Київський національний торговельно-економічний університет
Кафедра програмної інженерії та кібербезпеки

ВИПУСКНИЙ КВАЛІФІКАЦІЙНИЙ ПРОЕКТ

на тему:

Розробка інформаційної системи обліку замовлень на ремонт технічних засобів КНТЕУ

Студента 4 курсу, 10 групи,
напрям підготовки 6.050103
«Програмна інженерія» _____ Хрущ Олексій Геннадійович
підпис студента

Науковий керівник,
завідувач кафедри програмної
інженерії та кібербзпеки
доктор технічних наук,
професор _____ Криворучко Олена Володимирівна
підпис керівника

Гарант освітньої програми,
кандидат технічних наук,
доцент _____ Цензура Микола Олександрович
підпис керівника

Київ 2019

ЗМІСТ

ВСТУП

Актуальність теми: будь-який ремонт комп'ютерів, ноутбуків або серверів завжди починається з діагностики. Адже перш ніж лікувати хворого, треба зрозуміти, на що він хворий. Діагностика комп'ютера логічно розбивається на два етапи: спочатку майстер діагностує несправність, на яку скаржиться користувач, потім аналізує загальний стан комп'ютера. Первинна діагностика комп'ютера буде тим швидше і точніше, чим більше інформації майстер отримає від користувача, а саме, які дії проводилися за комп'ютером перед поломкою, які програми запускалася і які сайти відвідувались. Знаючи ці дані, майстру буде легше встановити причинно-наслідковий зв'язок і усунути несправність.

Для вирішення цієї проблеми – створюється веб-сайт, за допомогою якої користувачі мають змогу залишити замовлення, після чого відповідальні люди за ремонт техніки оброблюють замовлення, збирають первинну інформацію та приступають до починки техніки.

Подальша діагностика комп'ютера вимагає досвіду, високої кваліфікації, а також набору необхідних програм та тестування обладнання. При діагностиці апаратної частини комп'ютера перевіряються показники датчиків температури, напруги, різні показники стану жорсткого диска та інші. Якщо комп'ютер не завантажується, то несправність визначається за сигналами BIOS або показником post-тестера.

Проблеми обробки замовлень в центрі інформаційних технологій:

- тривалий процес обробки замовлень.
- відсутність систематизації замовлень.
- відсутність пошуку та фільтрації по замовленням.

- відсутність відстеження замовлень.

Об’єкт дослідження – організація послуг обліку ремонту.

Предмет дослідження – методи та алгоритми створення програмного забезпечення інформаційної системи обліку замовлень на ремонт технічних засобів КНТЕУ.

Мета – розробка та впровадження інформаційної системи обліку замовлень на ремонт технічних засобів КНТЕУ, спрямованої на збільшення ефективності функціонування закладу вищої освіти.

Завдання проекту – забезпечення безперебійної роботи центру інформаційних технологій та оптимізувати процес обробки замовлень на ремонт технічних засобів в онлайн.

Практична значущість – полягає в тому, що цей проект можна бути використовувати для обліку замовлень на ремонт технічних засобів в електронному вигляді та збільшити ефективність функціонування закладу вищої освіти.

РОЗДІЛ 1
ЗАГАЛЬНІ ВІДОМОСТІ ТА ВИМОГИ ДО СИСТЕМИ

1.1. Як працює сервер з PHP

PHP є популярною мовою на стороні сервера, що особливо добре для веб-додатків. Деякі з найбільших компаній і організацій з усього світу використовують PHP для своїх розробок. Велика кількість веб-сайтів і програм працює на PHP, тому це досить популярна мова програмування, але необхідно розуміти основи, щоб потім без особих ускладнень працювати з популярними фреймворками (наприклад, Laravel, CodeIgniter або Symfony), та розуміти, як працюють популярні веб-сайти, та як вони обробляють дані користувача.

PHP написаний у вигляді стандартних текстових файлів з розширенням .php. Файли PHP часто зберігаються в папці в загальному каталозі веб-сервера (або в кореневому каталозі веб-сервера). У більшості систем це буде називатися public або public_html.

Життєвий цикл запиту

Отже, вот що саме відбувається, коли користувач вводить URL *http://example.com* веб-клієнт (наприклад, браузер), клієнт видає на сервер запит GET (припустимо, що використовується Apache). Коли Apache отримує цей запит, він шукає файл з ім'ям index.php (або index.html – це індексні каталоги). Якщо знайдений файл з назвою index.php, то Apache по суті робить говорить, що це PHP-файл, тому що він має розширення .php. І він збирається дати це інтерпретатору PHP.

Після того, як Apache вирішить, що це файл PHP, він передає його інтерпретатору PHP. Коли PHP отримує файл, він читає його і виконує будь-

який PHP код, який він може знайти. Після цього файл інтерпретатора PHP дає вихідний код, якщо такий є, назад до Apache. Коли Apache отримує вихідні дані з PHP, він відправляє цей вихід назад до браузера, який передає його на екран.

PHP і Apache

Однак сучасні програми, створені на основі клієнтських MVC-середовищ, часто бачать роль зміни PHP в простому взаємодії з серверним сховищем даних.

На цій схемі припустимо, що користувач переходить на веб-сайт Laravel за адресою http://laravel.com/. Наведена нижче цифра обведена кругами, які висвітлюють різні етапи запиту. Покрокове пояснення кожного кроку за рисунком.

