відбувається незначна зміна даних, дія поширюється, що означає автоматичну синхронізацію оновлень через модель даних і представлення моделі.

AngularJS спроєктований як універсальний фреймворк, тому з його допомогою можна створити веб-додаток майже будь-якого типу. Він поставляється в комплекті з рядом нових функцій, серед яких розширені RXJS, останній HTTP-клієнт, поліпшена компіляція та інше. За допомогою TypeScript стає можливе використання статичної перевірки шрифтів, високопродуктивна типізація і об'єктно-орієнтовані шаблони.

Цей фреймворк побудований на основі компонентної архітектури, яка вбудовує глибоку прив'язку в компоненти. Кожен з цих компонентів містить елементи з відповідними функціональними можливостями. Крім того, елементи в цих компонентах добре вмонтовані і слабо пов'язані. Ця характеристика допомагає зробити компоненти багаторазовими, а також допомагає підвищити результати тестування і підтримувати майбутнє обслуговування.

Модульна концепція AngularJS дозволяє спростити поділ роботи на різні команди в великих проєктах. У пріоритеті мінімальна кількість коду, тому додатки AngularJS, як правило, компактні і легкі в редагуванні.

Недоліки:

- велика кількість фінального коду;
- нав’язування «жорсткої» структури;
- проблеми з маніпуляцією DOM;
- високий поріг входу.

При використанні AngularJS у його базовому вигляді під кінець розробки можна отримати велечезні перенавантажені моделі. Це, в свою чергу, вплине на швидкість роботи проєкту, навантаження на сервер та на можливість подальшої модернізації.

Даний фреймворк має широкий набір різноманітних функцій для вирішення широкого спектру задач. Та в той же час інтеграція зі сторонніми додатками викличе багато проблем, тобто AngularJS важко назвати «гнучкою» платформою.

Розробники можуть легко створювати індивідуальні DOM, але немає ніякої гарантії, наскільки добре ці елементи DOM працюватимуть. Для обробки маніпуляцій з DOM в якому є велика кількість даних, AngularJS покладається на «брудні перевірки» для управління змінами DOM (будь-яке редагування змінних призводить до оновлення DOM).

AngularJS є доволі складним для розуміння фреймворком. Звісно використовуючи cli (інтерфейс командного рядка, що використовується для ініціалізації розробки) навіть новачок зможе прийняти участь у роботі над проєктом, але без вивчення документації складні маніпуляції викличуть значні труднощі. В той же час базова документації не завжди зможе надати відповіді на необхідні питання про роботу фреймворку.

1.1.4 Yii2

На рисунку 1.4 наведено логотип платформи Yii2.

Рисунок 1.4 – Логотип Yii2

Yii - це безкоштовний об'єктно-орієнтований компонентний full-stack PHP фреймворк [5]. В основі Yii лежить інший фреймворк - PRADO, написаний на ASP.NET і згодом перенесений на PHP. Завдяки своїй основі компонентів, архітектурі і складної підтримки кешування, фреймворк підходить для розробки великомасштабних проєктів, таких як портали, форуми, системи управління контетном (CMS), систем електронної комерції, RESTful веб-сервісів та інші.

Переваги:

- низький поріг входу;
- вбудовані рішення для інтерфейсів;
- вбудовані механізми аутентифікації користувачів;
- генератор моделей, контроллерів та GRUD.

Завдяки тому, що Yii є full-stack фреймворком, він надає безліч перевірених і готових до використання функцій. Так, наприклад, в систему вбудований Bootstrap і багато власних модулів, які з ним пов'язані. Завдяки цьому, можна створювати спливаючі модальні вікна, випадаючі списки, навігаційні меню та інше.

Yii2 має спеціальний GUI інтерфейс для вибору різних генерацій (рис. 1.5) в той час як більшість інших фрейморків має тільки консольні.

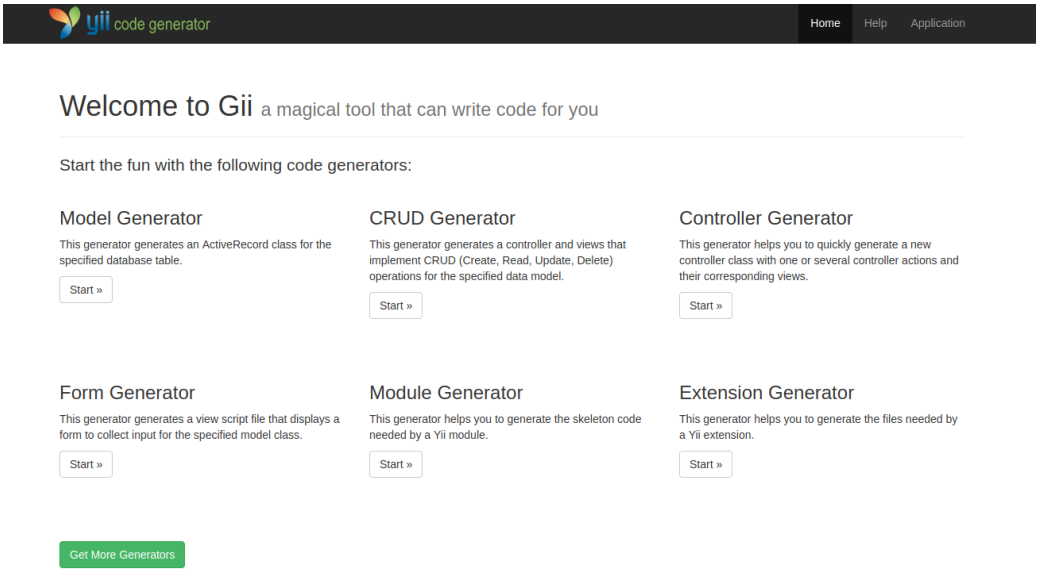


Рисунок 1.5 – Генератор Gii

CRUD генерує готові інтерфейси для управліннями даних у зазначеній йому моделі. Результатом стає готовий інтерфейс з певною адресою, в якому можна створювати нові записи таблиці, редагувати і видаляти їх. Це зручно використовувати для створення панелей адміністраторів. Сам інтерфейс використовує безліч вбудованих віджетів, які теж можна персоналізувати.

Недоліки:

- досить «жорстка» формування маршрутів;
- повільний розвиток;
- занадто великий зв’язок між frontend та backend частинами.

Yii має доволі жосткі обмеження (у порівняння з іншими фремворками) при формування маршрутів. Це стосується як шляху до контроллера так і додавання своїх правил. Також, для групування маршрутів, необхідно створювати окремий клас.

Безліч готових рішень для інтерфейсу, відразу роблять генерацію в уявленнях (views). Головною проблемою є те, що вони додають сценарії прямо в тіло сторінки, а в їх коді перемішаний php і html код. Такий код досить проблематично підтримувати, а також можуть виникнути труднощі при роботі зі сторонніми додатками.

1.1.5 Висновок за результатами аналізу

Підсумовуючи вищесказане, можна зробити висновок, що кожен з існуючих на даний момент фреймворків має як свої переваги так і свої недоліки. Обирати фреймворк для роботи необхідно опираючись на ряд простих і в той-же час важливих критеріїв. Серед таких критеріїв можна виділити:

- простоту в освоєнні;
- гнучкість налаштування;
- область застосування
- набір базових функцій.

Звертаючи увагу на такі критерії, а також підсумувавши розглянуті відмінності існуючих платформ, для роботи над даним проєктом було вирішено використовувати платформу Laravel. Окрім особливостей розглянутих у пункті 3.2, даний фреймфорк був обраний за його відносну простоту освоєння, виразний синтаксис та оформлення коду а також його популярність, завдяки якій вирішення деяких проблем потребувало значно менше часу.

1.2 Постановка задачі

Після аналізу існуючих альтернатив платформ розробки, для дослідження інформаційної технології інтеграції API СумДУ до сайтів на платформі Laravel, були поставлені такі задачі:

1) провести поглиблений аналіз обраної платформи розробки;
2) спроєктувати та розробити серверну частину;
3) спроєктувати та розробити клієнтську частину;
4) інтегрувати API СумДУ.

2 АНАЛІЗ МОЖЛИВОСТЕЙ API. МОДЕЛЬ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ

В наш час інформаційні технології стрімко розвиваються та починають охоплювати все більше сфер життєдіяльності людини. Яскравим прикладом є широке різноманіття різноспрямованих сайтів та платформ для їх розробки. Для того, щоб користувався популярністю та не був загублений серед конкурентів його необхідно своєчасно та швидко оновлювати та модернізувати. Для економії часу на такі дії все частіше розробники використовують різноманітні API.

2.1 Application Programming Interface

Основним завданням створення API було надати можливість програмістам істотно полегшити задачу при розробці різних додатків за рахунок використання вже готового коду (будь-якої стандартної функції, процедури, структури або постійного значення, які будуть в подальшому виконуватися в кінцевому продукті) [6]. API являє собою сукупність різних інструментів, функцій, реалізованих у вигляді інтерфейсу для створення нових додатків, завдяки якому одна програма буде взаємодіяти з іншого.

Завдяки API різні компоненти програми отримують можливість взаємодії один з одним, формуючи при цьому певну систему ранжирування, де абсолютно всі компоненти, які вище за рангом використовують інформацію з більш низьких за рангом компонентів. Також API може визначати функціональність, яку певна програма в формі бібліотеки або модуля зможе здійснювати, при цьому API дозволяє абстрагуватися від способу реалізації функціоналу. Приклади використання:

- процес швидкої реєстрації в додатку або на сайті використовуючи вже існуючий аккаунт у іншій соціальній мережі;
- інтеграція інформації із одних веб-сервісів до інших;
- отримання та використання метеорологічних або схожих даних.

Незважаючи на всі переваги, не слід забувати про деякі недоліки API, що можуть істотно вплинути на майбутній проєкт. Відмінності в API різних операційних систем істотно ускладнюють перенесення додатків між платформами. Також можуть виникнути проблеми при використанні різних API, що дозволяють отримати однаковий результат. Такі API зазвичай реалізовані з використанням програмних компонентів більш низького рівня абстракції. Це, в свою чергу, призведе до втрати функціональності при переході з більш низького рівня на більш високий.

Підсумовуючи все вищесказане можна зазначити, що API – це потужний інструмент для полегшення розробки або розширення функціоналу додатку або сайту.

2.2 Інформаційна технологія

Реалізація даної інформаційної технології буде полягати у створенні сайту типу «дошка оголошень» на платформі Laravel. Перш за все, необхідно зазначити, що розробляємий сайт буде оперувати великою кількістю даних. Це означає, що на відміну від одно-сторінкових сайтів, також необхідно реалізувати взаємодію клієнтської та серверної частин з базою даних.

Клієнтська частина – це те, що бачить користувач. При виконанні певних дій користувачем, з клієнтської частини відсилатимуться запити на серверну, яка в свою чергу даватиме відповідну реакцію [7].

В роботі буде реалізовано ряд функцій:

- авторизація та реєстрація користувачів;
- розподіл на ролі замовник або виконавець;
- пошук та фільтрація замовлень;
- простий чат;
- профіль користувача;
- можливість модерації контенту сайту.

Головною сторінкою сайту планується сторінка зі списком замовлень. Елементи даного списку представлятимуть собою назву, скорочений опис,

приблизні терміни виконання та оплату замовлення. Неавторизований користувач зможе переглядати лише ці дані.

Авторизація буде здійснюватись через введення логіну та паролю, якщо користувач незареєстрований, він матиме змогу відправити заявку на реєстрацію заповнивши відповідну форму.

Авторизовані користувачі матимуть змогу переглядати більш детальну (повне описання замовлення, прикріплені файли, інформація про замовника та інше) інформацію про замовлення. У ролі замовника користувач матиме змогу створити нове замовлення заповнивши спеціальну форму на головній сторінці, а також редагувати або видалити вже існуюче. У ролі виконавця він матиме змогу залишити свою пропозицію до обраного замовлення або відредагувати її. Також користувачі матимуть змогу зв’язуватись між собою за допомогою простого чату.

Також на головній сторінці буде реалізований гнучкий пошук та фільтрації створених замовлень. Серед критеріїв пошуку будуть: пошук по назві, пошук за категоріями, фільтрація за датою створення та інше.

У своєму профілі користувачі зможуть змінювати інформацію про себе, змінювати пароль, а також, в залежності від ролі, матимуть змогу переглядати свої замовлення або пропозиції.

При вході за спеціальним логіном та паролем адміністратор матиме змогу видаляти некоректні замовлення, приймати заявки на реєстрацію та блокувати/розблоковувати користувачів у разі порушення правил сайту. Такі користувачі отримають відповідне повідомлення при вході на сайт.

3 ЗАСОБИ РЕАЛІЗАЦІЇ

Розробка будь-якого програмного забезпечення починається з вибору засобів реалізації майбутнього проєкту. При виборі слід звернути увагу на простоту освоєння, зручність використання та спектр охоплюваних можливостей.

З урахуванням переліченого, в якості редактор коду був обраний Visual Studio Code, який має зручний інтерфейс та можливість підключення різноманітних розширень, які можуть спрости розробку проєктів.

Для розробки серверної частини використовувався фрейворк Laravel. В основі даного фреймворку лежить мова програмування PHP, одна з найпопулярніших мов для генерації html-сторінок на стороні веб-сервера. Для роботи з великою кількістю даних майбутнього проєкту була обрана вільна система реляційними керування базами даних MySQL, адже вона надає широкі можливості для роботи з динамічними веб-сторінками.

Клієнтська частина виконувалась за допомогою Bootstrap, що являє собою набір інструментів для створення веб-сайтів, який містить шаблони CSS та HTML та мови програмування JavaScript, однієї з найпопулярніших мов для сценаріїв веб-сторінок.

3.1 Редактор коду

Visual Studio Code - безкоштовний редактор коду, розроблений Microsoft для Windows, Linux і macOS [8]. На даний момент, один із найпопулярніших редакторів для кроссплатформенної розробки веб-додатків і програм для хмарних систем. VSCode включає в себе відладчик, інструменти для роботи з Git, виділення синтаксису, засоби для рефакторинга, автодоповнення типових конструкцій і контекстних підказок (рис. 3.1).

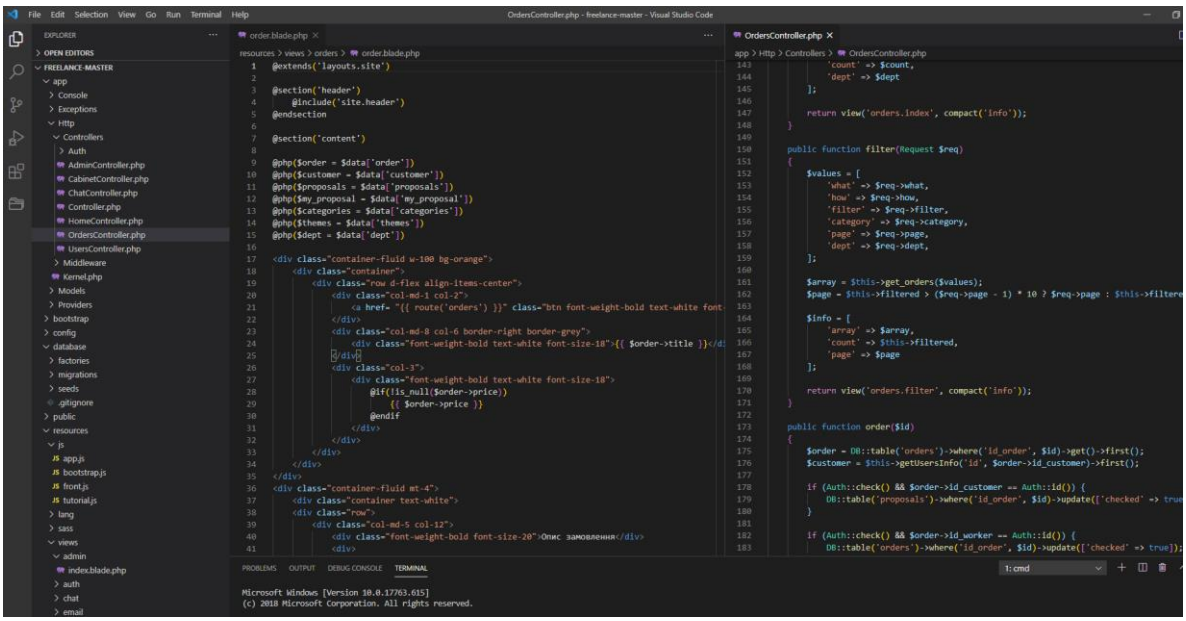


Рисунок 3.1 – Інтерфейс Visual Studio Code

Редактор надає можливість назначати власні поєднання клавіш і файли конфігурації, також має широкі можливості для кастомізації та широкий вибір розширень, що можуть значно спрости роботу над проєктом. Під час розробки були використані такі розширення:

- Prettier – полегшує читабельність коду в редакторі, виконує просте форматування коду при зберіганні;
- GitLens – полегшує навігацію та роботу з репозиторієм на GitHub за допомогою команд порівняння, відображення історії змін та інше;
- Laravel Blade Snippets – містить готові сніпети та команди, які використовуються в Laravel.

3.2 Laravel

Laravel — це PHP-фреймворк загального призначення з відкритим кодом, що використовується при розробці веб-додатків відповідно до шаблону MVC (Model-View-Controller).

Laravel часто порівнюють з цілою екосистемою, це зумовлено тим, що для фреймворку постійно розроблювались нові інструменти для вирішення різноманітних задач – від невеликих додатків для шеринга і коментування

фото до великих ПЗ, які використовують величезні компанії. Додатки на Laravel забезпечують більш високу продуктивність у порівнянні з додатками, створеними за допомогою інших фреймворків. Це можливо в тому числі завдяки системі кешування. Драйвер файлового кешування зберігає безліч елементів в файлової системі. Це дозволяє швидко розробляти програми. Також серед особливостей виділяють:

- зручне масштабування додатків;
- вбудоване тестування компонентів додатку;
- висока базова безпека додатків (наприклад захист від SQL–ін’єкцій або міжсайтових підробок запитів);
- шаблони , що надають можливість багаторазово використовувати один і той же шаблон в різних ділянках програми;
- консоль Artisan, в арсеналі команд якої є робота з міграціями, контроллерами і моделями, авторизацією і іншими базовими компонентами фреймворка.

Не менш важливою особливістю Laravel для даного проєкт є використання Model-View-Controller (MVC) архітектури (рис. 3.2).

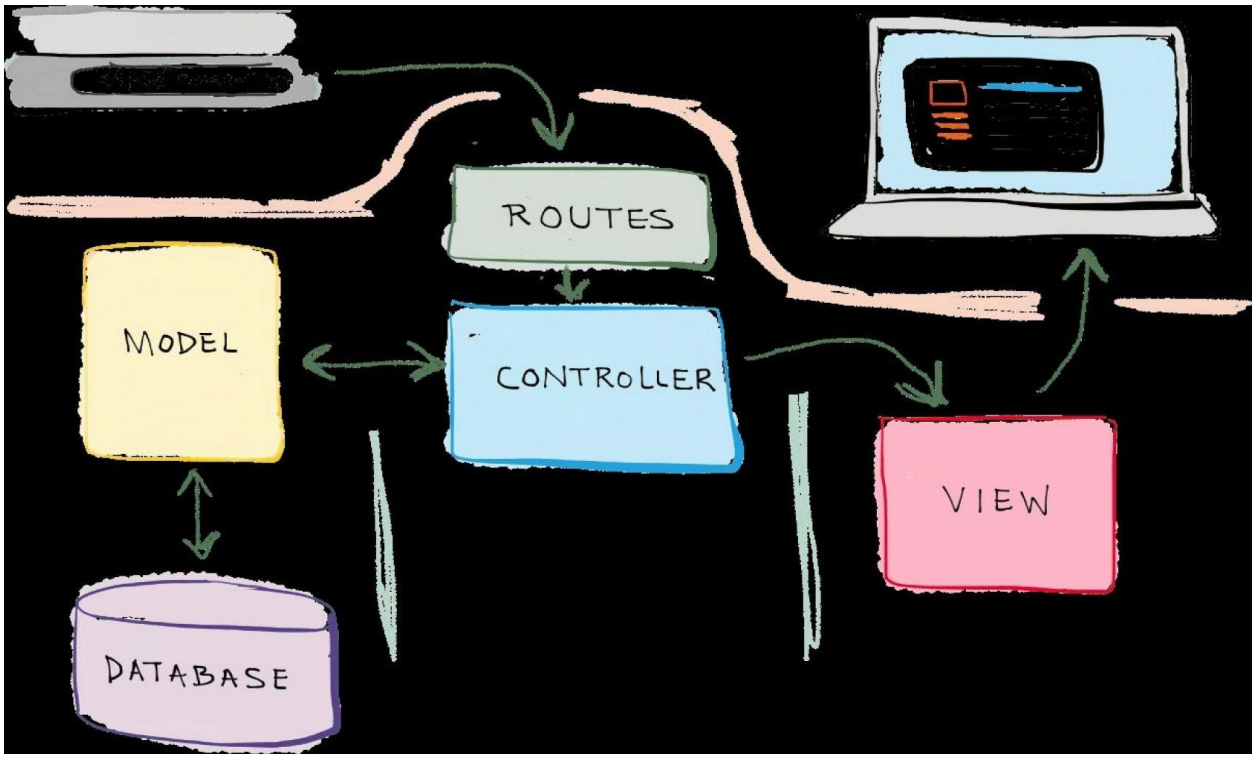


Рисунок 3.2 – Структура MVC [9]

Завдяки даній архітектурі, досягається чіткий поділ між трьома абстрактними частинами програми:

- моделлю;
- контроллерами;
- поданням.

Вони стають незалежними один від одного і можуть бути використані окремо. Це допомагає уникнути ситуацій, коли виправлення одних помилок в логіці пошкоджують старі напрацювання і призводять до ще більшого числа помилок в багатьох місцях. Будь-якій людині важко враховувати всі зв'язки і передбачити, в яких місцях і на що його новий код може негативно вплинути. Тому, єдино правильне рішення - позбутися від цих зв'язків.

3.3 MySQL

MySQL - вільна реляційна система управління базами даних [10]. Розробка та підтримка сайта MySQL здійснює корпорація Oracle, продукт розповсюджується як під GNU General Public License, так і під власною комерційною ліцензією. Саме завдяки такому замовленню майже в більш ранніх версіях з'явився механізм реплікації.

Зазвичай MySQL використовується як сервер, до якого звертаються локальні або видалені клієнти, проте до дистрибутиву входить бібліотека внутрішнього сервера, що дозволяє включати MySQL в автономні програми.

Гнучкість СУБД MySQL забезпечується підтримкою великої кількості типів таблиць: користувачі можуть вибрати як таблиці типу MyISAM, що підтримують повнотекстовий пошук, так і таблиці InnoDB, що підтримують транзакції на рівні окремих записів. Більш того, СУБД MySQL поставляється із спеціальним типом таблиць EXAMPLE, що демонструє принципи створення нових типів таблиць. Завдяки відкритій архітектурі і GPL-ліцензуванню, в СУБД MySQL постійно з'являються нові типи таблиць. Також важливою особливістю є механізм реплікації. Він дозволяє копіювати

дані з одного джерела в інше (або в безліч інших) і навпаки. При реплікації зміни, зроблені в одній копії об'єкта, можуть бути поширені на інші копії.

Для безпосередньої роботи з базою даних проєкту використовувався веб-додаток phpMyAdmin [11].