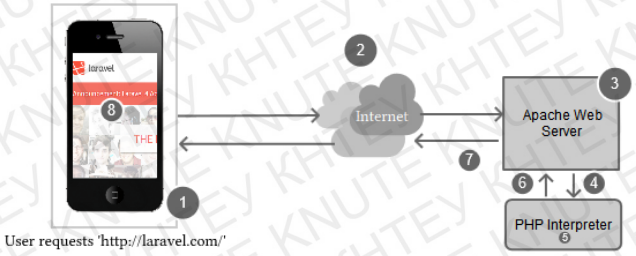

Рис 1.1 Работа сервера під час запиту

1. Користувач вводить «http://laravel.com» у свій браузер і натискає / вводить «enter».
2. Після того, як користувач натиснув «enter», браузер посилає запит сторінки через Інтернет на веб-сервер.
3. Веб-сервер отримує запит і аналізує інформацію запиту. Apache розуміє, що не вказан файл, тому він шукає індекс каталогу та знаходить «index.php».
4. Оскільки Apache знає, що потрібно відправляти файли, які закінчуються розширенням `.php`, в інтерпретатор PHP, він просить PHP виконати файл.

5. На цьому кроці, PHP виконує код, що міститься у файлі «index.php» з запиту. Під час цього кроку, PHP може взаємодіяти з базами даних, файловою системою або робити зовнішні виклики API.
6. Після того, як PHP закінчив виконувати файл «index.php», він надсилає вихідні дані до Apache.
7. Apache отримує вихідні дані з PHP і надсилає їх через Інтернет до веб-браузера користувача. Це називається – «веб-відповідь» (web-response).
8. Веб-браузер користувача отримує відповідь від сервера і переглядає веб-сторінку на комп'ютері або пристрої.

1.2. Як захищатися в сучасних веб-застосунках

Загальні вразливості

Коли річ заходить про Web Security, ймовірно, перше, що спадає на думку, це HTTPS, який являє собою протокол, який описує і гарантує, що дані, що передаються по з'єднанню, надійно зашифровані.

Коли говоримо про уразливість і атаки з боку зловмисників, мова йде про дані, більш конкретно конфіденційних даних і про те, як їх захищаємо. Це може бути що завгодно, наприклад токен сеансу, електронна пошта користувача і т.д.

Таблиця 1.1.

Огляд зберігання конфіденційних даних з відповідними вразливостями

Підхід до зберігання	Вразливий до	Імунітет до
Заголовок авторизації веб-сховища	XSS	CSRF
HTTP cookie	CSRF	XSS

На жаль, localStorage ніколи не розроблявся, щоб бути безпечним. Будь-який фрагмент JavaScript може читати і писати з/до нього, що відкриває XSS-атаку вразливості (як «шкідливий» скрипт, який читається з

localStorage і посилає дані на зовнішній «шкідливий» сервер).

Захист від XSS

У застосуванні JavaScript атака XSS (Cross-Site Scripting) є однією з найбільш шкідливих атак. На щастя, браузери пропонують і реалізують кілька функцій, які допомагають запобігти цим вразливостям.

Заголовок безпеки - це просто заголовок HTTP-відповіді, спеціально розроблений для цілей безпеки.

Заголовки безпеки (CSP & SRI, HSTS, HPKP)

- CSP

Заголовок CSP описує ряд директив (або правил), які дозволяють браузеру дозволяти або блокувати ресурси для завантаження та виконання.

Content-Security-Policy: default-src 'self'; ...directives

Політика безпеки контенту є потужною функцією. Це дає нам безліч варіантів і гнучкості, щоб доручити браузеру зрозуміти контекст нашої програми або веб-сайту і вжити необхідних дій. Іншими словами, це дозволяє контролювати майже все, що відбувається в браузері (скрипти, шрифти, зображення, мережеві запити тощо).

- HTTP Public Key Pinning

Він може знадобитися, коли потрібен високий рівень безпеки, наприклад, веб-сайт банку. Виступає за закріплення відкритого ключа HTTP і має можливість доручити браузеру перевірити набір даних криптографічних відкритих ключів на веб-сайті запитувача. Забезпечення цього заголовка також значно знижує можливі атаки MiTM з підробленими сертифікатами.

Public-Key-Pins: pin-sha256=”XXX” max-age=5184000

Захист від CSRF

Міжсайтова підробка запиту (Cross-Site Request Forgery) була навколо

довше, ніж XSS і традиційно застосовується до звичайних веб-додатків (з використанням серверної візуалізації і форм). Це можливість виконувати небажані дії від імені користувача, який пройшов автентифікацію.

Найбільш поширеним методом запобігання атак з підробкою запитів між сайтами (CSRF) – є додавання маркерів CSRF до кожного запиту і приєднання їх до сеансу користувача.

CORS

Cross-Origin Resource Sharing – вона визначає політику, яка захищає від одного походження з «крадіжки» даних іншого походження.

Це часто зустрічається при розробці веб-сайту на Javascript, яка надає мережеві запити іншим походженням. Для того, щоб запит «досягнув успіху», другий сервер походження повинен дати згоду на запит через заголовок Access-Control-Allow-Origin у відповіді. Перевірка виконується за допомогою запиту «preflight» з методом HTTP OPTIONS.

1.3. Технічне завдання

Розробити веб-сайт, яка складається з трьох частин – клієнтська частина, серверна частина, та серверна частина websocket. Серверну частину розробити за допомогою laravel фреймворку.

Спроектувати для роботи виключно для API запитів від frontend частини, авторизації та автентифікації. Для автентифікації користувача використати технологію JWT, яка зберігає токен, згенерований на сервері, тільки у користувача в локальному сховищі (localStorage), а за допомогою ключа звіряється на сервері. Весь сайт має бути доступний тільки для авторизованих користувачів.

Спроектувати реляційну базу даних, яка буде вмістити в собі всі необхідні таблиці для реалізації цього веб-сайту, а також систему ролей, яка не дозволить доступ до деяких частин сайту.

Сервер має підтримувати пошук, фільтрацію та реалізувати пагінацію для відображення списку користувачів, замовлень, обладнання та інших даних.

Реалізувати функціонал для збереження та завантаження файлів за доступами.

Розробити клієнтську частину за допомогою Vue.js фреймворку, та використанням екосистеми – vue-router, vuex, з подальшою компіляції за допомогою webpack та babel технологій.