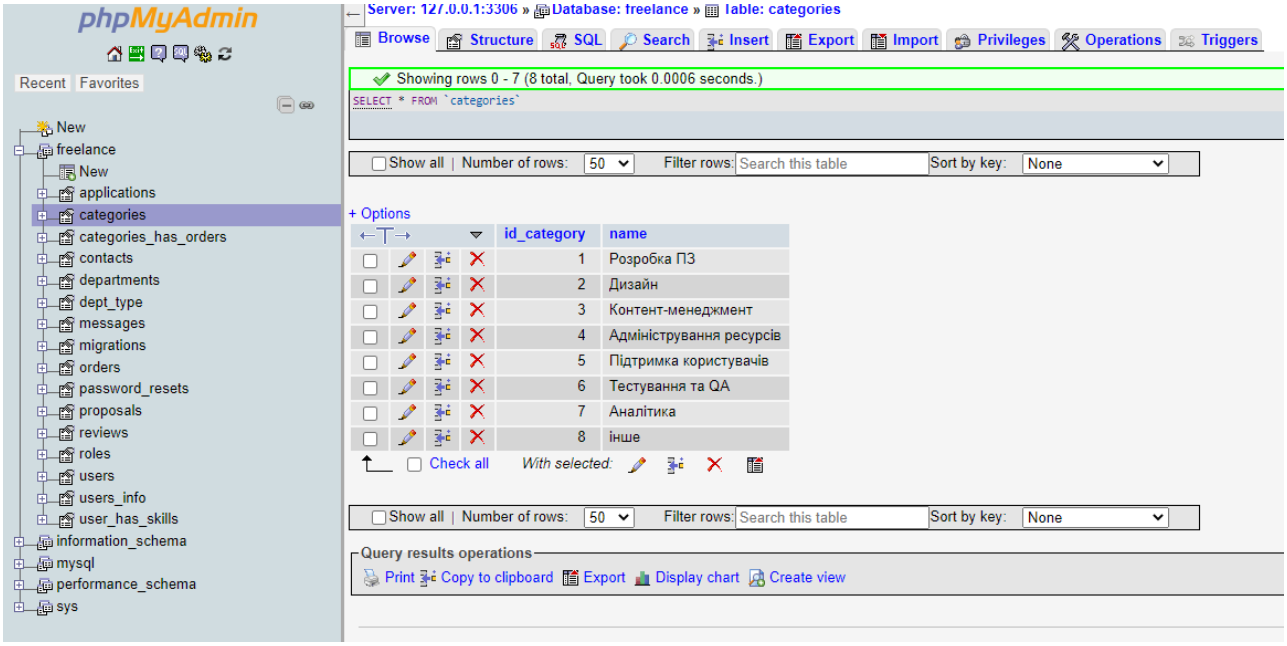


Рисунок 3.3 – Інтерфейс phpMyAdmin

Даний додаток написаний на мові PHP і представляє собою веб-інтерфейс для адміністрування СУБД MySQL. PhpMyAdmin дозволяє через браузер здійснювати адміністрування сервера MySQL, запускати команди SQL і переглядати вміст таблиць і баз даних. Головною перевагою додатку є можливість керувати СУБД MySQL без безпосереднього введення SQL команд.

3.4 Засоби реалізації клієнтської частини

Bootstrap - це відкритий і безкоштовний HTML, CSS і JS фреймворк, який використовується для швидкої верстки адаптивних дизайнів сайтів та веб-додатків [12].

Даний фреймворк включає в себе HTML- і CSS-шаблони оформлення веб-форм, кнопок, міток, блоків навігації та інших компонентів веб-інтерфейсу, включаючи JavaScript-розширення. Після підключення Bootstrap

до проєкту, розробнику стане доступна велика кількість класів і готових компонентів. Використовуючи їх можна дуже швидко і якісно створити сучасний адаптивний дизайн сайту. Класи Bootstrap можна розбити на 3 великі групи:

- класи для створення сітки (адаптивного макета сторінки);
- класи для стилізації контенту (тексту, коду, зображень, таблиць та іншої інформації);
- службові класи для вирішення найбільш часто зустрічаються допоміжних завдань (вирівнювання, управління відображенням, додавання кордонів та ін.).

Крім класів у фреймворку також є компоненти (готові об'єкти інтерфейсу). Це кнопки, форми, навігаційні меню, випадаючі списки, спливаючі панелі та інші. Також використання bootstrap значно полегшує адаптацію будь-якого сайту для коректного відображення на різноманітних пристроях.

Також, окрім переваг у Bootstrap є також недоліки. Основний полягає в тому, що він не підходить для проєктів з унікальним дизайном або елементами. В цьому випадку з Bootstrap для проєкту в основному беруть тільки сітку. Тому для даного проєкту також використовувалась сценарна мова програмування JavaScript.

JavaScript зазвичай використовується як інтегрована мова для програмного доступу до об'єктів додатку [13]. Найбільш широке застосування знаходить в браузерах як мова сценаріїв для додання інтерактивності веб-сторінкам. Серед архітектурних особливостей можна виділити:

- динамічна типізація;
- слабка типізація;
- автоматичне керування пам'яттю;
- прототипне програмування;
- функції як об'єкти першого класу.

Також дана мова є об'єктно-орієнтованою, але використання прототипного програмування [14] обумовлює відмінності в роботі з об'єктами в порівнянні з традиційними клас-орієнтованими мовами. Крім того, JavaScript має ряд властивостей, притаманних функціональним мовам, що робить мову більш гнучкою.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ

Проаналізувавши поставлені задачі та обравши засобі реалізації, почнемо розробку сайту. Розробка починається зі створення структури сайту, так званого «скелету», на який у подальшому, будуть «навішуватись» інші елементи проєкту.

4.1 База даних

Laravel має простий метод заповнення бази даних тестовими даними за допомогою класів-наповнювачів (seed classes) [15]. Ці класи зберігаються в database / seeds. Можна використовувати будь-яке ім'я для назви класу-наповнювача, але зазвичай використовують імена подібні до UsersTableSeeder. За замовчуванням клас DatabaseSeeder вже створено в папці наповнювачів. В цьому класі можна використовувати метод call для запуску інших наповнювачів, що дозволяє контролювати порядок наповнення.

Для створення наповнювача використовувалась базова команда: «php artisan make: seeder UsersTableSeeder». Команда створює наповнювач в папці database / seeds. В даному проєкті, у класі DatabaseSeeder, метод «call» використовувався для запуску додаткових класів-наповнювачів (рис. 4.1). Це робилось для того, щоб не створювати один великий клас-наповнювач.

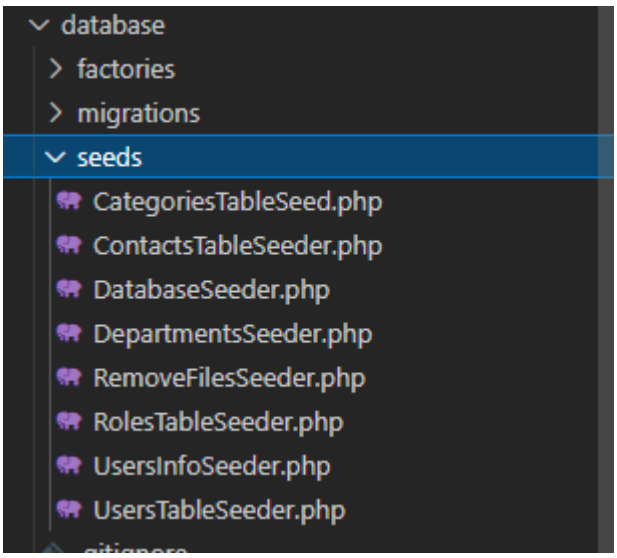


Рисунок 4.1 – Список створених класів-наповнювачів

За замовчуванням клас-наповнювач містить тільки один метод: run. Метод викликається, коли запускається команда «db: seed» Artisan command. У методі «run» в БД вставляються будь-які тестові дані будь-яким зручним способом (рис. 4.2).

database > seeds > CategoriesTableSeed.php

```php
<?php

use Illuminate\Database\Seeder;

class CategoriesTableSeed extends Seeder
{
    /**
     * Run the database seeds.
     *
     * @return void
     */
    public function run()
    {
        $values = [
            ['name' => 'Розробка ПЗ'],
            ['name' => 'Дизайн'],
            ['name' => 'Контент-менеджмент'],
            ['name' => 'Адміністрування ресурсів'],
            ['name' => 'Підтримка користувачів'],
            ['name' => 'Тестування та QA'],
            ['name' => 'Аналітика'],
            ['name' => 'інше'],
        ];

        DB::table('categories')->insert($values);
    }
}

```

Рисунок 4.2 – Приклад класу-наповнювача

Після створення класів-наповнювачів, використовуючи Artisan команду «php artisan db: seed», виконувалось заповнення БД. За умовчанням вона виконає клас DatabaseSeeder, який може бути використаний для виклику інших класів-наповнювачів. Також для заповнення БД використовувалась команда «php artisan migrate: refresh --seed», яка робить відкат змін і застосує всі міграції (рис. 4.3) заново.

```php
<?php

use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class Categories extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('categories', function (Blueprint $t
            $table->bigIncrements('id_category');
            $table->string('name', 45);
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('categories');
    }
}
```

Рисунок 4.3 – Приклад файлу міграції

Ця команда корисна для повного перестроювання БД і часто використовувалась при подальшому тестуванні взаємодії елементів проєкту.

4.2 Серверна частина

В даному пункті розглянуто роботу та взаємодію частин проєкту, що відповідають за серверну частину сайту, а саме: маршрутизація, контроллери та посередники (англ. middleware).

4.2.1 Маршрутизація

У Laravel є дуже простий і виразний метод визначення маршрутів, найпростіші маршрути обробляють URI (шлях) і функцію-замикання [16]. Всі маршрути (routes) Laravel визначені в файлах маршрутів, які розташовані в каталозі routes/. Ці файли автоматично завантажуються фреймворком. У файлі routes/web.php визначені маршрути для web-інтерфейсу. Ці маршрути

входять в групу посередників web, які забезпечують такі можливості, як стан сесії і CSRF-захист. Маршрути з файлу routes/api.php не підтримують стану і входять до групи посередників api. Маршрутизатор дозволяє реєструвати маршрути для будь-якого HTTP-запиту:

- Route::get($uri, $callback);
- Route::post($uri, $callback);
- Route::put($uri, $callback);
- Route::patch($uri, $callback);
- Route::delete($uri, $callback);
- Route::options($uri, $callback).

В даному проєкті використовуються маршрути для HTTP–запитів GET та POST, з пряв’язкою до дій відповідних контроллерів (рис. 4.4):

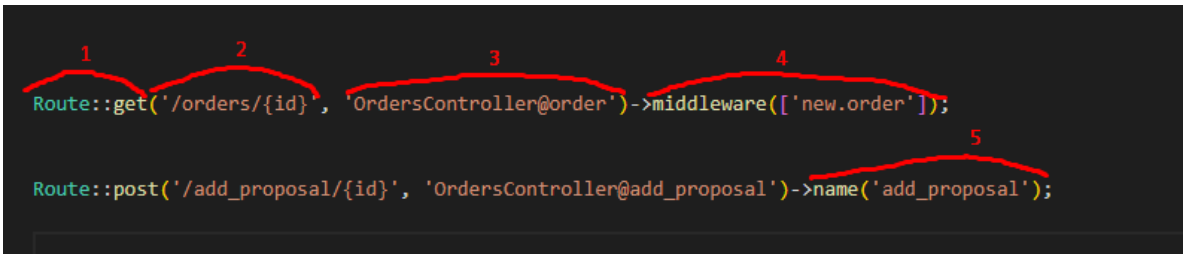


Рисунок 4.4 - Приклад машршутів проєкту

1) отримання або відправка відповідних даних;
2) URL за яким відбувається відповідна функція;
3) функція (order) відповідного контроллеру (OrdersController);
4) використання відповідного посередника за його ключем;
5) ім’я для подальшого використання в коді.