Розробити серверну частину під websocket для підтримувати оновлення в реальному часі, розробити за допомогою мови програмування node.

1.4. Висновки до розділу 1

Створення односторінкової веб-програми вимагає низки сучасних технологій розвитку, але завдяки розумній технології сервера та клієнта, SPA-системи швидко стають найбільш динамічними та доступними програмами.

- Більш просте відстеження стану – немає необхідності використовувати файли cookie, відправку форм, локальне сховище, сховище сеансів і т. д. Для запам'ятовування стану між завантаженнями двох сторінок.
- Вміст на кожній сторінці (заголовок, нижній колонтитул, логотип, банер з авторськими правами і т. Д.) Завантажується тільки один раз за типову сесію браузера.

Забезпечення безпеки веб-ресурсу – це процес, що поєднує в собі певний набір дій. Сформована система спершу досліджується на предмет безпеки, потім визначається ряд заходів і робіт, що проробляються для досягнення цієї безпеки.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ПРОГРАМНОГО ПРОДУКТУ

2.1. Опис програмного продукту

Ідеєю цього проекту є створення інформаційної системи, за допомогою якою працівники вищого навчального закладу мають можливість створювати електронне замовлення на ремонт технічного засобу.

За допомогою цього сайту, вищий навчальний заклад отримає:

- Систему керування замовленнями, з можливістю підкріпленням файлів та системою коментування.
- Систему керування користувачами з редагування декілька ролей на одного користувача.
- Систему керування обладнання з можливістю підкріпленням файлів.
- Можливість редагувати глобальні налаштування на сайті (зміна логотипу, назви, favicon та ін.).
- Можливість мати доступ до сайту без доступу до інтернету (за допомогою PWA технології).
- Можливість редагувати маніфест файлу в онлайн, для встановлення веб-сайту на робочий стіл смартфону або ж на ПК.
- Систему ролей, де кожна роль має свої доступи до окремих частин сайту.
- Отримувати нову інформацію в реальному часі за допомогою технології Websocket.

2.2. Проектування бази даних

Всі запити на сервері відбуваються через Eloquent ORM – забезпечує красиву, просту реалізацію ActiveRecord для роботи з базою даних. Кожна таблиця бази даних має відповідну «Модель», яка використовується для взаємодії з цією таблицею. Моделі дозволяють запитувати дані у таблиці, а також вставляти нові записи в таблицю.

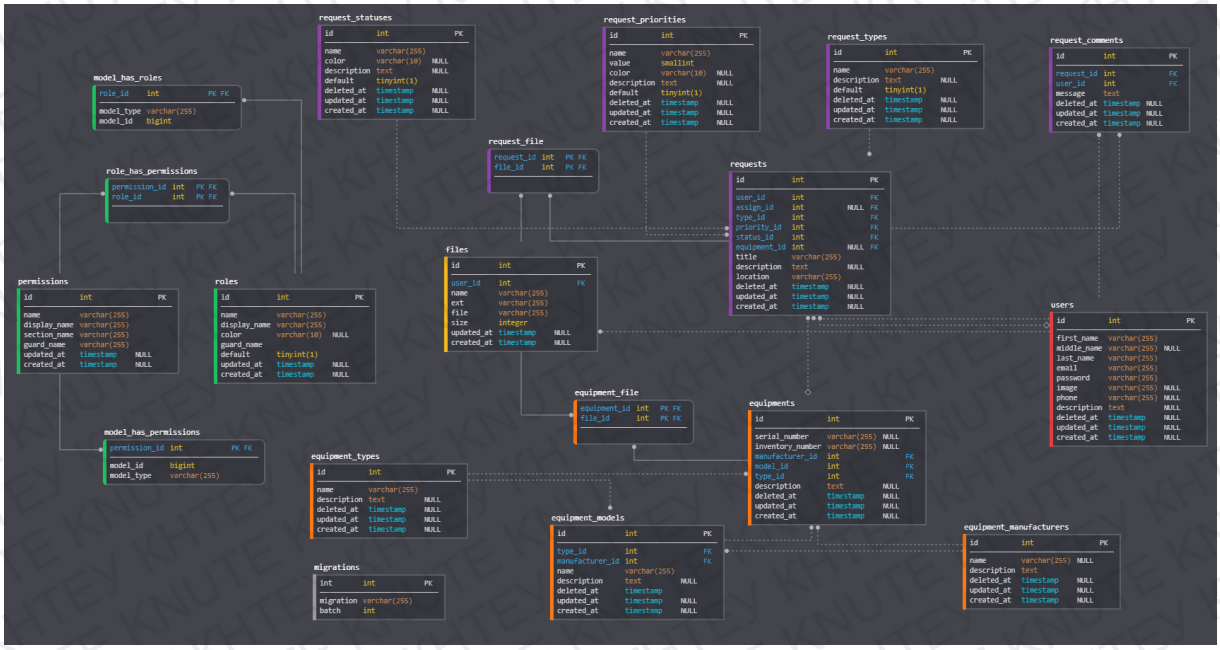

Рис 2.1. Дизайн бази даних

База даних поділена на секції:

- Зелений колір – це таблиці, які пов'язані з ролями та доступами, присутні морфні зв'язки.

1. roles – список всіх ролей на сайті.
2. model_has_roles – які ролі має модель, має морфний зв'язок, наприклад – користувач (1) має роль адміністратора (1).
3. role_has_permissions – які доступи має окрема роль.
4. permissions – список всіх доступів.
5. model_has_permissions – які доступи має модель, має морфний зв’язок.

- Помаранчевий колір – це таблиці, які пов’язані з обладнанням.

1. equipments – список всіх обладнань.
2. equipment_types – список всіх типів обладнань.
3. equipment_manufacturers – список всіх виробників обладнань.
4. equipment_models – список всіх моделей обладнань.
5. equipment_file – зв’язок many to many між файлами та обладнанням.

- Сірий колір – це таблиці, які пов’язані з мають різні значення.

1. migrations – це таблиця, яка створюється під час міграції від фреймворку laravel, містить в собі всі створені таблиці.
2. settings – містить в собі різні дані, наприклад – для збереження глобальних налаштувань, має постійний кеш.

- Червоний колір – це таблиці, які пов’язані з користувачем.

1. users – список всіх користувачів.

- Жовтий колір – це таблиці, які пов’язані з файлами.

1. files – список всіх завантажених файлів на сайт (не містить зображення профіля).

- Фіолетовий колір – це таблиці, які пов’язані з замовленнями.

1. requests – список всіх замовлень.
2. request_statuses – список всіх статусів замовлень.
3. request_priorities – список всіх пріоритетів замовлень.
4. request_types – список всіх типів замовлень.
5. request_comments – список всі коментарів замовлень.
6. request_file – зв’язок many to many між файлами та замовленнями.

2.3. Опис інструментарію використаного при розробці проекту

- Для розробки *серверної частини*, були обрані такі технології:

Для системи авторизації була обрана технологія **JWT** (JSON Web Tokens) – це відкритий стандарт (RFC 7519), який визначає компактний і автономний спосіб безпечної передачі інформації між сторонами як об'єкт JSON.

PHP – це інтуїтивна мова сценаріїв на стороні сервера. Як і будь-яка інша мова сценаріїв, вона дозволяє розробникам створювати логіку у створенні вмісту веб-сторінок і обробляти дані, повернуті з веб-браузера. PHP також містить ряд розширень, які дозволяють легко взаємодіяти з базами даних, витягуючи дані для відображення на веб-сторінці і зберігаючи інформацію, введену відвідувачем веб-сайту, назад до бази даних.

MySQL – це швидка, проста у використанні СУБД, яка використовується для багатьох малих і великих підприємств. MySQL розробляється, продається і підтримується компанією MySQL AB.

PhpStorm – це інноваційне інтегроване середовище розробки (IDE), розроблене JetBrains для PHP і веб-розробників. Програмне забезпечення використовує IntelliJ IDEA, щоб дозволити розробникам писати код на декількох мовах, включаючи CSS, HTML5, JavaScript і Emmet.

Laravel – це фреймворк PHP з відкритим вихідним кодом, який є надійним і легким для розуміння. Вона слідує шаблону model-view-controller.

Predis – гнучкий і повноцінний клієнт Redis для PHP і HHVM.

- Для розробки *клієнтської частини*, були обрані такі технології:

JavaScript – це мова сценаріїв або програмування, яка дозволяє реалізовувати складні речі на веб-сторінках - кожен раз, коли веб-сторінка більше, ніж просто статично відобразити інформацію для перегляду – відображення своєчасного оновлення вмісту, інтерактивних карт,

анімованих 2D / 3D-графіка, прокручування відео-музичних автоматів і т.д.

Vue – це прогресивний фреймворк для створення користувацьких інтерфейсів. На відміну від фреймворків-монолітів, Vue створений придатним для поступового впровадження. Його ядро в першу чергу вирішує завдання рівня уявлення (view), що спрощує інтеграцію з іншими бібліотеками та існуючими проектами.

Webpack – це статичний модуль bundler для сучасних програм JavaScript. Коли webpack обробляє програму, вона внутрішньо створює графік залежностей, який відображає кожен модуль, необхідний вашому проекту, і генерує один або більше пакетів.

Babel – це інструмент, який використовується в основному для перетворення коду ECMAScript 2015+ у зворотну сумісну версію JavaScript у поточних і старих браузерах або середовищах.

Eslint – це тип статичного аналізу, який часто використовується для пошуку проблемних шаблонів або кодів, які не відповідають певним стилям. Для більшості мов програмування існують кодові лінти, і компілятори інколи включають linting в процес компіляції.

Scss – це таблиця стилів, складена до CSS. Вона дозволяє використовувати змінні, вкладені правила, міксини, функції та багато іншого, все з повністю CSS-сумісним синтаксисом. Scss допомагає зберігати великі таблиці стилів добре організованими.

Socket.io-client – модуль дозволяє підключатися до socket.io сервера через ws(s) протокол.

- Для розробки *серверної частини для websocket*, були обрані такі технології:

TypeScript – це мова програмування з відкритим вихідним кодом, розроблена та підтримувана корпорацією Майкрософт. Це сувора

синтаксична надмножина JavaScript і додає до мови додаткову статичну типізацію.

Jest – це тестування JavaScript, з акцентом на простоту. Він працює з проектами, що використовують: Babel, TypeScript, Node.js, React, Angular і Vue.js.

Socket.io - дозволяє здійснювати зв'язок у режимі реального часу, двонаправлений і на основі подій. Він працює на кожній платформі, браузері або пристрої, орієнтуючись однаково на надійність і швидкість.

2.4. Проектування схеми спілкування PHP і Node

Оскільки мови програмування php та node є різними, необхідно якось передавати дані між цими мовами, тобто між серверної частини на laravel та серверної частини на Node для Websocket. Для цього був обраний open-source розробка, яка є сховищем структури даних у пам'яті, який може використовуватись як база даних, кеш і брокера повідомлень.

Для цього в php створюємо систему подій на створення, редагування та видалення всіх даних з реляційної бази даних. Наприклад, як тільки запис користувача був успішно видалений, створюємо подію та за допомогою Pub/Sub, відправляємо в базу даних Redis, з цим додаємо дані, яка подія, в яку(і) кімнати (rooms) відправляти повідомлення користувачам.

На стороні nodejs створюємо систему, яка підписується на всі події від серверної частини laravel, та вже через ws протокол відправляє дані всім користувачам, які є в необхідних кімнатах (rooms), окрім автора повідомлення. Тим самим, за цією схемою роботи відбувається постійне відображення свіжої інформації на сайті.