4.2.2 Контроллери

Замість того, щоб визначати всю логіку обробки запитів у вигляді замикань в файлах маршрутизації, Laravel надає змогу організувати її за допомогою класів контролерів [17]. Контролери можуть групувати пов'язану з обробкою HTTP-запитів логіку в окремий клас. Вони зберігаються в папці app/Http/Controllers.

В даному проєкті використовується декілька контроллерів (рис. 4.5) і кожен з них включає в себе набір функцій для роботи з серверною частиною функціоналу сайту.

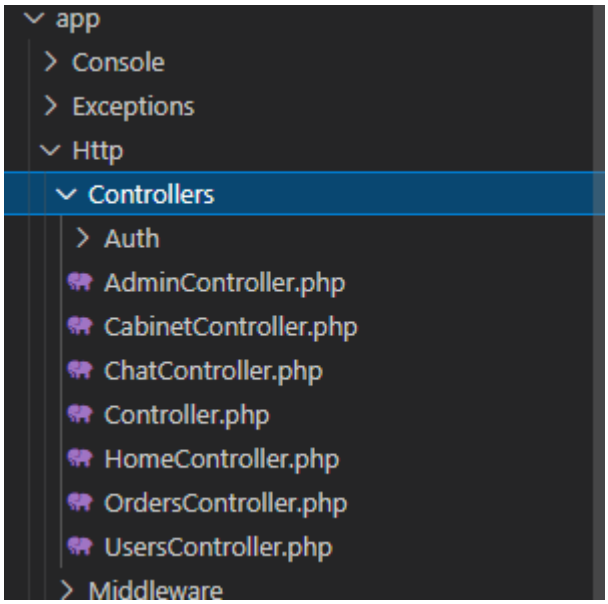


Рисунок 4.5 – Список контроллерів проєкту

4.2.3 Middleware

Посередники в Laravel надають зручний механізм для фільтрації HTTP-запитів [18]. Наприклад, один з базових посередників відповідає за перевірку аутентифікації користувача: якщо користувач не аутентифікований, його перенаправить на екран входу в систему; якщо ж користувач аутентифікований, посередник дозволить пройти далі. Також є базові посередники для додавання особливих заголовків до всіх відповідей додатку (CORS-посередник) або ведення логів вхідних даних. Для створення посередників використовується команда Artisan «php artisan make:middleware {name}». Всі посередники розташовані в каталозі app/Http/Middleware.

Для того, щоб посередники зупискались для кожного HTTP-запиту їх необхідно добавити їх до методу «$middleware» класу app/Http/Kernel.php. Для використення посередників разом з необхідними маршрутами, їм необхідно надати спеціальний ключ у методі «$routeMiddleware» класу Kernel.php (рис. 4.6).

```
protected $routeMiddleware = [
    'auth' => \App\Http\Middleware\Authenticate::class,
    'auth.basic' => \Illuminate\Auth\Middleware\AuthenticateWithBasicAuth::class,
    'bindings' => \Illuminate\Routing\Middleware\SubstituteBindings::class,
    'cache.headers' => \Illuminate\Http\Middleware\SetCacheHeaders::class,
    'can' => \Illuminate\Auth\Middleware\Authorize::class,
    'guest' => \App\Http\Middleware\RedirectIfAuthenticated::class,
    'signed' => \Illuminate\Routing\Middleware\ValidateSignature::class,
    'throttle' => \Illuminate\Routing\Middleware\ThrottleRequests::class,
    'verified' => \Illuminate\Auth\Middleware\EnsureEmailIsVerified::class,
    'is.admin' => \App\Http\Middleware\AdminCheck::class,
    'is.customer' => \App\Http\Middleware\isCustomer::class,
    'logged.in' => \App\Http\Middleware\LoginCheck::class,
    'new.order' => \App\Http\Middleware\newOrder::class,
];
```

Рисунок 4.6 – Ключі посередників проєкту

Посередники, які часто використовуються з маршрутами веб-UI та API, доцільно об’єднувати в групи зі спільним ключем. Це виконується у методі «$middlewareGroups». Також можна надавати додаткові параметри. Наприклад, якщо необхідна перевірка того, чи є у аутентифікованого користувача певна «роль». Додаткові параметри будуть передаватися в посередник після аргументу «$next».

4.3 Клієнтська частина

В даному пункті розглянуто роботу над клієнтською частиною сайту, рішення прийняті в ході роботи, використання обраних засобів реалізації, їх взаємодію між собою та з серверною частиною.

4.3.1 Шаблони Blade

Фреймворк Laravel надає простий, але потужний шаблонізатор Blade [19]. На відміну від інших популярних шаблонізаторів для PHP Blade не обмежує розробників у використанні чистого PHP-коду. Всі уявлення Blade скомпільовані в чистий PHP-код і кешовані, поки в них немає змін, тобто він практично не навантажує програму. Файли уявлень Blade використовують розширення .blade.php і зберігаються в каталозі resources/views/.

Дві головні переваги використання Blade (рис. 4.7) - успадкування шаблонів і секції.

```
@extends('layouts.site')

@section('header')
    @include('site.header')
@endsection

@section('content')

@php($data = $info['data'])
@php($categories = $info['categories'])
@php($dept = $info['dept'])

<div class="d-none" id="my_id" data-id="{{ Auth::id() }}"></div>
<div class="container-fluid">···
</div>

<div class="flash-message fixed-bottom text-center">···
</div>

@endsection

@section('footer')
    @include('site.footer')
@endsection
```

Рисунок 4.7 – Приклад шаблону Blade

Для вибору шаблону, який повинен бути «успадкований» дочірнім шаблоном, використовується директива @extends. Blade-директива @include дозволяє включати одне уявлення в інше, включаючи всі змінні та параметри. Директива @section/@endsection визначає секцію вмісту, а за допомогою директиви @yield можна відобразити вміст даної секції в іншому шаблоні.

4.3.2 Верстка з використанням Bootstrap

Для початку роботи з Bootstrap, його необхідно встановити. На офіційному сайті [12] представлено декілька способів встановлення: завантаження скомпільованого або повного коду, використовуючи додаток jsDelivr або за допомогою менеджеру пакетів (npm, Composer, RubyGems та інші).

В даному проєкті використовується власний дизайн розроблений з урахуванням базових принципів bootstrap та беручи за основу його сітку. Такий підхід значно спростив наповнення сайту елементами та їх подальшу компоновку.

Стильові класи bootstrap дозволяють значно спростити роботу над типовими елементами (кнопки, форми, випадаючі меню та інші) сторінки сайту (рис. 4.8).

```
<div class="container collapse bg-light-grey text-white my-2" id="new-order">
  <form method="POST" action="{{ route('save_order') }}" enctype="multipart/form-data">
    @csrf
    <div class="row">
      <div class="col-md-6 col-12 mt-4">
        <p class="font-size-35 font-weight-bold bg-orange text-center">Створення замовлення</p>
      </div>
    </div>
    <div class="d-flex row justify-content-around">
      <div class="form-group col-md-5 col-12">
        <label for="title" class="font-size-20">Назва*</label>
        <input type="text" class="form-control text-white border-0 bg-deep-dark" id="title" name="title"
          required>
        <label for="description" class="font-size-20 mt-2">Інформація*</label>
        <textarea class="form-control text-white border-0 bg-deep-dark" name="description" id="description"
          rows="5" required></textarea>
        <input id="add-files" type="file" class="btn badge-pill bg-white mt-2 d-none" multiple="multiple"
          name="files[]">
      </div>
```

Рисунок 4.8 – Приклад коду сторінки з використанням класів bootstrap

Кожен сучасний сайт повинен бути в змозі адаптуватися під різноманітні пристрої. Під адаптивністю сайту розуміють його здатність коректно відображати контент своїх сторінок не зважаючи на роздільну здатність екрану пристрою користувача. В даному аспекті сітка bootstrap надає одну з найкомфортніших систем контрольних точок та пов’язаних з ними класів для компоновки елементів.

4.3.3 Сценарні вирази

Незважаючи на різноманітність готових рішень що надає bootstrap, деякі елементи сайту (Додаток) необхідно було зробити власноруч. В цьому допомогла мова сценарних виразів JavaScript та її бібліотека jQuery.

jQuery - набір функцій JavaScript, що фокусується на взаємодії JavaScript і HTML [20]. Бібліотека jQuery дозволяє легко отримати доступ до будь-якого елементу DOM, звертатися до атрибутів та вмісту елементів

DOM, а також маніпулювати ними. Також бібліотека jQuery надає зручний API для роботи з AJAX. В даному проєкті, за допомогою JavaScript були розроблені:

- вікна підтверджень та попереджень;
- обмін повідомленнями між користувачами;
- заміна класів деяких елементів.
- навігація між сторінками.

4.4 Використання API СумДУ

Для інтеграції до розробленого сайту були обрані API для можливості авторизації через особистий кабінет та список кафедр та інститутів СумДУ.

Можливість входу на сайт використовуючи дані іншої соціальної мережі реалізована на багатьох сайтах, це зумовлено можливістю зекономити час необхідний на реєстрацію користувача. В даному проєкті, на сторінці входу, була додана кнопка «Вхід Ч/З кабінет». До даної кнопки прив’язано виконання спеціальної події, а саме, звернення до маршрутів. Через маршрути «Route::post('/cabinet-login', 'CabinetController@cabinetLogin')->middleware('guest')->name('cabinet-login');» та «Route::get('/cabinet-request', 'CabinetController@cabinetRequest')->name('cabinet-request');» виконується звернення контроллнеру CabinetController.php (Додаток), що відповідає за обробку даних між сайтом та особистим кабінетом.

```php
public function cabinetRequest(Request $req) {
    $this->key = !empty($req['key']) ? $req['key'] : "";
    $this->mode = !empty($req['mode']) ? $req['mode'] : 0;
    session(['key' => $this->key]);
    $this->token = $this->key;
    if (!empty($this->key)) {
        switch ($this->mode) {
            case 0:
                break;
            case 2:
                return response(File::get(public_path($this->icon)), 200)->header('Content-Type', 'image/png');
            case 3:
                return response($this->info, 200)->header('Content-Type', 'text/plain');
            case 100:
                return response('', 200)->header('X-Cabinet-Support', $this->mask);
            default:
                exit;
        }
    }
}
```

Рисунок 4.9 – Функція cabinetRequest

Функція cabinetRequest (рис. 4.9) відповідає за надання Особистому кабінету СумДУ інформації (логотип, адреса, ліцензія та інше) про розробляємий ресурс в залежності від «case».

```php
public function cabinetLogin(Request $req)
{
    $this->key = session('key');
    $response = json_decode(file_get_contents($this->cabinet_api . 'getPerson?key=' . $this->key));
    if($this->key) {
        $person = $response->result;
        $fresh = false;
        if (!User::where('guid', $person->guid)->exists()) {
            if (User::where('email', $person->email)->exists()) {
                DB::table('users')->where('email', $person->email)->insert('guid', $person->guid);
            }
            else {
                $data = [
                    'id_role' => 'Замовник',
                    'email' => $person->email,
                    'name' => $person->name,
                    'surname' => $person->surname,
                    'guid' => $person->guid
                ];
                $new = new RegisterController();
                $new::create($data);
                $fresh = true;
            }
        }
        $user = User::where('email', $person->email);
        $uid = $user->get('id')->first()->id;
        Auth::loginUsingId($uid, true);
        if(Auth::check())
```

Рисунок 4.10 – Функція cabinetLogin

Функція cabinetLogin (рис. 4.10) відповідає за обробку інформації, що надається Особистим кабінетом на запис ресурсу. Перш за все перевіряється чи існує в базі даних користувач, що намагається увійти. Якщо ні, у відповідній таблиці бази створюється новий запис і до нього заносяться дані (ім’я, прізвище, email тощо) отримані від Особистого кабінету. Після цього, виконується перевірка ролі користувача, у разі її відсутності, система запропонує користувачеві обрати роль.