Оскільки в проекті додана база даних Redis, то система кешування даних перенесено з файлової системи до цієї бази даних, тим самим пришвидшивши відгук запита у два рази.

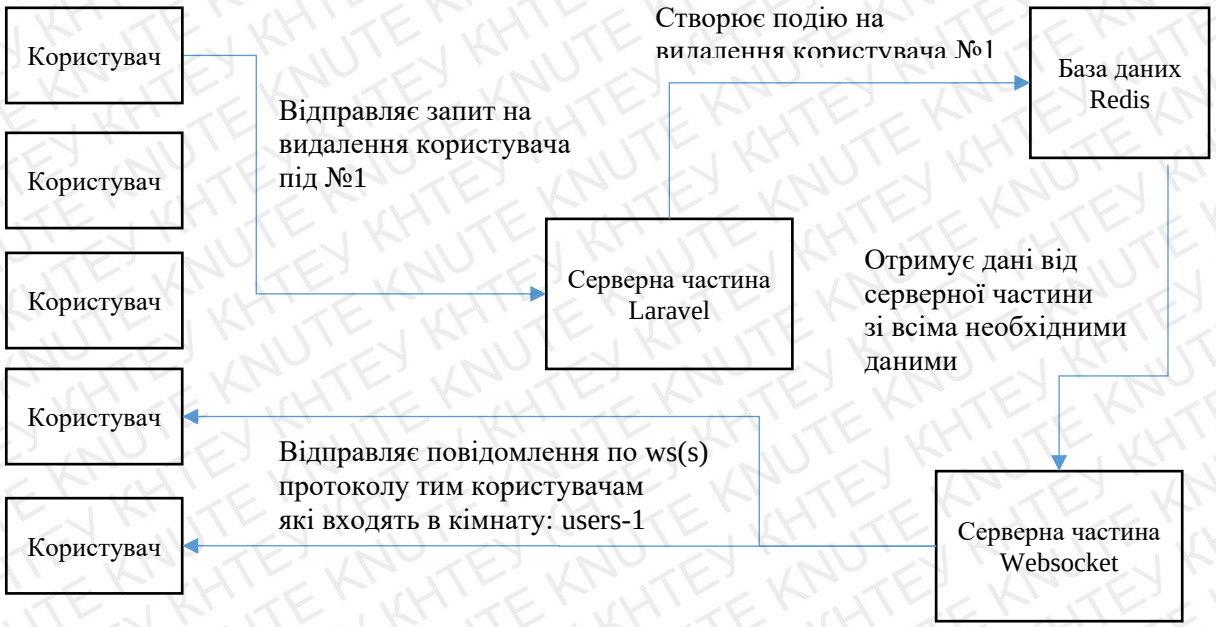

Рис 2.2. Приклад відправлення повідомлення після видалення користувача

2.5. Проектування безперервної інтеграції та доставки (CI/CD)

Кодова база проекта постійно зростає, тому необхідна система, яка б перевіряла на успішний білд всіх підсистем програми, запуск тестів та відображення покриття тестами файлів у відсотках.

Для цього в клієнтській частині, а також в серверній частині Websocket був доданий та налаштований CircleCI скрипт (див. додаток А). Після кожного пулл реквеста, CircleCI збирає проект та запускає тести.

Рис 2.3. Принцип роботи CircleCI сервіса

Якщо тести не пройшли перевірку, то сервіс відправляє повідомлення до користувача для того, щоб швидко виправити. Після тестів, створюється папка з файлами, яка відображає покриття тестами файлів і всі ці дані по тестам відправляються до Codecov сервіса, який відображає покриття

файлів в графіках.

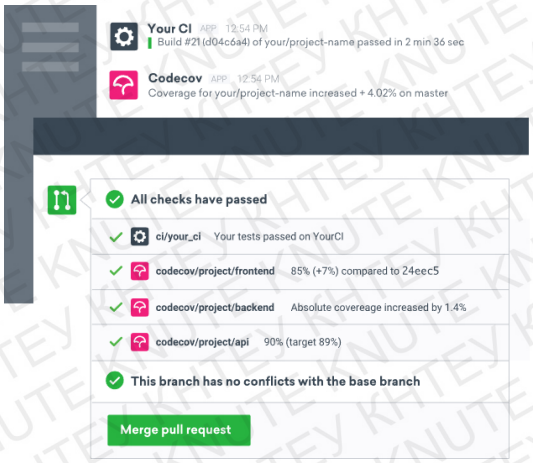

Рис 2.4. Приклад відображення повідомлення в CircleCI

2.6. Висновки до розділу 2

В даному розділі були обрані мови програмування для підсистем, а саме:

- Серверна частина на мові програмування PHP з використанням Laravel фреймворку.
- Клієнтська частина на мові програмування Javascript з використанням Vuejs фреймворку.
- Серверна частина під Websocket на мові Typescript (node).

Обрана база даних MySQL та спроектована база даних, які має різні типи зв'язку між таблицями.

Було поставлено технічне завдання для розробки веб-сайту, та описані всі технології, які будуть використовуватись в проекті.

Спроектовано схему для спілкування для абсолютно різних мов програмування – PHP та Node за допомогою використання бази даних Redis, яка водночас виступає в якості кешу для серверної частини.

Налаштували постійну перевірку системи (CI/CD) на успішний білд всіх підсистем, а також запуску написаних тестів.

РОЗДІЛ 3
РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

3.1. Структура клієнтської частини

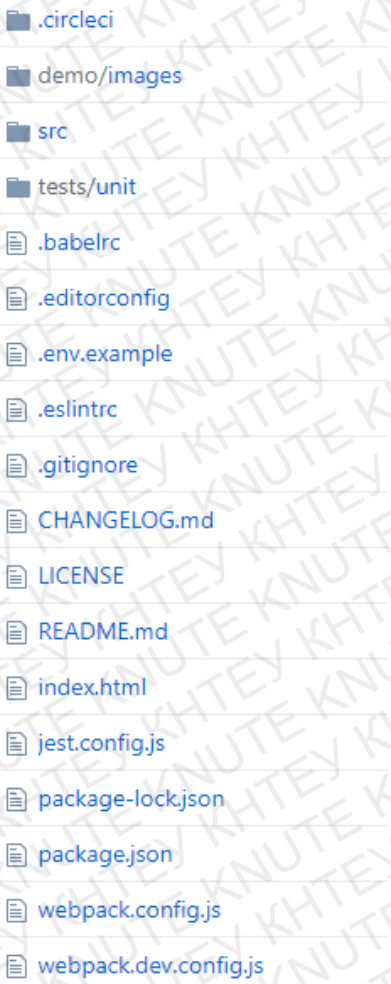

Рис. 3.1. Структура клієнтської частини, root папка

Це root папка, яка вміщає в собі базову інформацію про проект, а також різні конфігурації для різних модулів та плагінів, невеликий опис деяких файлів/папок:

- .circleci – розміщуються файл конфігурації для CircleCI сервісу.
- src – розміщуються вихідні файли проекту.

- tests – розмішуються файли, які запускаються при запуску тестів.
- .babelrc – містить конфігурацію для налаштування babel плагіну.
- .env.example – основний файл конфігурації сайту в цілому, для роботи необхідно скопіювати вміст у .env файл.
- .eslintrc – файл конфігурації для eslint плагіну (для форматування кода к єдиному стилю).
- jest.config.js – містить конфігурацію по запуску тестів.
- package.json – перелічені список всіх бібліотек, які використовуються в системі.
- webpack.config.js та webpack.dev.config.js – містить в собі конфігурацію по зборці всього проекту для webpack.

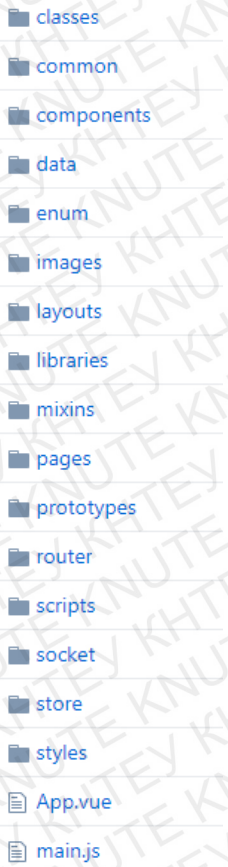

Рис. 3.2. Структура клієнтської частини, src папка

- classes – розміщуються класи, які вміщають в собі запити для роботи з API.
- common – розміщуються Vue та js файли, які мають типову структуру та використовуються багато разів (наприклад – діалогове вікно на видалення якого-небудь запису).
- components – розміщуються компоненти, близько 100, які використовуються в інших компонентів, зазвичай з pages папки.
- data – розміщуються файли, які містять дані, такі як: валідація, меню, колонки під таблиці і т.д.
- enum – розміщуються файли, які мають якісь перелік даних, наприклад – всі типи дозволів до системи (permissions)
- layouts – розміщуються vue компоненти, в яких є базова заготовка або логіка до сторінок
- mixins – розміщуються файли, функціонал яких багаторазово використовуються в vue компонентів
- pages – перелік сторінок, які є на сайті
- prototypes – глобальні функції в Vue
- router – реалізація Vue-router бібліотеки
- socket – реалізація socket.io-client, та логіка по обробці даних с websocket
- store – реалізація і модулі для Vuex бібліотеки
- styles – зберігання глобальних scss стилей
- App.vue – головний компонент, всієї системи
- main.js – імпорт найважливіших файлів та початок функціонування всієї системи при заході на сайт, а також є головним файлом для webpack.

3.2. Розробка системи автентифікації

Для початку необхідно створити всі умови на сервері. Для цього був створені необхідний роут, за яким користувач має змогу отримати токен для подальшої роботи з сервером:

Route::post('auth/login', 'AuthController@login');

Також для валідації вхідних даних, був створений AuthRequest клас (див. додаток Б), який має правила для перевірки даних з запита. Відправляючи POST запит на url адресу */api/auth/login*, з даними e-mail (email) і пароля (password).

Запит потрапляє на AuthController клас (див. додаток В), login метод, який в свою чергу робить запит до бази даних та знаходить користувача з таким email.

*SELECT * FROM `users` WHERE `email` = ? AND `users`.`deleted_at` IS NULL LIMIT 1*

Перевіряється хеш пароля і, якщо всі перевірки пройшли успішно – сервер генерує JWT токен та повертає браузеру json, який складається з:

- Повідомлення
- Токена
- Дані про користувача
- Доступи

В свою чергу, клієнтська частина має interceptors, який обробляє всі відповіді на запити, які надходять через axios бібліотеку (див. додаток Г), тобто перше користувачу відобразиться повідомлення, що він успішно увійшов у систему. Після чого оновлюються дані користувача в системі, його доступи та відправляється ще один запит на сервер, для приєднання до кімнат у Websocket (див. додаток Ґ).

Рис. 3.3. Інтерфейс сторінки авторизації

3.3. Написання тестів для серверної частини Websocket

Для того, щоб написати тест для перевірки роботи Websocket, необхідно запустити серверну частину Websocket та базу даних Redis.