Дана функція дозволить користувачеві, у якого є доступ до особистого кабінету СумДУ, одразу зареєструватися на сайті та автоматично заповнити деякі поля профілю. Слід зазначити, що для реєстрації без доступу до особистого кабінету СумДУ, користувачеві доведеться заповнити відповідну форму, а потім дочекатися її прийняття адміністратором.

Сумський державний університет має доволі велику кількість кафедр, управлінь, департаментів та інших. Власноруч створити такий список з усіма найменуваннями дуже складно, а також подальше його оновлення було-б непростою задачею, адже кафедри та департаменти можуть змінювати свою назву, об’єднуватись з іншими або розпадатись. Саме тому для сайту був інтегрований вже готовий та постійно оновлюємий список кафедр та інститутів.

Даний API вбудовується в базу даних сайту, для зручності використання список був поділений на дві пов’язані таблиці: «departaments» та «categories».

```php
public function settings()
{
    if (Auth::user()->id_role == 1) {           return redirect('/admin');       }
    $types = DB::table('dept_type')->get();
    $dept = [];
    foreach ($types as $one) {          $dept[$one->type_name] = DB::table('departments')->where('id_type', $one->id_type)->get()->toArray();       }
    $categories = DB::table('categories')->get();
    $skills = DB::table('user_has_skills')->where('id', Auth::id())->get();
    $string = '';
    foreach ($skills as $one) {          $string .= $one->id_category . '|';       }
    $my_dept = DB::table('users')->where('id', Auth::id())->get()->first();
    $my_dept = $my_dept->id_dept;
    $data = [
        'dept' => $dept,
        'categories' => $categories,
        'string' => $string,
        'types' => $types,
        'my_dept' => $my_dept,
    ];
    return view('users.settings', compact('data'));
}
public function save_settings(Request $req)
{
    DB::table('users')->where('id', Auth::id())->update(['id_dept' => $req->id_dept == 0 ? null : $req->id_dept]);
    $req->session()->flash('alert-success', 'Кафедру користувача успішно оновлено!');
    return back();
}
```

Рисунок 4.11 – Функції settings та save_settings

Для сторінки налаштувань замовника, використовуються функції «settings» та «save_settings» (рис. 4.11). Вони необхідні для отримання з бази списку кефедр та оновлення відповідного параметру користувача у таблиці «users». За схожим принципом виконується взаємодія на панелі адміністратора та на сторінці реєстрації. За обробку даних цих сторінок відповідає AdminController.php. В подальшому, за допомогою даного списку, планується розширити можливості пошуку на сайті.

5 ІНСТРУКЦІЯ ПО РОБОТІ

На головній сторінці розміщений список замовлень пошук та фільтрація. Не авторизований користувач зможе лише переглядати скорочену інформацію про замовлення. Для входу на сайт необхідно натиснути відповідну кнопку «Вхід». Також при першому відвідування сайту користувача буде перенаправлено на сторінку для базового ознайомлення з функціоналом сайту (Додаток).

На сторінці входу, користувач має ввести електронну пошту та пароль. Також тут можна виконати вхід через кабінет або відновити пароль. Для реєстрації користувачеві необхідно заповнити спеціальну форму (рис. 5.1), після цього необхідно буде дочекатися підтвердження заявки адміністратором.

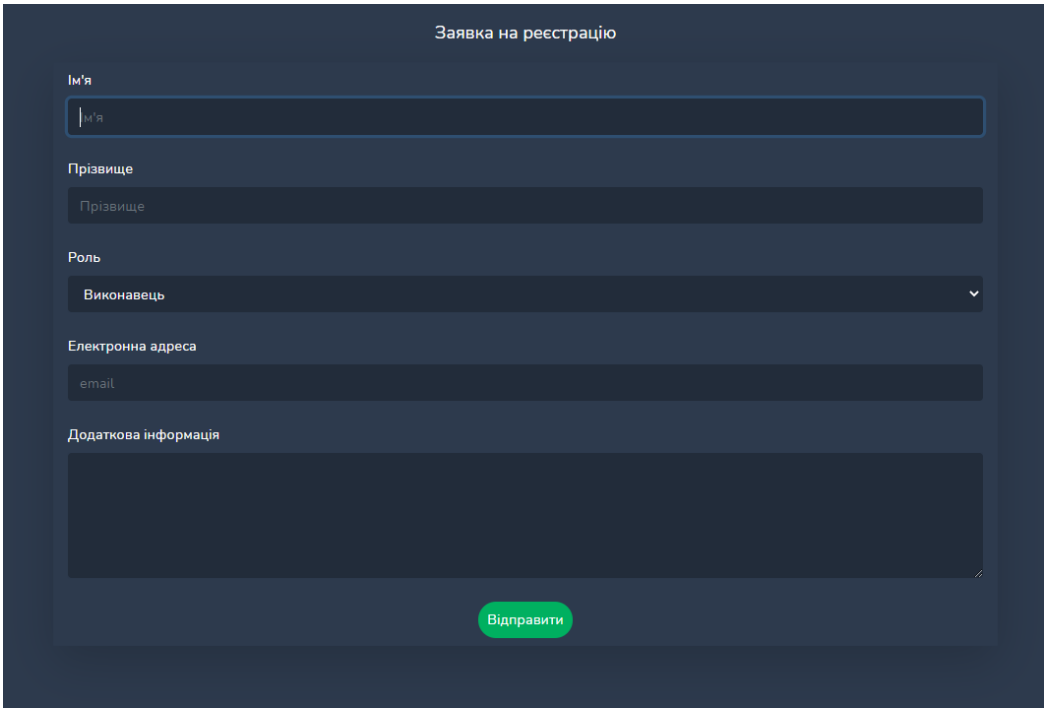


Рисунок 5.1 – Форма реєстрації

Після виконання входу користувач отримає доступ набору функцій, що залежать від його ролі: замовник або виконавець (Додаток).

Для авторизованого користувача головна сторінка доповниться декількома функціями, що були недоступні для неавторизаваного

користувача (рис. 5.2). В вверхній частині сайту знаходиться навігаційне меню, також користувач матиме змогу переглядати будь-яке замовлення.

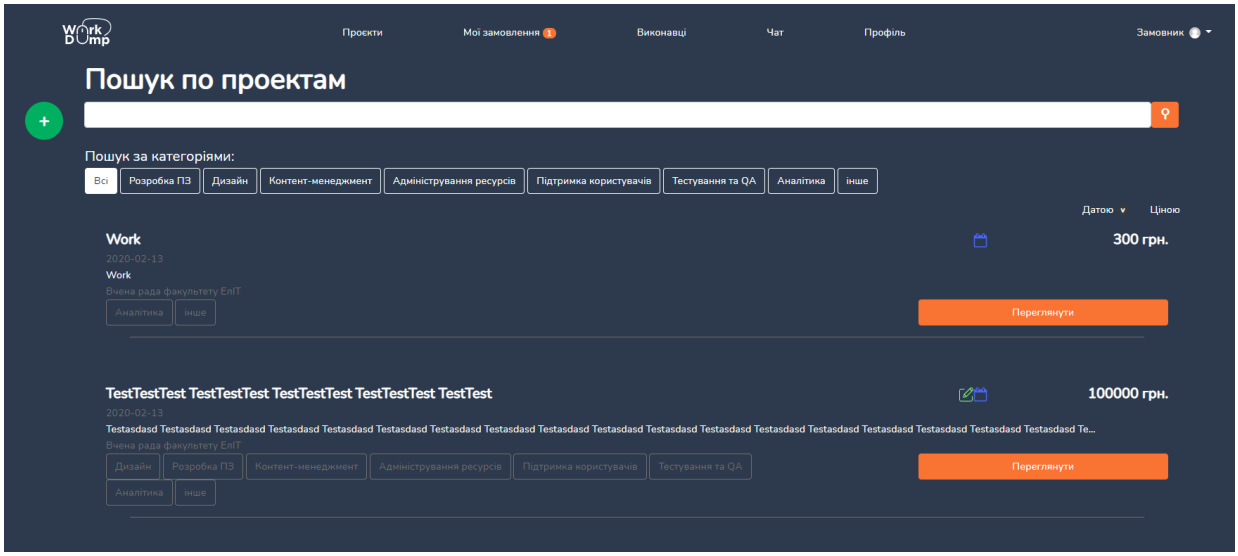


Рисунок 5.2 – Інтерфейс головної сторінки

Для створення нового замовлення користувачеві необхідно натистути на «+», що відкриє відповідну форму (рис. 5.3). Після її заповнення створене замовлення з’явиться в загальному списку.

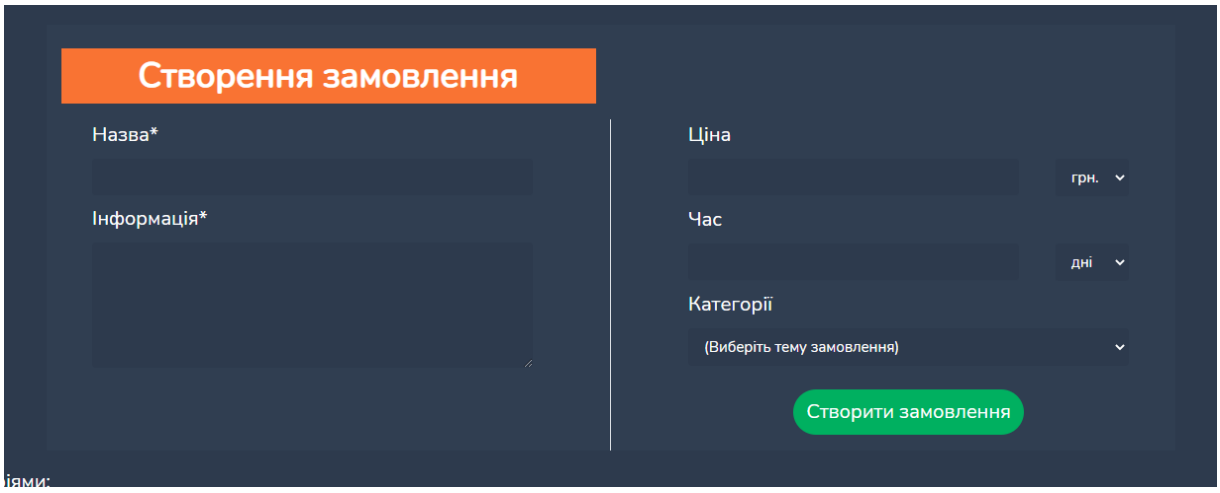


Рисунок 5.3 – Форма створення замовлення

Подальше слідкування за створеними замовленнями виконується через відповідну сторінку «Мої замовлення». На даній сторінці відображаються замовлення з трьома можливими статусами:

- відкриті – відображаються в загальному списку, не мають підтвердженого виконавця, можуть бути змінені;
- виконуються – не відображаються в загальному списку, мають виконавця, не можуть бути знінені;
- завершені – список виконаних замовлень; заповнивши спеціальну форму можна залишити відгук виконавцю.

В особистому профілі (рис. 5.4) користувач матиме змогу відредагувати свої персональні дані, переглянути свої замовлення або переглянути відгуки залишені виконавцями.

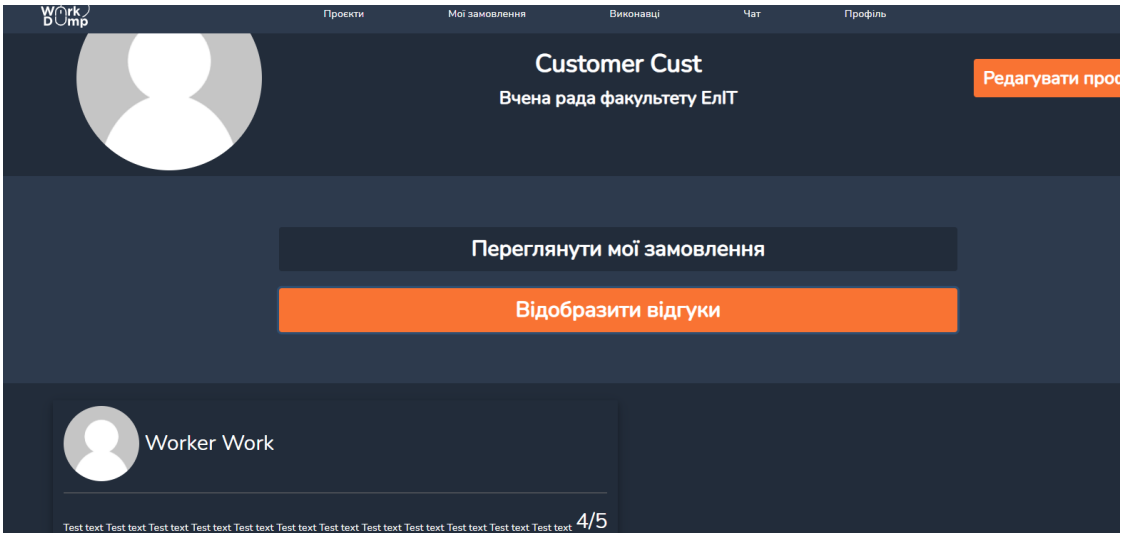


Рисунок 5.4 – Профіль користувача

Вибравши пункт бокового меню «Налаштування» замовник матиме змогу змінити свій пароль, тип та назву підрозділу у відповідних випадаючих списках (рис. 5.5).

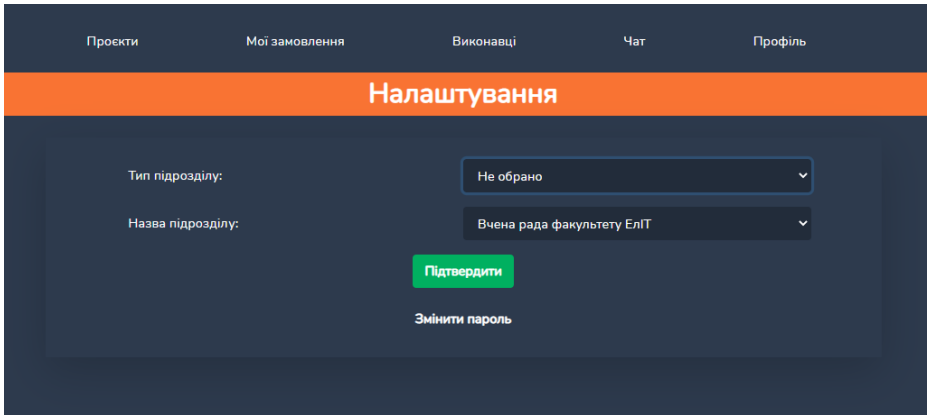


Рисунок 5.5 – Сторінка налаштувань

Також на сайті реалізована панель адміністратора (рис. 5.6). Вхід до даної панелі виконується через сторінку входу.

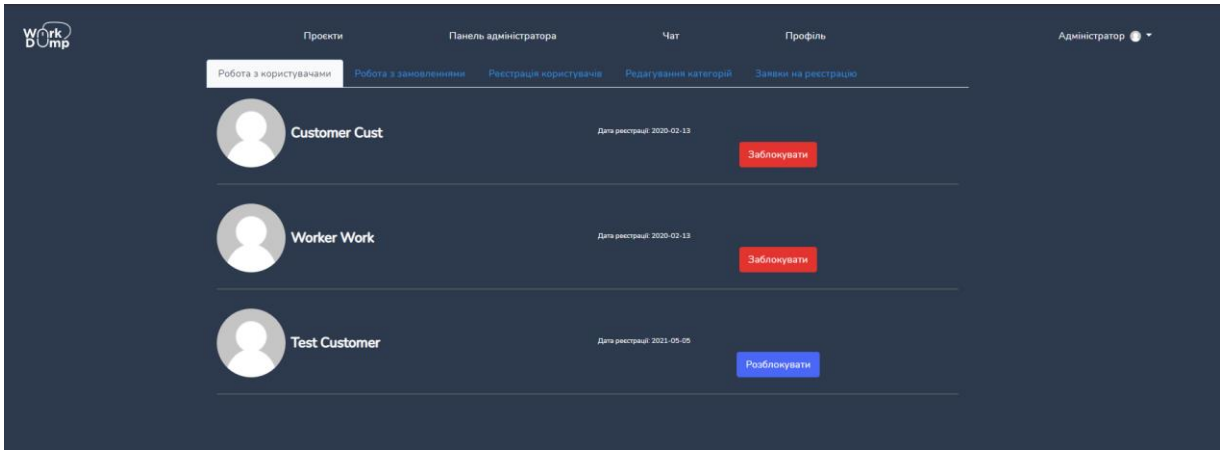


Рисунок 5.6 – Панель адміністратора, вкладка «Робота з користувачами»

На панелі адміністратора, у вкладці «Робота з користувачами» (рис. 5.7), адміністратор матиме змогу переглядати список користувачів, і в разі необхідності блокувати або розблоковувати користувачів, що порушили правила сайту.

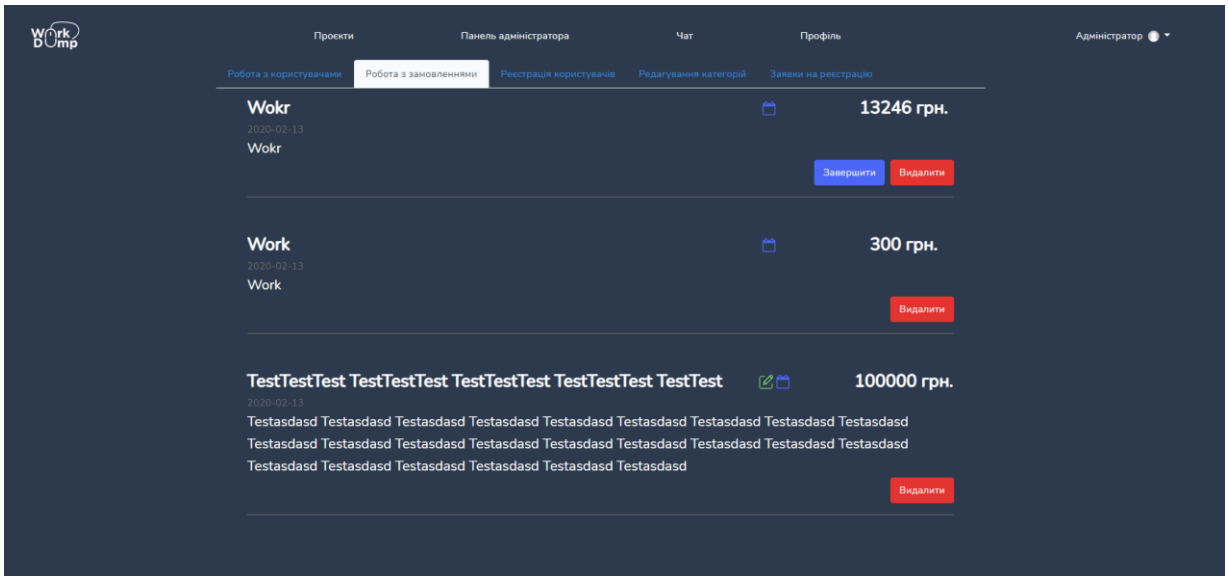


Рисунок 5.7 – Вкладка «Робота з замовленнями»

У вкладці «Робота з замовленнями» (рис. 5.8) адміністратор може достроково завершити замовлення у статусі «виконується» або видалити ті, що у статусі «відкриті»

Вкладка «Реєстрація користувача» являє собою форму реєстрації нового користувача.

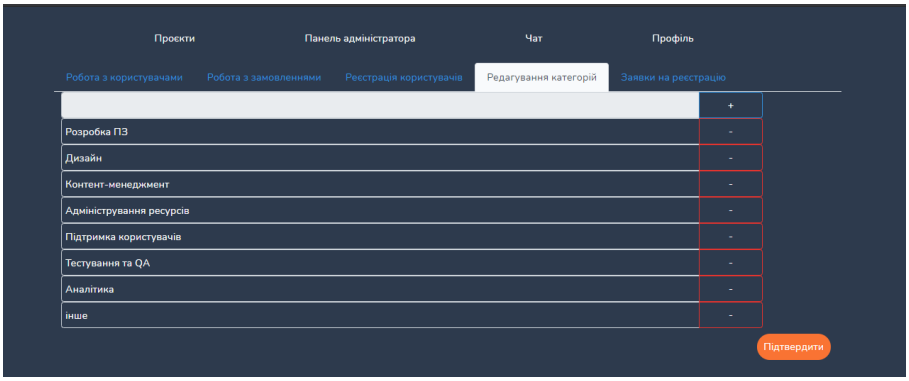


Рисунок 5.8 – Вкладка «Редагування категорій»

Вкладка «Редагування категорій» (рис. 5.9) призначена для редагування списку категорій, що використовуються на сайті.

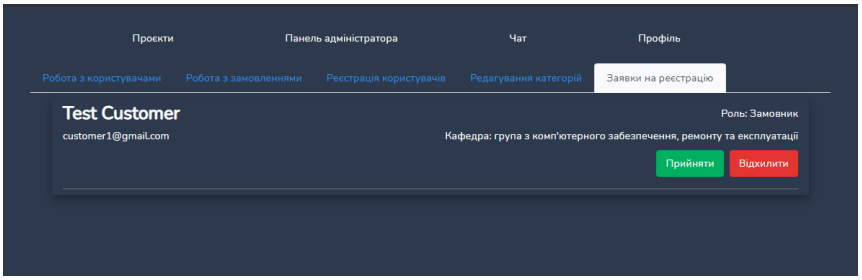


Рисунок 5.9 – Вкладка «Заявки на реєстрацію»

Вкладка «Заявки на реєстрацію» (рис. 5.9) призначена для розгляду огляду залишених заявок на реєстрацію з можливістю прийняти або відхилити заявку.

ВИСНОВОК

В кваліфікаційній роботі було досліджено та реалізовано інформаційну технологію інтеграції API СумДУ до стороннього сайту на платформі Laravel.

Проведено аналіз існуючих платформ розробки, їх основні переваги та недоліки. На основі цього аналізу сформований ряд критеріїв для вибору майбутньої платформи. Спираючись на ці критерії була обрана платформа Laravel.

В роботі проаналізовані головні особливості платформи Laravel. Також наведено бузпосереднє застосування цих особливостей про розробці інформаційної технології.

Серверна частина сайту виконана за допомогою сценарної мови PHP, яка є основою платформи Laravel та системи керування базами даних MySQL. Для роботи над клієнтською частиною сайту використовувались Bootstrap (набір інструментів для створення веб-сайтів, який містить шаблони CSS та HTML) та JavaScript (мова програмування, що найчастіше використовується для сценаріїв вебсторінок). На сайті реалізована система додавання та редагування замовлень, вибору ролі користувача з відповідними можливостями, простий чат та панель адміністратора.

В якості інтегрованих API були задіяні список кафедр, факультетів та інститутів, а також авторизація через особистий кабінет. Список кафедр розширив можливості налаштування профілю користувача у ролі замовника, в той час як авторизація через особистий кабінет надала змогу використовувати дані для автоматичного заповнення профілю користувача на сайті.

Результати роботи були опробовані на конференції IMA 2021 [21].

В кваліфікаційеій роботі було доведено, що інтеграція API СумДУ до сайтів на платформі Laravel можлива та доцільна.