Створюємо файл *socket.test.ts*.та імпортуємо необхідні файли та залежності:

```
import io from 'socket.io-client';
import Redis from 'ioredis';
import * as types from '../src/enum/types';
import { port } from '../src/env';
```

Робимо підключення до бази даних Redis та створюємо 2 дві змінні для подальшого підключення до websocket серверу:

```
let socket1: SocketIOClient.Socket;
let socket2: SocketIOClient.Socket;
const redis = new Redis();
```

Далі необхідно зробити 2 підключення до websocket, для цього перед початком всіх тестів (beforeEach хук в jest), необхідно створити підключення:

```
beforeEach((done) => {
  socket1 = io.connect('http://localhost:' + port);
```

```
    socket2 = io.connect('http://localhost:' + port);
    socket1.on('connect', () => if (socket1.connected && socket2.connected) done(); );
    socket2.on('connect', () => if (socket1.connected && socket2.connected) done(); );
  });
```

Створюємо тест, в якій користувач (socket1) відправляє запит на серверну частину, щоб мати можливість прослуховувати події, які відбуваються в кімнаті *test-1,* та після успішної авторизація, за допомогою Pub/Sub технології в Redis, серверна частина Websocket отримує дані, та додає цей сокет до необхідної кімнати. В свою чергу користувач (socket2) створює новий запис через сервер, та за допомогою Redis, серверна частина Websocket отримує подію для кімнати *test-1* і відправляє по ws протоколу оновлення через активний з'єднання.

```
test('Socket1 joins the room, and Socket2 makes an event', (done) => {
    socket1.once('test', (data: any) => {
      expect({data: {name: 'Test'}, type: types.CREATE}).toStrictEqual(data);
      done();
    });
    // Socket1 join to room: test-1
    redisEvent('test', types.JOIN, socket1.id, null, ['test-1']);
    // Socket2 create something
    redisEvent('test', types.CREATE, socket2.id, {name: 'Test'}, ['test-1', 'test-2']);
  });
});
```

Після закінчення всіх тестів, для коректного завершення виконання команди, необхідно завершити всі активні з'єднання, тому після виконання всіх тестів (afterAll хук в jest) додаємо:

```
afterAll(async (done) => {
  await redis.quit();
  done();
});
```

3.4. Висновки до розділу 3

В даному розділі було описано практичну частину випускного кваліфікаційного проекту. Описана структура клієнтської частини проекту, яка була розроблена для підтримання кодової бази проекту.

Також був розроблений API для сервера, який дотримується паттерна проектування MVC.

JWT технологія для автентифікація показала швидкодію, ніж звичайна реалізація та зберігання токена в базі даних, оскільки немає необхідності робити запити в БД. В базовій конфігурації JWT, токен змінюється кожний час, що збільшує надійність аккаунта від можливих атак.

Для розробки в режимі dev, був створений seeder для багатьох таблиць, які вже мають базові наповненнями різними контрольними прикладами.

Завдяки розробленій системі ролей – маємо можливість назначати декілька ролей на одного користувача, що робить систему більш гнучкою до змін та можливостей для корегування веб-сайту.

ВИСНОВКИ ТА ПРОПОЗИЦІЇ

В результаті виконання випускного кваліфікаційного проекту був визначений об’єкт, предмет дослідження, а також мета та завдання проекту. В першому розділі було розглянуто:

- Принцип роботи сервера з мовою програмування PHP під час виконання запиту до певного сайту.
- Як захищатися в сучасних веб застосунках – XSS, CSRF, CORS та інші.
- Створено технічне завдання – які технології, та мови програмування необхідні для створення проекта.

В другому розділі описаний програмний продукт, які системи керування будуть реалізовані на сайті, спроектували базу даних на 19 таблиць, які поділені на кольори і мають різні зв’язки між собою, а також:

- Описаний інструментарій, який був використаний при розробці веб-сайту.
- Розроблена схема роботи різних мов програмування – PHP і Node, за допомогою використання бази даних Redis.
- Налаштована безперервна безперервної інтеграції та доставки (CI/CD).

В третьому розділі була описана структуру клієнтської частини, яка поділена на багато папок, які містять в собі певну логіку, що дозволяє без ускладнень розробляти нові розділи, або вдосконалювати існуючі. Також була розроблена система автентифікації на основі JWT токенів, яка охоплює всі підсистеми веб-сайту, починаючи від інтерфейсу і закінчуючи проектуванням і розробкою серверної частини Websocket, на яку в

подальшому були написані тести.

За допомогою цього проекту, були вирішені такі проблеми при роботі з замовленнями:

- Тривалий процес обробки замовлень.
- Відсутність систематизації замовлень.
- Відсутність пошуку та фільтрації по замовленням.
- Відсутність відстеження замовлень.

Даний випускний кваліфікаційний проект був виконаний в повній відповідності до поставленого завдання. З метою подальшого розвитку та удосконалення розробленого веб-сайту – можливе доопрацювання в таких напрямках:

- Розширення функціоналу, а саме створення нових розділів, наприклад – база знань, на якому буде розміщена будь-яка інформація, яка допоможе як працівниками, так і звичайним користувачам цієї системи.
- Оптимізація та рефакторінг існуючого функціоналу для збільшення відкликання від інтерфейсу, прискорення обробки та ін.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Энди Бадд, Камерон Молл, Саймон Коллизон. CSS Mastery: Advanced Web Standards Solutions. — М.: Вильямс, 2007. — 272 с. — ISBN 1-59059-614-5

Інтернет-ресурси

1. Laravel - The PHP Framework For Web Artisans – Режим доступу: https://laravel.com/
2. PHP: Hypertext Preprocessor – Режим доступу: https://www.php.net/
3. MySQL – Режим доступу: https://www.mysql.com/
4. JSON Web Tokens (jwt.io) – Режим доступу: https://jwt.io/
5. JetBrains: Developer Tools for Professionals– Режим доступу: https://www.jetbrains.com/
6. Composer – Режим доступу: https://getcomposer.org/
7. Redis – Режим доступу: https://redis.io/
8. Vue.js – Режим доступу: https://vuejs.org/
9. Webpack – Режим доступу: https://webpack.js.org/
10. Socket.IO – Режим доступу: https://socket.io/

Додаток А

Лістинг коду «CircleCI скрипт для CI/CD»

```
version: 2.1
orbs:
  codecov: codecov/codecov@1.0.4
jobs:
  build:
    docker:
    - image: circleci/node:10.0
    # Specify service dependencies here if necessary
    # CircleCI maintains a library of pre-built images
    # documented at https://circleci.com/docs/2.0/circleci-images/
    # - image: circleci/mongo:3.4.4
    working_directory: ~/repo
    steps:
    - checkout
    # Download and cache dependencies
    - restore_cache:
        keys:
        - v1-dependencies-{{ checksum "package.json" }}
        # fallback to using the latest cache if no exact match is found
        - v1-dependencies-
    - run: npm i
    - save_cache:
        paths:
        - node_modules
        key: v1-dependencies-{{ checksum "package.json" }}
    - run: npm run test:coverage
    - codecov/upload:
                    file: coverage/*.json
```

Додаток Б

Лістинг коду «Клас для валідації запиту»

```
<?php
namespace App\Http\Requests;
use Illuminate\Foundation\Http\FormRequest;

class AuthRequest extends FormRequest
{
   /**
    * Determine if the user is authorized to make this request.
    *
    * @return bool
    */
   public function authorize()
   {
      return true;
   }

   /**
    * Get the validation rules that apply to the request.
    *
    * @return array
    */
   public function rules()
   {
      return [
         'email' => 'required|email|max:191',
         'password' => 'required|between:6,191',
      ];
   }
}
```

Додаток В

Лістинг коду «Обробка для авторизації на сервері»

```
/**
 * Auth the user by login and password.
 *
 * @param  AuthRequest  $request
 * @return \Illuminate\Http\JsonResponse
 */
public function login(AuthRequest $request)
{
    if (! $token = JWTAuth::attempt($request->only('email', 'password'))) {
        return response()->json(['message' => __('app.auth.login_error')], 422);
    }

    $user = Auth::user();

    return response()->json([
        'message' => __('app.auth.login_success'),
        'token' => $token,
        'user' => $user,
        'permissions' => $user->getAllPermissions()->pluck('name'),
    ]);
}
```

Додаток Г

Лістинг коду «Обробка всіх запитів від сервера»

```
'use strict'
import { isArray, isObject } from '@/scripts/helpers'
import { Message, Notification } from 'element-ui'
import { runLoadingService } from '@/scripts/dom'
import StorageData from '@/classes/StorageData'
import logout from '@/scripts/logout'
import types from '@/enum/types'
import store from '@/store'
import axios from 'axios'

axios.defaults.baseURL = '/api'

/** @type {Array} */
let requestsToRefresh = []

/** @type {boolean} */
let isRequestToRefresh = false

axios.interceptors.response.use(
  (resp) => {
    // Notification
    if (resp.config.method !== 'get' && resp.data.message) {
      Message({ message: resp.data.message, type: types.SUCCESS })
    }

    return resp
  },
  async (err) => {
    if (!err || !err.response) {
```

```
  return Promise.reject(err)
 }

 const { response, config } = err

 // User is not auth
 if (response.status === 401) {

  if (!store.state.profile.isLogin) {
   return Promise.reject(err)
  }

  // After token is refreshed - send requests to all 401 statusCode
  const retryOriginalRequest = new Promise((resolve) => {
   requestsToRefresh.push((access_token) => {
    config.headers['Authorization'] = 'Bearer ' + access_token
    config.baseURL = null
    resolve(axios(config))
   })
  })

  // User is auth, probably token is expired, try renew
  if (!isRequestToRefresh) {
   isRequestToRefresh = true

   // Disable all interface
   const loadingService = runLoadingService('Оновлюється токен безпеки')

   // Send request to refresh token
   try {
```

```
      await axios.post('auth/refresh')
       .then(({ data }) => {
        axios.defaults.headers['Authorization'] = 'Bearer ' + data.token
        StorageData.token = data.token
        requestsToRefresh.forEach(callback => callback(data.token))
       })
       .catch(() => {
        logout()
       })
       .finally(() => {
        loadingService.close()
        requestsToRefresh = []
        isRequestToRefresh = false
       })
     } catch (e) {
      return Promise.reject(err)
     }
    }
    return retryOriginalRequest
  }

  // No access, etc
  if (response.status === 403) {
    store.dispatch('profile/update')
  }

  // Notification / Message
  if (isObject(response.data)) {
    if (response.data.message) {
      Message({ message: response.data.message, type: types.ERROR })
```

```
    }

    // Show validate form if exists from backend
    if (isObject(response.data.errors)) {
      let message = ''

      /**
       * @example {'image.png': ['File not saved']}
       */
      Object.entries(response.data.errors).forEach(([key, val]) => {
        message += `<strong>${key}</strong>:<br>`
        if (isArray(val)) {
          message += '<ul>'
          val.forEach((error) => {
            message += '<li>' + error.replace(/<(?:.|\n)*?>/gm, '') + '</li>'
          })
          message += '</ul>'
        }
      })

      Notification.error({ title: 'Помилка валідації', duration: 6000, dangerouslyUseHTMLString: true, message })
    }
  }

  return Promise.reject(err)
 }
)

export default (Vue) => Vue.prototype.$axios = axios
```

Додаток Ґ

Лістинг коду «Auth action в Vuex сховищі»

```
auth({ commit }, data) {
  commit('SET_LOADING', true)

  return axios.post('auth/login', data)
    .then(({ data }) => {
      // Update axios and localStorage
      axios.defaults.headers['Authorization'] = `Bearer ${data.token}`
      StorageData.token = data.token

      // Sync rooms by permissions on socket server
      syncEvents()

      // Update store
      commit('SET_USER', data.user)
      commit('SET_PERMISSIONS', data.permissions)
      commit('SET_IS_LOGIN', true)

      router.push({ name: DEFAULT_ROUTE_NAME })
    })
    .finally(() => {
      commit('SET_LOADING', false)
    })
}
```