СПИСОК ЛІТЕРАТУРИ

1. Laravel - https://laravel.com/
2. Symfony - https://symfony.com/doc/current/index.html
3. Django - https://docs.djangoproject.com/en/3.2/
4. AngularJS - https://docs.angularjs.org/guide
5. Yii2 - https://www.yiiframework.com/doc/guide/2.0/en
6. API - https://en.wikipedia.org/wiki/API
7. Клієнт-серверна архітектура [електронний ресурс] - https://habr.com/ru/post/495698/
8. Visual Studio Code - https://code.visualstudio.com/learn
9. 17 приемуществ использования Laravel в IT–индустрии - https://wezom.com.ua/blog/17-preimuschestv-ispolzovanija-laravel-v-it-industrii
10. What is MySQL? - http://dev.mysql.com/doc/refman/5.7/en/what-is-mysql.html
11. Bringing MySQL to the web - https://www.phpmyadmin.net/
12. BootStrap - https://getbootstrap.com/docs/4.1/getting-started/introduction/
13. JavaScript HTML DOM Document - https://www.w3schools.com/js/js_htmldom_document.asp
14. Learning Javascript with Object Graphs - http://howtonode.org/object-graphs
15. База данных: наполнение данными (seeding) - https://laravel.su/docs/5.2/seeding
16. Маршрутизация - https://laravel.ru/docs/v5/routing
17. Котроллеры - https://laravel.ru/docs/v5/controllers
18. Middleware - https://laravel.ru/docs/v5/middleware
19. Шаблоны Blade - https://laravel.ru/docs/v5/blade
20. jQuery - http://www.wisdomweb.ru/JQ/jquery-first.php
21. Фесенко О.І., Ободяк В. К. Інтеграція API СумДУ з сайтами на платформі Laravel: Інформатика, математика, автоматика. IMA::2021. Матеріали міжнародної науково-технічної конференції студентів та молодих учених (Суми, 19-23 квітня 2021 року) – Суми: СумДУ, 2021. –. С. 37.

ДОДАТОК

Файл front.js

```
$("document").ready(function() {
    const Swal = require('sweetalert2');
    $(".alert").delay(3000).slideUp();
    $(window).scroll(function() {
        var $height = $(window).scrollTop();
        if($height > 150) {
            $('#anchor').addClass('d-flex').show();
        } else {
            $('#anchor').removeClass('d-flex').hide();
        }
    });
    $("#anchor").on("click", function () {
        $("html").animate({ scrollTop: 0 }, 300);
    });
    $.ajaxSetup({
        beforeSend: function (xhr) {
            xhr.setRequestHeader('X-CSRF-TOKEN', $('meta[name="csrf-
token"]').attr('content'));
        }
    });
    update_listeners();
    $(".to-profile").on('click', function () {
        window.location.href = '/profile/' + $(this).attr('data-id');
    });
    $('.to-edit').on('click', function () {
        window.location.href = '/profile';
    });
    $('#new_order-toggle').on('click', function () {
        $(this).find('.order_circle').css('transition', 'transform 0.1s linea
r').css('transform', $('#new-
order').css('display') == 'none' ? 'rotate(360deg)' : 'rotate(0deg)').text($(
'#new-order').css('display') == 'none' ? '-' : '+')
    });
    $(".badges_reset").on('click', function (e) {
        e.preventDefault();
        $(this).closest('form').get(0).reset();
        theme_badges_build();
    });
    $('.disable-comment').on('change', function () {
        if (!$('.reviews-rating,.reviews-comment').prop('disabled')) {
            $('.reviews-rating,.reviews-comment').prop('disabled', true);
            $('.reviews-rating,.reviews-comment').prop('required', false);
        } else {
            $('.reviews-rating,.reviews-comment').prop('disabled', false);
            $('.reviews-rating,.reviews-comment').prop('required', true);
        }
    });
    $('#delete_span').on('click', function () {
        Swal.fire({
            title: '<span class="text-white">Видалення</span>',
            html: "<span class='text-
white'>Ви впевнені, що хочете це зробити?</span>",
            showCancelButton: true,
            showCloseButton: true,
            background: '#303E51',
            confirmButtonColor: '#d33',
            cancelButtonColor: '#303e51',
            confirmButtonText: 'Видалити',
            cancelButtonText: 'Скасувати'
```

```
        }).then((result) => {
            if (result.value) {
                $('#delete').submit();
            }
        });
    });

    $(".toggle-box").on('click', '.toggle-plus', function () {
        let counter = parseInt($(this).closest('.container').attr('data-
id')) + 1;
        let name = 'new-' + $(this).closest('.container').attr('id');
        let str = "<div class='form-row input-
group'><input type='text' class='form-control col-10 bg-deep-dark text-
white' name='"+name+"-"+counter+"'><input type='button' class='btn-outline-
danger form-control col-1 toggle-minus bg-deep-dark text-white' value='-
'></div>";
        $(this).closest('.container').attr('data-id', counter);
        $(this).closest('.toggle-box').append(str);
    });
    $(".toggle-box").on('click', '.toggle-minus', function () {
        $(this).closest('.form-row').remove();
    });
    if (window.location.href.indexOf('/admin') >= 0) {
        dept_block_toggle();
    }
    $("#id_role").on('change', function () {
        dept_block_toggle();
    });
    function dept_block_toggle() {
        if ($("#id_role").val() == "Замовник") {
            $("#dept-block").removeClass('d-none').addClass('d-flex');
        }
        else {
            $("#dept-block").removeClass('d-flex').addClass('d-none');
        }
    }
    $('#message_input').on('keypress', function (e) {
        if(e.which == 13) {
            e.preventDefault();
            $('#chat-form').submit();
        }
    });
    $('#chat-form').on('submit', function(e) {
        e.preventDefault();
        let text = $('#message_input').val();
        if (text !== "") {
            $('#message_input').val('');
            $.ajax({
                url: '/chat',
                method: 'post',
                data: {
                    'text': text,
                    'id_to': $('.open-contact').attr('data-id'),
                },
                success: function (data) {
                    update_chat(data);
                    update_contact($('.open-contact'));
                }
            });
        }
    });
    $('#file_input').on('change', function () {
        if ($(this).prop('files')[0].size <= 5242880) {
            let data = new FormData(),
```

```
                file = $(this).prop('files')[0];
            data.append('file', file);
            data.append('name', file.name);
            data.append('id_to', $('.open-contact').attr('data-id'));
            $.ajax({
                url: '/send_file',
                method: 'post',
                data: data,
                contentType: false,
                processData: false,
                success: function (res) {
                    if (res === 'error') {
                        Swal.fire({
                            icon: 'error',
                            title: 'Помилка',
                            text: 'Виникла помилка при збереженні файлу'
                        });
                    }
                    else {
                        update_chat(res);
                        update_contact($('.open-contact'));
                    }
                }
            });
        }
        else {
            Swal.fire({
                icon: 'error',
                title: 'Помилка',
                text: 'Розмір файлів не має перевищувати 5мб'
            });
        }
        $('#file_input').val('');
    });
    $('#contacts-list').on('click', '.contact', function () {
        if (!$(this).hasClass('open-contact')) {
            $('.open-contact').removeClass('open-contact');
            $(this).addClass('open-contact');
            $('.contact').removeClass('pointer');
            $('.contact:not(.open-contact)').addClass('pointer');
            $(this).find('.messages-count').addClass('d-none');
            $('#chat-form .d-none').removeClass('d-none');
            $.ajax({
                headers: {
                    'X-CSRF-TOKEN': $('meta[name="csrf-
token"]').attr('content')
                },
                url: '/get_messages',
                method: 'post',
                data: {
                    'id': $(this).attr('data-id'),
                },
                success: function (data) {
                    update_chat(data);
                }
            });
        }
    });
    $('.app-event').on('click', function (e) {
        e.preventDefault();
        Swal.fire({
            html: "<span class='text-
white'>Ви впевнені, що хочете це зробити?</span>",
            background: '#303E51',
```

```
            showCancelButton: true,
            showCloseButton: true,
            confirmButtonColor: '#d33',
            cancelButtonColor: '#303e51',
            confirmButtonText: 'Так',
            cancelButtonText: 'Hi'
        }).then((result) => {
            if (result.value) {
                $(this).closest('form').submit();
            }
        });
    });
    if (window.location.href.indexOf('/chat') >= 0) {
        setTimeout(() => {
            check_messages()
        }, 4000);
    }
    if ($('#routes').length) {
        setTimeout(() => {
            check_header()
        }, 4000);
    }
    $('#file_selector').on('click', function(e) {
        e.preventDefault();
        $('#file_input').trigger('click');
    });
        setTimeout(() => {
            check_header()
        }, 4000);
    }
    function update_chat(data) {
        let new_chat = '';
        for (let i = 0; i < data.length; i++) {
            new_chat += `<div class="flex-row"><div class="`
                + ($('#my_id').attr('data-
id') == data[i]['id_from'] ? 'float-left' : 'float-right')
                + (data[i]['file'] ? ' bg-green this-is-file pointer' : ' bg-
light') + ` m-2 p-2  min-width-25 rounded" data-id="`
                + data[i]['id_message'] + `"><span title="`
                + data[i]['created_at'] + `">`
                + data[i]['text'] + `</span><br><span class="float-
right font-italic">`
                + data[i]['time'] + `</span></div></div>`;
        }
        $('#messages-list div').remove();
        $('#messages-list').append(new_chat);
    }
    $("#dept_type").on("change", function() {
        let item = $(this).children("option:selected").attr("id");
        $(".depts").hide();
        $("#dept-block ." + item).show();
    });
});
```

Файл CabinetController.php

```
<?php

namespace App\Http\Controllers;

use App\Http\Controllers\Auth\LoginController;
use App\Http\Controllers\Auth\RegisterController;
use App\Http\Middleware\RedirectIfAuthenticated;
use Closure;
use Illuminate\Cookie\CookieJar;
use Illuminate\Http\Request;
use App\Models\User;
use Illuminate\Support\Facades\Auth;
use DB;
use Illuminate\Support\Facades\Cookie;
use Illuminate\Support\Facades\File;
use Illuminate\Support\Facades\Log;
use Illuminate\Support\Facades\Redirect;
use Illuminate\Support\Facades\Session;
use Illuminate\Support\Facades\Hash;
use Illuminate\Support\Facades\Storage;

class CabinetController extends Controller
{
    //
    protected $info = "Інформаційний сервіс \"WorkDump\" для розміщення онлай
н-об'яв на виконання робіт";   // Service description
    protected $icon = "/workdump-cabinet.png";      // Service icon (48x48)
    protected $mask = 13;           // Service modes 3,2,0 (1101 bits)

    protected $cabinet_api = "https://cabinet.sumdu.edu.ua/api/";
    protected $cabinet_service = "https://cabinet.sumdu.edu.ua/index/service/
";
    protected $cabinet_service_token = "MusKavQn";

    protected $token = "Ztf8SiCvMY5bnjmkrr3u1bP8bx9oB3S8EUumbU1EvDnooeBds1d7"
;

    // Получаем параметры GET запроса
    protected $key;
    protected $mode;

    public function cabinetRequest(Request $req) {
        $this->key = !empty($req['key']) ? $req['key'] : "";
        $this->mode = !empty($req['mode']) ? $req['mode'] : 0;
        session(['key' => $this->key]);
        $this->token = $this->key;
        if (!empty($this->key)) {
            switch ($this->mode) {
                case 0:
                    break;
                case 2:
                    return response(File::get(public_path($this-
>icon)), 200)->header('Content-Type', 'image/png');
                case 3:
                    return response($this->info, 200)->header('Content-
Type', 'text/plain');
                case 100:
                    return response('', 200)->header('X-Cabinet-
Support', $this->mask);
                default:
                    exit;
            }
```

```
        }
        //return response();
    }

    /**
     * @param Request $req
     * @return $this
     */
    public function cabinetLogin(Request $req)
    {
        //$auth_key = $req->input('key');
        $this->key = session('key');
        $response = json_decode(file_get_contents($this-
>cabinet_api . 'getPerson?key=' . $this->key));
        //dd($response, $this->key, Cookie::get(), $this->token);
        //dd(Session::all(), $this->key);
        //if ($response->status == 'OK') {
        if($this->key) {
            $person = $response->result;
            $fresh = false;
            if (!User::where('guid', $person->guid)->exists()) {
                if (User::where('email', $person->email)->exists()) {
                    DB::table('users')->where('email', $person->email)-
>insert('guid', $person->guid);
                }
                else {
                    $data = [
                        'id_role' => 'Замовник',
                        'email' => $person->email,
                        'name' => $person->name,
                        'surname' => $person->surname,
                        'guid' => $person->guid
                    ];
                    $new = new RegisterController();
                    $new::create($data);
                    $fresh = true;
                }
            }
            $user = User::where('email', $person->email);
            $uid = $user->get('id')->first()->id;
            Auth::loginUsingId($uid, true);
            if(Auth::check())
                if($fresh)
                {
                    return view('auth.role');
                }
                else
                {
                    return Redirect('orders');
                }
            else {
                return back()-
>withErrors("Помилка входу. Спробуйте пізніше");
            }
        }
        else {
            //throw an error
            return Redirect::away($this->cabinet_service . $this-
>cabinet_service_token);
            //return Redirect::away('https://cabinet.sumdu.edu.ua/?goto=http:
//workdump-test.sumdu.edu.ua/cabinet-login');
            //return Redirect::away('https://cabinet.sumdu.edu.ua/?goto=http:
//workdump-test.sumdu.edu.ua/cabinet-login');
```

```
            //return Redirect::away($this->cabinet_service . $this-
>cabinet_service_token);
            return back()->withErrors("Помилка входу. Спробуйте пізніше");
        }

    }

    public function cabinetLogout(){

        if (isset($_REQUEST['logout']) && isset($_SESSION['person'])) {
            $result = json_decode(file_get_contents($this-
>cabinet_api . 'logout?key=' . $this->key), true);
        }
    }

}
```

Базове ознайомлення з функціоналом сайту

1

Зареєструйтеся

Натисніть на кнопку Вхід у верхньому правому кутку. У відкрившомуся вікні натисніть Залишити заяву. Заповніть форму відповідно, обравши Вашу роль на ресурсі (Замовник для створення замовлень на виконання робіт або Виконавець для виконання наявних замовлень) та натисніть Відправити. Дочекайтеся відповіді на вказану електронну пошту. Ви маєте отримати дані для входу на ресурс. Використовуйте їх для входу на ресурс на сторінці входу у формі справа.

2

Залишайте заяви та пропозиції

Якщо Ви Замовник:

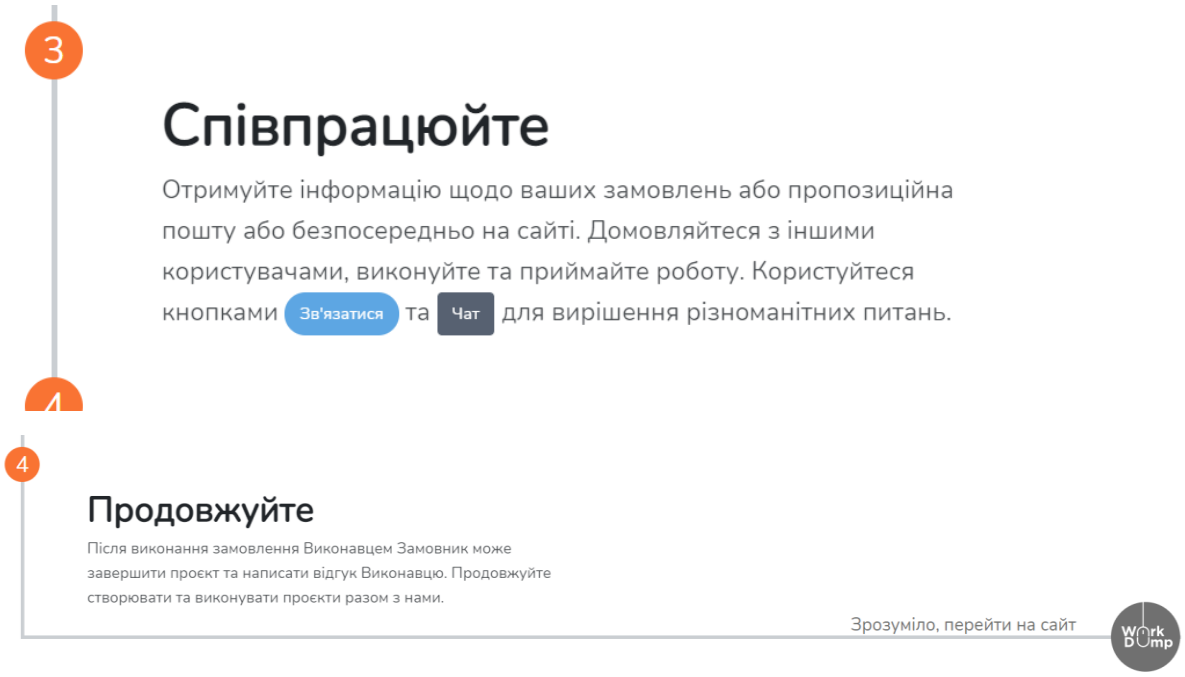
Натисніть + щоб відкрити форму створення замовлення.

Заповнуть форму вказавши необхідні дані, строки та вартість за необхідністю.

Якщо Ви виконавець:

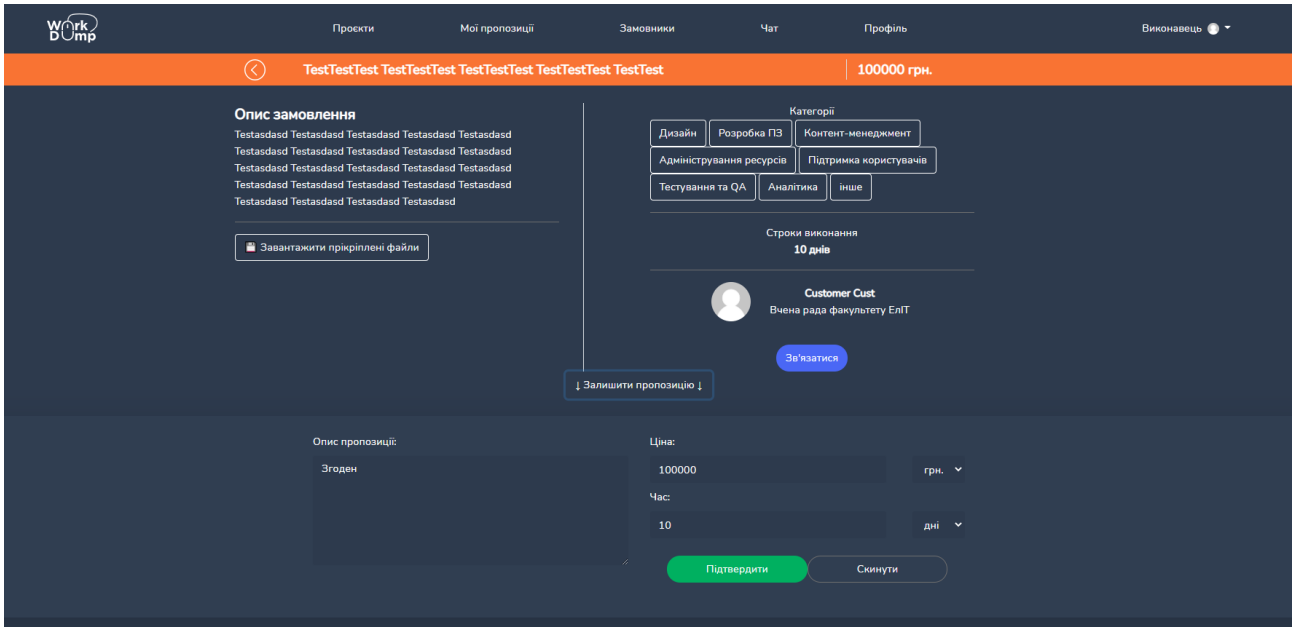
Знайдіть замовлення, що Вас цікавить на головній сторінці. Натисніть Переглянути та ознайомтеся детальніше. Якщо ви бажаєте виконати замовлення натисніть ↓ Залишити пропозицію ↓ та заповніть форму пропозиції, за необхідністю запропонуйте свої строки та вартість та поясніть чим обумовлені зміни.

3

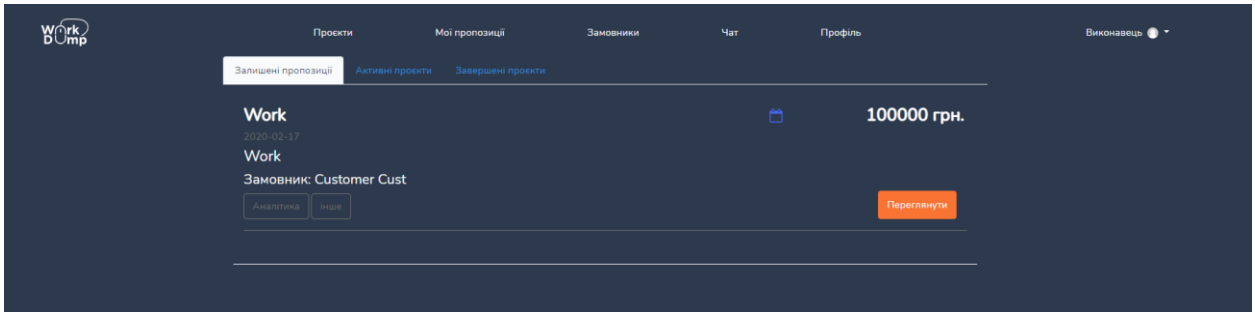

3
Співпрацюйте
Отримуйте інформацію щодо ваших замовлень або пропозиційна пошту або безпосередньо на сайті. Домовляйтеся з іншими користувачами, виконуйте та приймайте роботу. Користуйтеся кнопками Зв'язатися та Чат для вирішення різноманітних питань.
4
Продовжуйте
Після виконання замовлення Виконавцем Замовник може завершити проєкт та написати відгук Виконавцю. Продовжуйте створювати та виконувати проєкти разом з нами.
Зрозуміло, перейти на сайт

Функціонал виконавця:

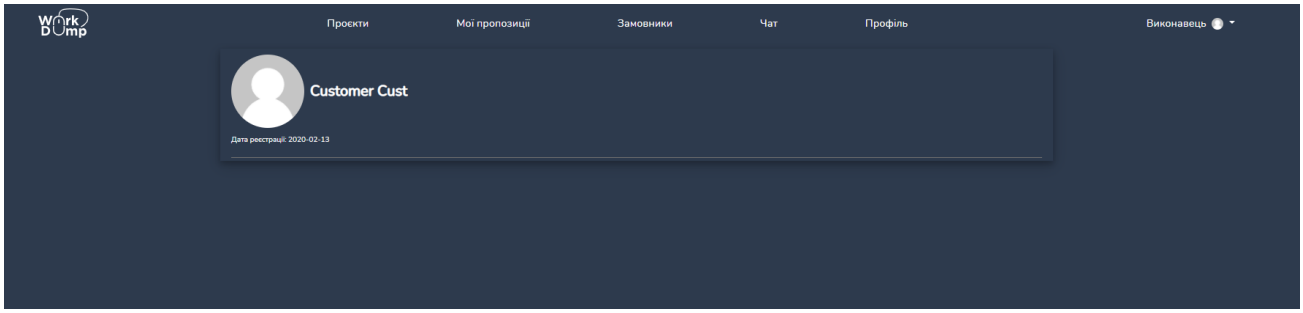
- залишення пропозиції:



- сторінка перегляду всих залишених пропозицій:



- сторінка перегляду списку замовників:



- сторінка налаштувань та зміни паролю: