МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

Кваліфікаційна робота бакалавра

УНІВЕРСАЛЬНА СИСТЕМА КЕРУВАННЯ КОНТЕНТУ БЛОГІВ ТА ІНТЕРНЕТ-МАГАЗИНІВ

Здобувач освіти гр. ІН-83 Дрож ДЕНИС

Науковий керівник,
кандидат фізико-математичних наук,
асистент кафедри комп'ютерних наук Ольга ШУТИЛЄВА

Завідувач кафедри
доктор технічних наук, професор Анатолій ДОВБИШ

СУМИ 2022

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

Затверджую_____
Зав. кафедрою Довбиш А.С.
"_____" _____2022 р.

ЗАВДАННЯ

до кваліфікаційної роботи

здобувача вищої освіти четвертого курсу, групи ІН-83 спеціальності «122 – Комп'ютерні науки» денної форми навчання Дрожа Дениса Юрійовича.

Тема: «УНІВЕРСАЛЬНА СИСТЕМА КЕРУВАННЯ КОНТЕНТУ БЛОГІВ ТА ІНТЕРНЕТ-МАГАЗИНІВ»

Затверджена наказом по СумДУ
№_____ від _____2022 р.

Зміст пояснювальної записки: 1) літературний огляд за обраною тематикою роботи; 2) постановка завдання для розробки; 3) вибір оптимальних інструментів для розробки; 4) практична реалізація.

Дата видачі завдання «_____»_____ 2022 р.
Керівник роботи _____Ольга ШУТИЛЄВА
Завдання прийняв до виконання _____ Денис ДРОЖ

РЕФЕРАТ

Записка: стор.46 , рис. 32, табл.0, додаток 2, джерел 14.

Об’єкт дослідження CMS для створення інтернет-магазинів та блогів.

Мета роботи розробка CMS для створення інтернет-магазинів та блогів і збільшення ефективності розробника.

Методи дослідження було обрано PHP, Laravel, Eloquent, MySQL, JavaScript, Tailwind CSS, Bootstrap, Bash, Nginx, Blade, Imagick.

Результати було розроблено систему управління контенту для створення інтернет-магазинів та блогів. Після встановлення даної системи розробник має змогу сконцентрувати увагу на виконанні особливих бізнес задач клієнта, а клієнт може створювати товари для магазину та пости для блогу та відображати їх кінцевому користувачеві, отримувати замовлення.

Дана частина знаходиться в альфа-версії розробки, для її створення було застосовано як клієнтські бібліотеки такі як Tailwind CSS, Bootstrap, так і серверні такі як Imagick, Eloquent, Artisan.

CMS, PHP, MYSQL, JAVASCRIPT,
NGINX, CSS, LARAVEL

ЗМІСТ

ВСТУП

У теперішній час наявність сайту, будь-то інтернет магазин, блог, сторінка продукту це дуже важливо для бізнесу та його просування, для висловлювання своїх думок, просування продуктів різного характеру, від програмного забезпечення до послуг чи фізичних товарів.

Проектування сайтів та їх розробка з нуля це завжди довго та дорого. Витрачається багато часу на проектування, оцінювання, менеджмент, саму розробку та тестування, особливо якщо сайт розробляється не на базі фреймворків та в умовах відсутності добре спроектованих та легко підтримуваних готових рішень.

Для спрощення розробки використовують дуже часто системи управління контентом, які реалізують більшу частину бізнес завдань одразу, але в більшості випадків вони є самописними, погано запроектованими та важко підтримуваними, що завжди впливає на кінцевого користувача за рахунок швидкості роботи ресурсу.

Через це вибір розробників завжди падає на фреймворки та по суті розробку з нуля, для замовника це завжди гірше через більшу ціну кінцевого продукту. Все зазначене вище, свідчить про **актуальність теми**.

Метою кваліфікаційної роботи є вирішення у майбутньому цього питання, оскільки в даній роботі розробляється система управління контентом на базі фреймворку Laravel, що полегшує підтримку сайту розробниками та використовує оптимальні рішення архітектури для подібних ресурсів.

Випускна робота допоможе частково вирішити питання, пов'язані з розробкою та підтримкою інтернет-магазинів та блогів.

1. ЛІТЕРАТУРНИЙ ОГЛЯД

1.1 Основні поняття та структура сайту

CMS − це система управління контентом веб ресурсу, яка дозволяє створювати веб сайти з мінімальною або середньою степенню доробки зі сторони розробника. У більшості випадків CMS використовуються для створення інтернет магазинів або блогів. З найпопулярніших можна виділити WordPress як CMS для створення блогів та Opencart як CMS для створення інтернет магазинів.

Laravel − це фреймворк для мови програмування PHP що заснований на базі фреймворку Symfony. Він реалізує для розробника маршрутизацію сайту, базову структуру, з'єднання з базою даних та інструменти для її використання, шаблонізацію представлення та інші допоміжні функції.

PHP − це скриптова мова програмування для розробки веб додатків або консольних додатків з використанням з консолі серверу.

CMS яка буде розроблена у ході виконання буде мати MVC структуру сайту як найпоширенішу, оскільки розробка йде на базі фреймворку Laravel.

Ця структура представляє собою зв'язок рівня Модель − Представлення − Контролер, де модель займається вибіркою даних та їх оновленням та зв'язками, контролер займається маршрутизацією та обробкою даних, а представлення відображає ці дані користувачу або приймає їх.

1.2. Інструменти для розробки веб-сайтів

Існує безліч різних інструментів для розробки веб-сайтів, що допомагають скоротити час на його розробку. Інструменти мають різний рівень складності для розробника тому в більшості випадків розробник спеціалізується на декількох інструментах. Серед таких інструментів можна виділити фреймворки, модулі, плагіни, CMS.

Фреймворк − це програмне забезпечення що реалізує структуру майбутнього продукту та дає базові інструменти та функції для його подальшої розробки. Найпопулярніші фреймворки на базі мови програмування PHP це Laravel, Symfony, Codeigniter. Всі вони продовжують розробку та мають великий рівень використання серед веб додатків.

Модулі та плагіни − це пакетне програмне забезпечення для мови програмування або фреймворку що вирішує деякі бізнес задачі розробника, наприклад модуль Imagick що вирішує проблему роботи з зображеннями в PHP та Laravel, оскільки реалізує функції для їх обробки. Другий приклад це модуль для MySQL повнотекстового пошуку Sphinx.

1.3 Огляд CMS для створення інтернет магазинів та блогів

У більшості всіх систем управління контентом розробник або менеджер буде керувати сайтом за допомогою адміністративної панелі сайту яку реалізує сама CMS. CMS також реалізує базову структуру бази даних та сайту, можливість створення основних сутностей, які в більшості випадках являються сутностями товарів, постів та суміжними з ними таких як замовлення або коментарі сутностей за допомогою адміністративної панелі. Види цих сутностей міняються в залежності від потреб та самої CMS.

Для стилізації сайту під замовника в таких системах зазвичай використовується система тем з вже готових наборів які можна регулювати через налаштування сайту, але дозволяється і самостійна стилізація за допомогою розробника, оскільки доступ до файлів систем завжди відкритий. Всі вони використовують шаблонізатори що дозволяють спростити працю з структурою сторінок та стилями.

Шаблонізатори по суті представляють собою компілятор, що компілює структуру шаблону в html з php. Приклади таких шаблонізаторів це Blade, Twig, TPL.

Також серед CMS існують системи які наслідують принцип no-code, тобто замовник не використовує код при стилізації або розробки сайту, а користується тільки адміністративною панеллю де реалізовані конструктори сторінок і користувач лише їх позиціонує та стилізує за допомогою функціоналу.

Прикладом таких CMS є Weblium − це система управління контентом що розроблена в Україні та являє собою конструктор сторінок, можливість реалізації інтернет магазину, системи роботи з клієнтами, також має функціонал управління пошуковими налаштуваннями та оптимізацією в пошуку, а також налаштування маркетингу та таргетованої реклами. Можливості даної CMS показано на рисунку 1.1.

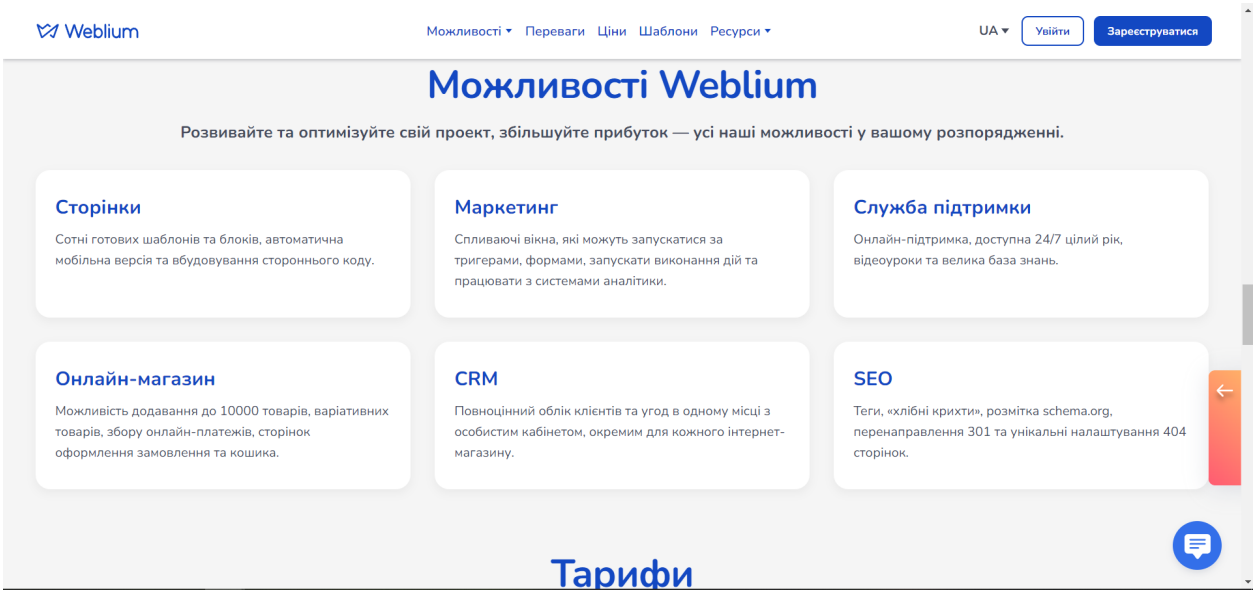

Рисунок 1.1 – Зображення можливостей CMS Wblium [3]

Функціонально всі CMS надають функціонал для розробки мінімального інтернет магазину або блогу, а також дає можливість встановлення плагінів які доповнюють цей функціонал, наприклад фільтри для товарів або атрибути товарів, розумний пошук. Весь функціонал завжди представлено в адміністративній панелі та розроблений таким чином що

менеджер який не є розробником може створити на базі CMS сайт з мінімальним функціоналом

Переваги CMS:

- простота встановлення;
- не потребує навичок програмування для мінімального функціоналу;
- порівняно невелика вартість у розробці;
- не має складнощів в управлінні не розробниками.

Недоліки CMS:

- більшість CMS мають погану архітектуру;
- більшість CMS не тримають високу завантаженість;
- слабка підтримка.

CMS можна також розділити по типу встановлення та підтримки розробником, а саме коробочні рішення, наприклад Shopify, рішення з відкритим кодом, або інакше open source, наприклад як Opencart.

Коробочні CMS – використовуються для розробки стороннім розробником є тільки фронтенд частина, без явної можливості доробки серверну частину, але все ж таки це можна вирішити або через проміжний API інтерфейс, або можна подати заявку і отримати можливість за допомогою додатків доповнювати бекенд частину.

CMS з відкритим кодом мають велику перевагу над коробочними рішеннями через те, що в них у розробника є можливість реалізувати будь-яку бізнес логіку для замовника. Також такі CMS часто приймають виправлення багів від ком’юніті, тобто від інших розробників які користуються цим продуктом, через систему контролю версій. Це частково гарантує більш швидке розвинення таких систем управління контентом. Прикладами таких систем є Opencart, Joomla, WordPress. Головним недоліком цих систем є те що майже в кожній з них зазвичай погана оптимізація готових запитів, погана архітектура та структура бази даних.

Фреймворки по своїй суті являє собою набір інструментів який прискорює розробку за допомогою готових рішень щодо банальних завдань таких як маршрутизація, з'єднання з базою даних та інші. Найяскравішими прикладами фреймворків на базі мови програмування PHP є Laravel, Symfony, Codeigniter, Zend, CakePHP. Все вони реалізують базові функції веб-додатку що дозволяє розробнику зосередити свою увагу на реалізації бізнес логіки.

Самописні системи управління контентом – це системи, які розроблені під конкретного замовника. У більшості випадків такі системи написані на базі фреймворків або самописного коду. Такі системи завжди мають закритий код, оскільки не розповсюджуються відкрито. Якщо вони побудовані на базі фреймворку то це гарантує певні плюси в підтримці коду та частковій оптимізації деяких функції бізнес логіки, та ці плюси не відносяться до самописних ресурсів, оскільки частіше за все такі сайти є важко підтримувані через відсутність будь-якої документації щодо коду який забезпечує роботу з базою даних, маршрутизацією, тощо.

Для розробки CMS довготривалої перспективи з гарною підтримкою будь-ким з розробників підходять найбільше саме фреймворки, вони зазвичай краще ніж готові CMS з особистою архітектурою. В моєму проекті було обрано фреймворк Laravel. Далі розглянемо фреймворки Laravel та Symfony.

1.3.1 Огляд Laravel

Laravel − це безкоштовний фреймворк з відкритим вихідним кодом що забезпечує розробку веб-додатків будь-якого рівня складності на базі архітектурної моделі Model-View-Controller. Випускається за ліцензією MIT. Наразі актуальна версія фреймворку 9.0 що вийшла в цьому році. Має наступні шаблони проектування:

- ActiveRecord;
- MVC;
- Singleton;

– Factory;
– Registry;
– Front Controller;
– Facade;
– Presenter;
– DI;
– Service Locator.

Також Laravel як фреймворк має наступні можливості:

– пакети − дозволяють створювати та підключати модулі у форматі Composer до програми на Laravel. Багато додаткових можливостей вже доступні у вигляді таких модулів;

– Eloquent ORM – реалізація шаблону проектування ActiveRecord на PHP. Дозволяє чітко визначити відносини між об'єктами бази даних. Стандартний для Laravel будівельник запитів Fluent підтримує ядро Eloquent;

– логіка програми – частина програми, що розробляється, оголошена або за допомогою контролерів, або маршрутів (функцій-замикань). Синтаксис оголошень схожий на синтаксис, який використовується у каркасі Sinatra;

– зворотна маршрутизація – пов'язує між собою посилання та маршрути, що генеруються додатком, дозволяючи змінювати останні з автоматичним оновленням пов'язаних посилань. Під час створення посилань за допомогою іменованих маршрутів Laravel автоматично генерує кінцеві URL-адреси;

– REST-контролери – додатковий шар для поділу логіки обробки GET- та POST-запитів HTTP;

– автозавантаження класів – механізм автоматичного завантаження класів PHP без необхідності підключати файли їх визначень у include. Завантаження на вимогу запобігає завантаженню непотрібних компонентів; завантажуються лише ті, які дійсно використовуються;

– упорядники уявлень – блоки коду, які виконуються при генерації уявлення;

– інверсія управління – дозволяє отримувати екземпляри об'єктів за принципом зворотного керування. Також може використовуватися для створення та отримання об'єктів-одинаків;

– міграції – система управління версіями для баз даних. Дозволяє пов'язувати зміни в коді програми зі змінами, які потрібно внести до структури бази даних, що спрощує розгортання та оновлення програми;

– юніт-тести – відіграє дуже велику роль в Laravel, який сам по собі містить велику кількість тестів для запобігання регресії;

– пагінація – спрощує генерацію сторінок, замінюючи різні способи вирішення цього завдання єдиним механізмом, вбудованим в Laravel;

– підтримка NoSQL систем управління базами даних та Redis;

– Безліч готових адміністративних панелей, шаблонів та CRUD;

– шаблонізатор Blade.

Серед користувачів Laravel є досить багато великих бізнес компаній, наприклад Disney, Twitch, Warner Brothers. The New York Times. Це вказано на їх сайті. Головна сторінка Laravel показана на рисунку 1.2. Великі користувачі Laravel показані на рисунку 1.3.

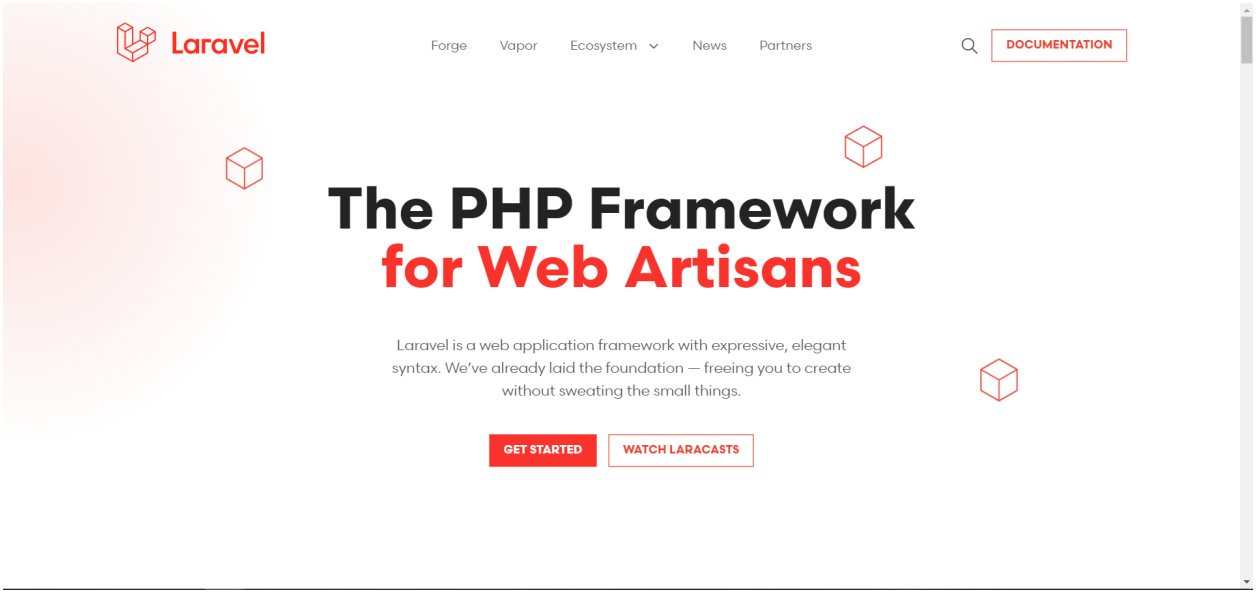

Рисунок 1.2 – Головна сторінка Laravel

Також одним з користувачем Laravel є українська компанія Evergreen що займається розробкою свого продукту на базі нейронної мережі по розпізнаванню, зрівнюванні обличь, а також знаходження фейкових фото та відео, розпізнаванню документів.

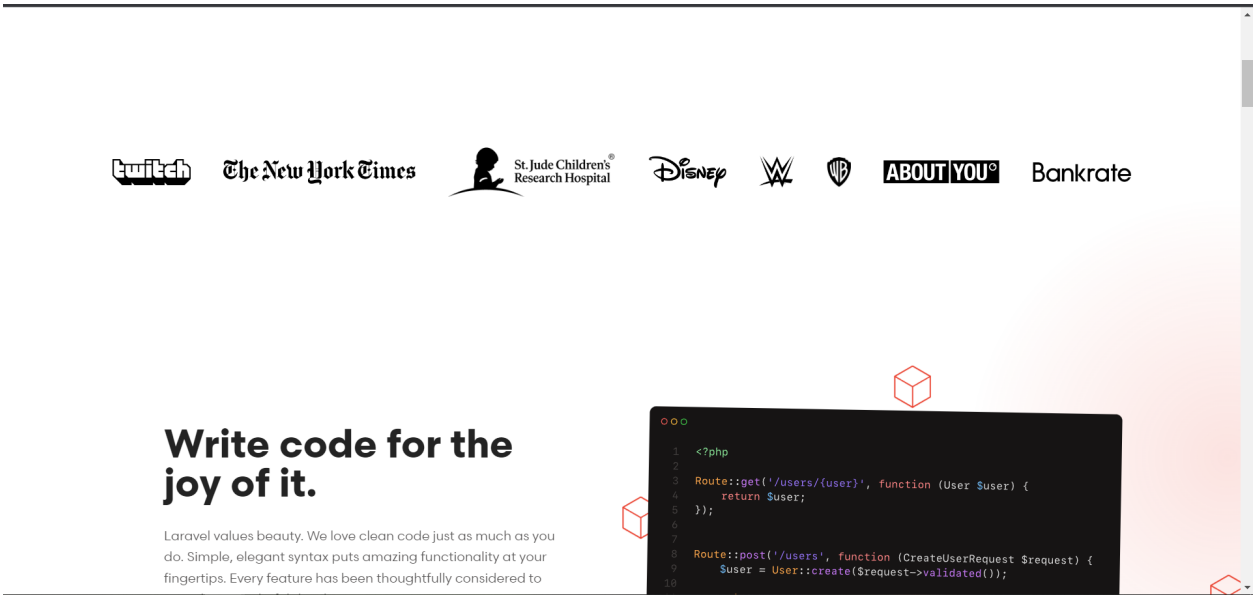

Рисунок 1.3 – Користувачі Laravel

Даний продукт має адміністративну та клієнтську частину розроблену саме на Laravel. Назва продукту FaceID. Візуальне представлення можна побачити на рисунках 1.4, 1.5 та 1.6.

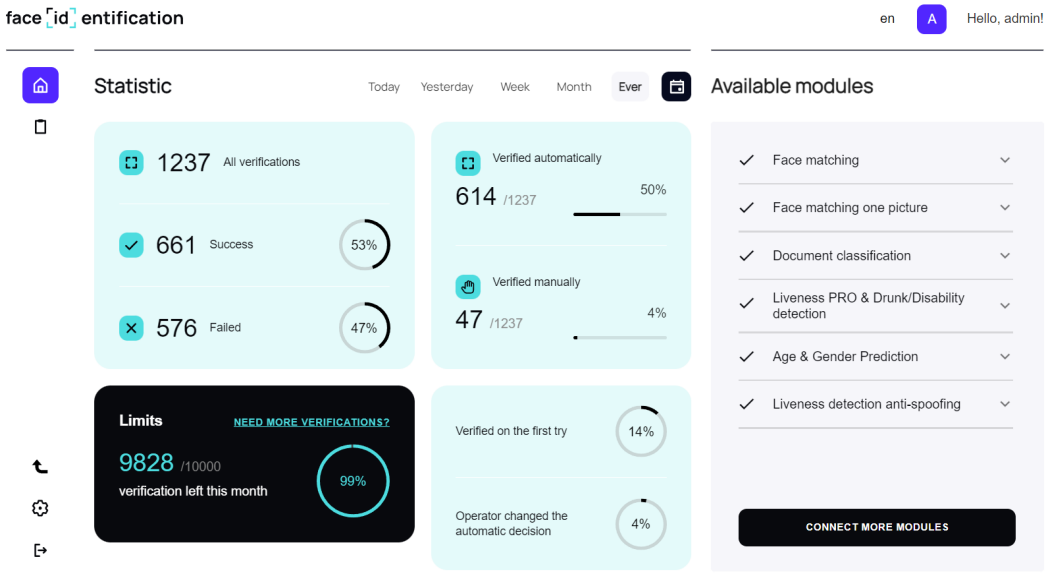

Рисунок 1.4 – Візуальне представлення продукту FaceID

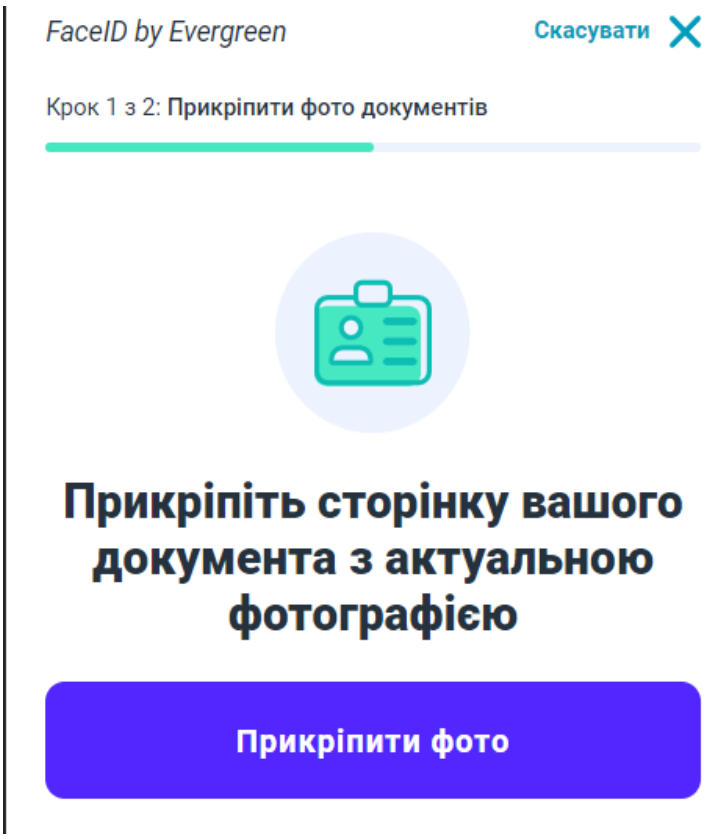

Рисунок 1.5 – Візуальне представлення продукту FaceID

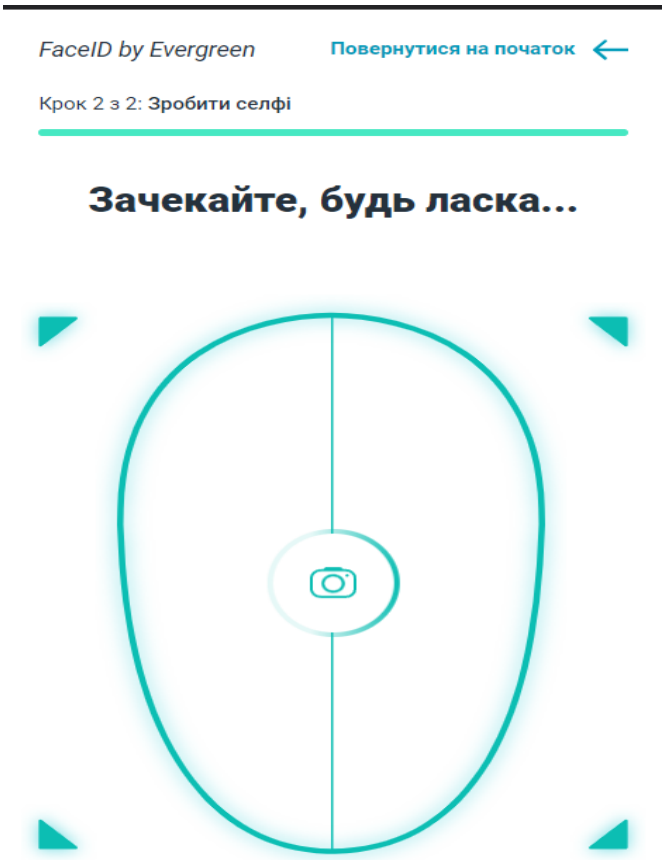

Рисунок 6 – Візуальне представлення продукту FaceID

Також Laravel має дуже багато пакетів з готовими рішеннями для деяких задач, наприклад:

– Laravel Breeze − це мінімальна, проста реалізація всіх функцій аутентифікації Laravel, включаючи вхід, реєстрацію, скидання пароля, перевірку електронної пошти та підтвердження пароля. Типовий шар перегляду Laravel Breeze складається з простих шаблонів Blade, оформлених за допомогою Tailwind CSS. Breeze є чудовою відправною точкою для створення нової програми Laravel, а також є чудовим вибором для проектів, які планують підняти свої шаблони Blade на новий рівень за допомогою Laravel Livewire;

– Eloquent − об’єктно-реляційний картограф, який робить взаємодію з базою даних приємною. При використанні Eloquent кожна таблиця бази даних має відповідну модель, яка використовується для взаємодії з цією таблицею. Окрім отримання записів із таблиці бази даних,

моделі Eloquent також дозволяють вставляти, оновлювати та видаляти записи з таблиці;

– Guzzle HTTP-клієнт − що дозволяє швидко робити вихідні HTTP-запити для зв'язку з іншими веб-додатками. Обгортка Laravel навколо Guzzle зосереджена на найпоширеніших випадках використання та чудовому досвіді розробника;

– PHPUnit − пакет що необхідний для розробки unit тестів для веб додатку. Входить із коробки, а файл phpunit.xml вже налаштовано для вашої програми. Фреймворк також постачається зі зручними допоміжними методами, які дозволяють чітко тестувати свої програми;

– Artisan − це інтерфейс командного рядка, що входить до складу Laravel. Artisan існує в корені вашої програми як сценарій artisan і надає ряд корисних команд, які можуть допомогти вам під час створення програми.

Щоб допомогти розробнику дізнатися більше про те, що відбувається у програмі, Laravel надає надійні служби журналів, які дозволяють реєструвати повідомлення у файли, журнал системних помилок і навіть у Slack, щоб сповістити всю команду розробників веб додатку.

Журнал Laravel заснований на каналах. Кожен канал представляє певний спосіб запису інформації журналу. Наприклад, один канал записує файли журналу в один файл журналу, а слабкий канал надсилає повідомлення журналу в Slack. Повідомлення журналу можуть бути записані на кілька каналів залежно від їх серйозності.

Під капотом Laravel використовує бібліотеку Monolog, яка забезпечує підтримку різноманітних потужних обробників журналів. Laravel спрощує налаштування цих обробників, дозволяючи змішувати та поєднувати їх, щоб налаштувати обробку журналів додатку.

1.3.2 Огляд Symfony

Symfony − це безкоштовний фреймворк, написаний на PHP. Symfony пропонує рішення та керування веб-додатками, що дозволяє легко вирішувати

рутинні завдання веб-програміста. Працює лише з версії PHP 5 і вище. Має зворотний зв'язок з базою даних, наприклад MySQL, PostgreSQL, SQLite або будь-яка інша PDO-сумісна СУБД. Інформація про реляційну базу даних у проекті повинна бути зв’язана з об'єктною моделлю. Це можна зробити за допомогою інструменту ORM. Symfony надає можливість обрати один з таких, саме:

– Propel − це об’єктно-реляційне відображення (ORM) з відкритим вихідним кодом для баз даних SQL у PHP 5.5. Він дозволяє отримати доступ до вашої бази даних за допомогою набору об’єктів, надаючи простий API для зберігання та отримання даних. На додаток до своїх можливостей ORM він забезпечує створення запитів, міграцію схеми бази даних, зворотне проектування існуючої бази даних та багато іншого.

– Doctrine − це бібліотека що поєднує в собі кілька бібліотек PHP, які в основному зосереджені на зберіганні бази даних та відображенні об’єктів. Основними проектами є Object Relational Mapper (ORM) і рівень абстракції бази даних (DBAL), на яких він побудований.

Symfony є безкоштовним і взаємодіє за ліцензією MIT. Актуальна версія Symfony 6.0. На базі даного фреймворку написані такі відомі CMS як Drupal та CMS eZ Publish, а саме написані вони на базі Symfony 2.0. Головна сторінка Symfony зображена на рисунку 1.7.

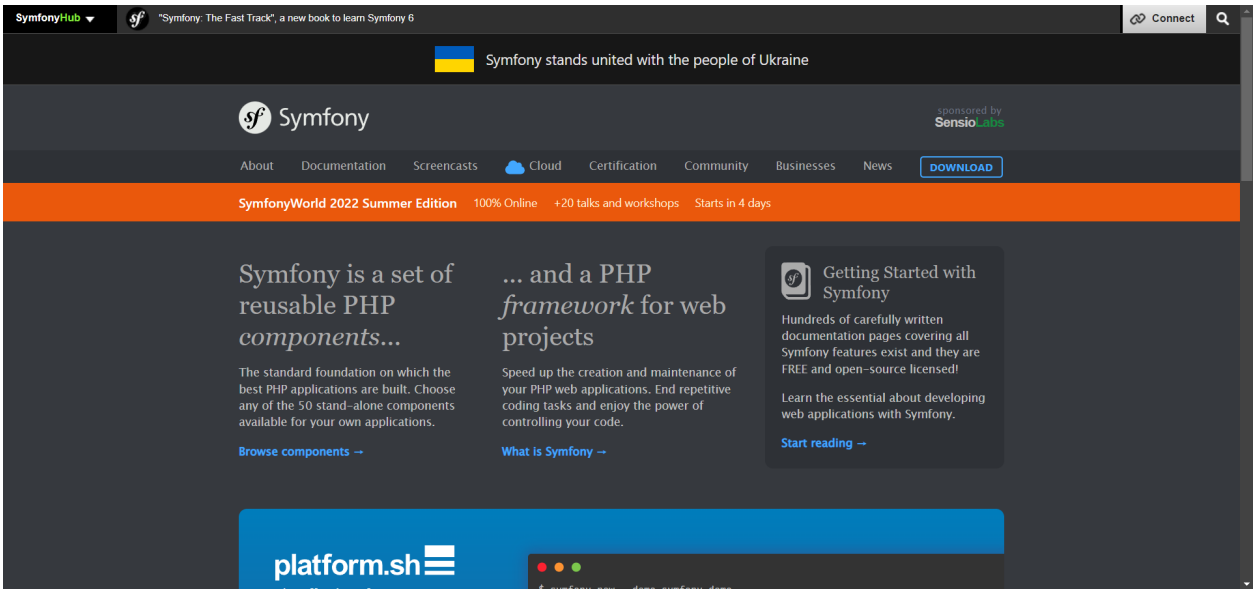

Рисунок 1.7 – Головна сторінка Symfony

Symfony легко інтегрується з Monolog, найпопулярнішою бібліотекою журналів PHP, для створення та зберігання повідомлень журналу в різних місцях та ініціювання різноманітних дій. Наприклад, за допомогою Monolog ви можете налаштувати реєстратор на різні дії на основі рівня повідомлення.

Створення та обробка HTML-форм є важким і повторюваним. Розробнику потрібно постійно мати справу з відтворенням полів форми HTML, перевіркою надісланих даних, відображенням даних форми в об’єкти та багато іншого. Symfony містить потужну бібліотеку форм, яка надає всі ці функції та багато іншого для справді складних сценаріїв.

На базі Symfony або ж її бібліотек побудовано більшість інших фреймфорків та мікрофреймворків для мови програмування PHP, наприклад Laravel в перших версіях була по суті обгорткою над Symfony що спрощувала розробку, в подальшому Laravel стала більш унікальною, але все ж більшість необхідних для роботи пакетів використовує саме з Symfony.

Завдяки тому, що на Symfony дуже величезна кількість пакетів та бібліотек, більша частина розробки може лягти лише на конфігурацію цих пакетів. Наприклад, для того щоб створити аутентифікацію користувача з різними її видами, захистом та обмеженнями доступу необхідно лише

зконфігурувати один конфігураційний файл. У якості мови для таких файлів використовується не зовсім звичний для інших фреймворків Yaml − це зручна для людини мова серіалізації даних для всіх мов програмування.

Symfony проста у використанні завдяки методу програмування Ruby-On-Rails, чистому дизайну і гарному коду. Symfony пропонує помічники Ajax, плагіни та інтерфейс адміністратора, що робить програмування повних додатків по-справжньому простим. Розробники можуть зосередитися на прикладній логіці, не витрачаючи часу на створення нескінченних файлів конфігурації XML. Symfony може використовуватися для створення високонадійних надійних додатків для підприємств, оскільки він допомагає розробникам тестувати, налагоджувати і документувати проекти, надаючи їм вичерпний контроль над налаштуваннями динамічних бібліотек. Symfony виконує протокол проектування Model-View-Controller, який відокремлює бізнес-логіку від рівня уявлення.

1.4 Постановка задачі

Метою даної кваліфікаційної роботи є розробка системи управління контентом на базі фреймворку Laravel для створення інтернет магазинів чи блогів. Тобто кінцевий продукт має мати адміністративну панель для адміністрування даних сутностей, а також їх уявлення на клієнтській стороні. Продукт повинен мати API для створення сутностей, початкову структуру бази даних, міграції для створення необхідних таблиць в базі даних та команду для встановлення, а також команду для автоматичного створення даних, таких як користувач адміністратор, тощо.

У процесі розробки необхідно реалізувати наступні задачі:

– Зробити огляд існуючих допоміжних інструментів для розробки та обрати потрібні;

– розробити інтерфейс за принципом Usability;

– спроектувати архітектуру бази даних;

- спроектувати архітектуру коду;
- почати розробку функціоналу згідно вищезазначених пунктів;
- зробити висновки по роботі.

У разі успішного створення демо-версії даного продукту буде можливість продовжити його розвиток та на базі даного продукту створювати на замовлення різноманітні інтернет магазини та блоги з великим навантаженням, у розробників буде доступ до зрозумілою документації та зрозумілий інтерфейс розробки за рахунок базування проекту на фреймворку Laravel.

2. МЕТОДИКА ВИРІШЕННЯ ПОСТАВЛЕНИХ ЗАДАЧ

2.1 Проектування бази даних та архітектури CMS

Діаграма зв’язків між об’єктами (ER) − це тип блок-схеми, яка ілюструє, як «суб’єкти», такі як люди, об’єкти або поняття, пов’язані один з одним у системі. Діаграми ER найчастіше використовуються для проектування або налагодження реляційних баз даних у сферах програмної інженерії, бізнес-інформаційних систем, освіти та досліджень. Також відомі як ERD або моделі ER, вони використовують певний набір символів, таких як прямокутники, ромби, овали та сполучні лінії, щоб відобразити взаємозв’язок сутностей, відносин та їхніх атрибутів. Вони відображають граматичну структуру з сутностями як іменники, а відносини як дієслова.

База даних нашої CMS буде мати в основі наступні сутності:

– User − сутність що представляє собою користувача та містить дані про нього, а саме ім’я, email, пароль в зашифрованому вигляді, токен доступу до API інтерфейсу, ідентифікатор групи користувача та дати створення, а також оновлення даних користувача;

– Group − сутність що відображає групи користувачів, таких як наприклад покупець та адміністратор, та розділяє користувачів по доступу до сторінок та сутностей;

– Product − сутність що відображає товар для продажу на сайті, якщо створюється інтернет магазин;

– Post − сутність що відображає новини або пости для показу відвідувачам сайту, якщо сайт створюється як блог чи на сайті планується додатково блог;

– Category − спільна сутність для сутностей Product и Post, що представляє собою категорію товару чи новини для їх групування чи сортування по даній сутності. Зв’язок с сутністю відбувається за допомогою ідентифікатору сутності;

– Comment − спільна сутність що представляє собою сутність для коментарів до Product або Post. Має універсальне призначення. Зв’язок з сутністю відбувається за допомогою ідентифікатору сутності, що з нею зв’язується;

– Order − сутність що відображає собою замовлення товару користувачем на сайті, а саме містить суму товарів та інформацію про користувача;

– OrderItem − сутність, що містить у собі інформацію про товари сутності Order, а саме їх ідентифікатор, кількість, ціну та дати оновлення або додавання. Зв’язок с сутностями Order та Product відбувається за допомогою ідентифікаторів цих сутностей;

– Setting − сутність що представляє собою налаштування CMS та містить скорочену назву налаштування та значення у вигляді серіалізованого рядку JSON;

– Migration − сутність що відображає пройдені міграції бази даних. Необхідна для уникнення повторного запуску міграцій що вже були виконані, щоб не створювати помилки.

У вигляді ER діаграми наша база даних буде мати вигляд як показано на рисунку 2.1. Також необхідно створити ERD кодової частини нашого додатку для проектування і написання коду згідно цієї діаграми. У вигляді ER діаграми код буде мати додаток як показано на рисунку 2.2.

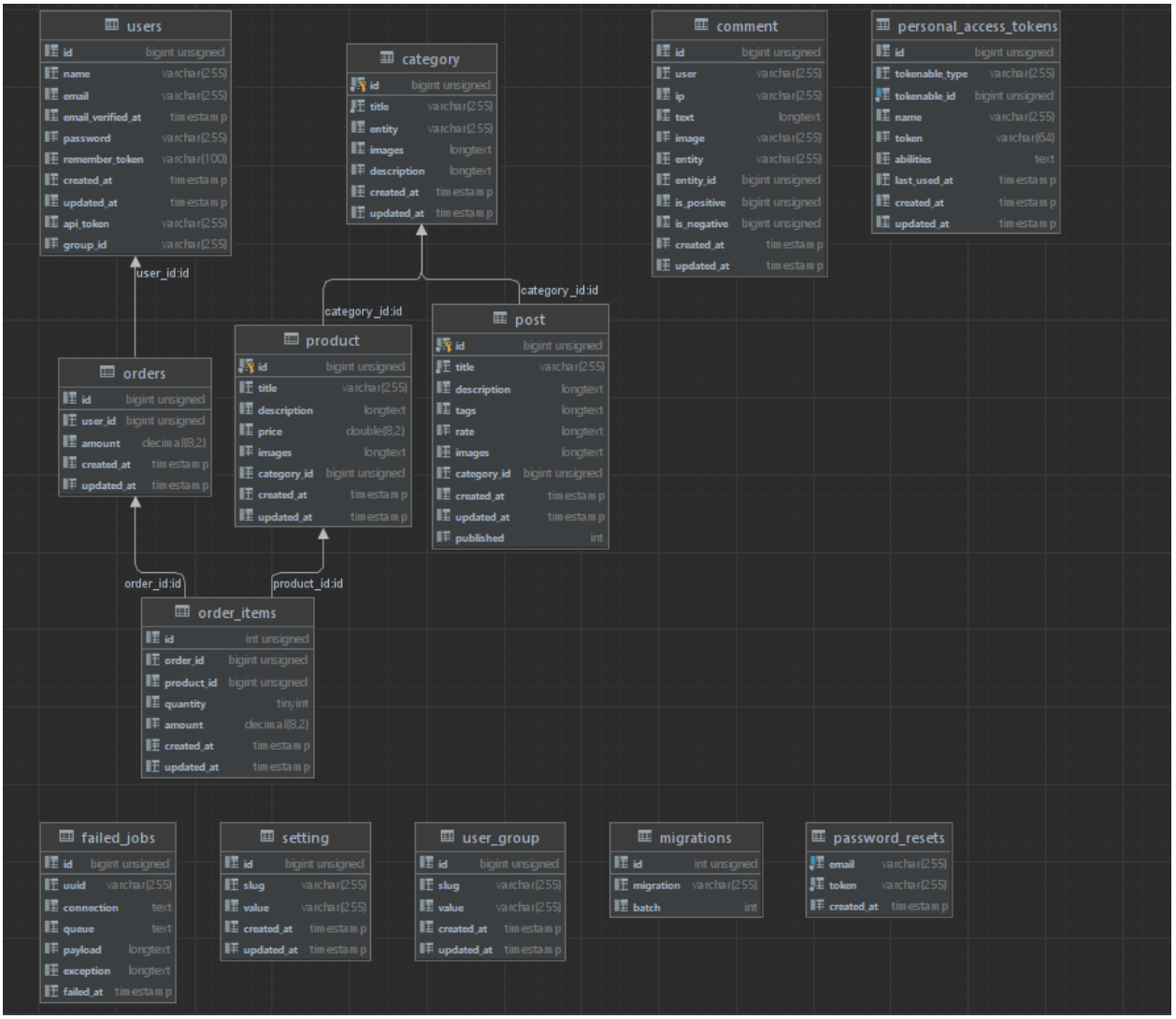

Рисунок 2.1 – ER діаграма бази даних CMS

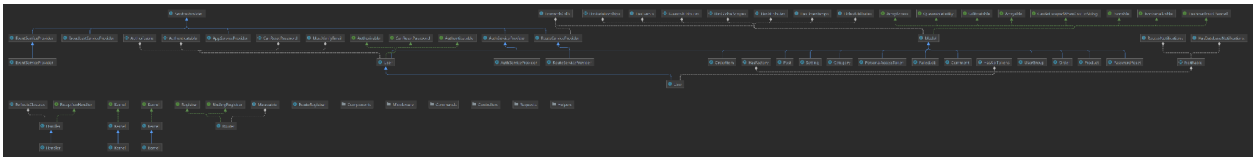

Рисунок 2.2 – ER діаграма додатку CMS

2.2 Розробка CMS

У розробці даної CMS необхідно виділити декілька етапі, а саме:

– створення міграцій для базової моделі бази даних;

– створення моделей для сутностей бази даних;

– створення CRUD контролерів для сутностей;

– створення контролерів що будуть відповідати за відображення сторінок які не відносяться до сутностей;

– створення класів валідації даних що надходять на інтерфейс;

– створення Middleware сутностей для певніх дій між виконанням користувачем дії та початком роботи контролера;

– створення клієнтської частини адміністративної панелі;

– створення клієнтської частини для користувачів сайту;

– дизайн.

Створення міграцій для базової моделі бази даних, моделей для сутностей баз даних та контролерів буде відбуватися за допомогою використання Artisan інтерфейсу фреймфорку, а саме за допомогою команди генерації php artisan make:model ModelName.

Оскільки моделі та контролери створюються лише з базовою структурою та в базової директорії то необхідно буде виконати також php artisan code:models для моделей та генерації їх остаточної структури. Контролери необхідно рознеси по відповідним їм директоріям Client та Admin згідно до розділення їх ролі. Кінцева структура контролерів зображена на рисунку 2.3.

Кожна з сутностей потребує створення окремого класу валідації вхідних даних який буде наслідувати клас Request та перевизначати його функції для валідації даних.

Створення адміністративної панелі проходить на базі мінімалистичної стилістики Laravel Breeze та стилів TailWindCss. Для всіх сутностей представлення в адміністративній панелі буде шаблоним та представлятиме собою структуру з блоку меню, блоку створення та блоку контенту. Приклад зображено на рисунку 2.4.

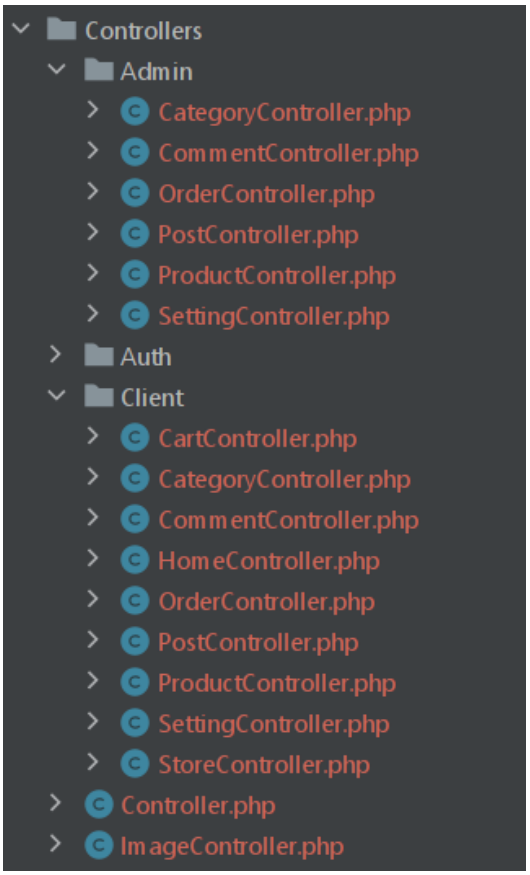

Рисунок 2.3 – Структура контролерів CMS

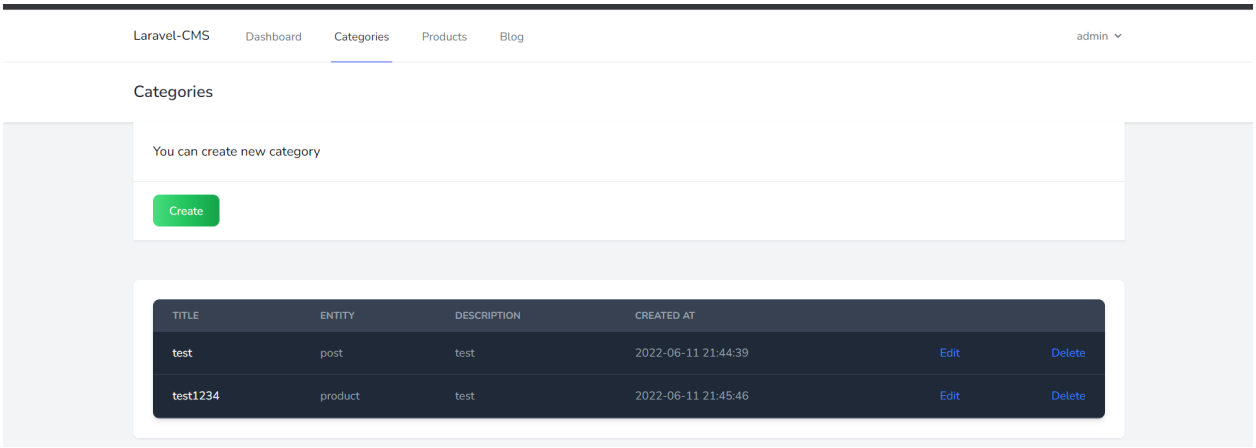

Рисунок 2.4 – Макет адміністративної частини CMS

Макет сайту − це попередній рисунок, який визначає зовнішній вигляд сайту та його функціонал. Дизайн сайту повинен бути виконаний кращих рішеннях UI/UX дизайну, в даному випадку в мінімалістичному та зрозумілому кінцевому користувачу. Користувач не повинен заходити в глухий кут, блукати по сайту і вирішувати завдання. Якщо інтерфейс занадто

складний і незрозумілий, більшість користувачів більше не повертатиметься до ресурсу.

Усі розділи та сторінки повинні мати єдину логіку оформлення − кількість стовпців у макеті, розташування повторюваних елементів, заголовків, шрифтів.

Архітектура сайту часто залишається на розробку дизайнера, особливо якщо він має досвід UI/UX. Його основне завдання − побудувати структуру сторінок і правильно їх посилати, щоб користувач міг легко переміщатися по сайту. Це полегшується якщо:

– існує кілька шляхів, що ведуть до однієї сторінки.;

– існує дизайн, який допоможе вам зрозуміти, що елементи інтерактивні і їх можна натиснути;

– наявне використання логічних піктограм і чітких закликів до дії;

– дизайн сайту відображає його тематику.

Дизайн клієнтської частини сайту буде виконано за допомогою Bootstrap. Приклад дизайну зображено на рисунку 2.5.

Рисунок 2.5 – Макет клієнтської частини CMS

Це потужний, розширюваний і багатофункціональний набір інструментів інтерфейсу. Дозволяє створювати та налаштовувати за допомогою Sass компоненти, використовувати попередньо вбудовану систему та компоненти та втілювати проекти в життя за допомогою потужних плагінів JavaScript. Сам дизайн буде також виконаний в мінімалістичному стилі.

3. ПРАКТИЧНА РЕАЛІЗАЦІЯ

3.1 Архітектура CMS

CMS буде мати дві частини представлення, а саме адміністративну та користувацьку.

Адміністративна частина сайту буде давати користувачеві з групою адміністратор створювати, редагувати та видаляти такі сутності як налаштування, товари, пости, категорії, коментарі, замовлення. А також переглядати вже існуючі елементи цих сутностей.

Клієнтська частина сайту буде надавати користувачеві з групою покупець або не зареєстрованому користувачеві перегляди такі сутності як товари та пости, а також можливість зареєструватися та виконати аутентифікацію в системі, після чого користувачеві буде надано можливість придбати товари. Адміністративна частина буде містити наступні сторінки:

– dashboard – відображає основну інформацію про середовище, останні замовлення, коментарі;
– сторінка перегляду існуючих категорій у вигляді таблиці;
– сторінка редагування існуючих категорій;
– сторінка створення категорій;
– сторінка перегляду існуючих товарів у вигляді таблиці;
– сторінка редагування існуючих товарів;
– сторінка створення товарів;
– сторінка перегляду існуючих постів у вигляді таблиці;
– сторінка редагування існуючих постів;
– сторінка створення постів;
– сторінка перегляду існуючих замовлень у вигляді таблиці;
– сторінка редагування існуючих замовлень;
– сторінка створення замовлень в адміністративній панелі;

– сторінка перегляду коментарів до товарів з можливістю видалення;

– сторінка перегляду існуючих налаштувань у вигляді таблиці;

– сторінка створення нових налаштувань;

– сторінка редагування існуючих налаштувань;

– сторінка аутентифікації.

Клієнтська частина буде містити наступні сторінки:

– головна сторінка що відображає останні товари та пости з можливістю додавання до корзини та переходу до всіх товарів або постів, а також переходу в сам товар або пост;

– сторінка перегляду постів;

– сторінка перегляду товарів;

– сторінка перегляду посту;

– сторінка перегляду товару;

– сторінка реєстрації;

– сторінка замовлення з можливістю його завершення;

– сторінка аутентифікації.

Архітектура серверної частини буде представляти собою модифіковану версію Model View Controller, а саме також будуть наявні Helpers, Middleware, Request сутності для валідації, модифікації, перевірки даних та виконання повторюваного коду.

3.2 Програмна реалізація

Для початку реалізації проекту необхідно налаштувати середовище для розробки. А саме потрібно встановити пакети необхідні для розробки. Серед таких пакетів та додатків MySQL, Nginx, PHP 8.x, Node 14.x, ImageMagick, Composer. Вся розробка буде проходити на базі WSL на яку встановлено Unix-подібну операційну систему Ubuntu.

MySQL – це безкоштовна система управління реляційною базою даних.

Nginx – веб-сервер і поштовий проксі-сервер, що працює на Unix-подібних операційних системах.

Node – програмна платформа, заснована на движку V8, що перетворює JavaScript з вузькоспеціалізованої мови в мову загального призначення.

ImageMagick – набір програм для читання та редагування файлів безлічі графічних форматів.

Composer – це пакетний менеджер рівня додатків для мови програмування PHP, який надає засоби управління залежностями в PHP-додатку.

WSL – шар сумісності для запуску Linux-програм в операційній системі Windows 10.

Також необхідно ознайомитись з такими поняттями як сервер та домен.

Сервер – це спеціальне апаратне забезпечення, спеціалізоване для виконання на ньому певного сервісного програмного забезпечення і зберігання інформації. Сервери бувають фізичними та віртуальними [14].

Доменне ім'я − символьне ім'я, що служить для ідентифікації областей, які є одиницями адміністративної автономії в мережі Інтернет, у складі такої області, що стоїть за ієрархією. Кожна з таких областей називається доменом.

В випадку локальної розробки домен та окремий виділений сервер непотрібні, оскільки сервером буде виступати наш комп’ютер та Nginx в локальній мережі, а домен буде встановлено як localhost або ж 127.0.0.1. Також є можливість запуску проекту для розробки за допомогою серверу вбудованого в мову програмування PHP, а саме за допомогою допоміжної команди Artisan Laravel php artisan serve.

Ця команда запустить веб сервер з логуванням доступу до ресурсів сайту та помилок нашого додатку на базі localhost та порту 8000. Важливо те що дану функцію з зображень безпеки заборонено використовувати на сайті який знаходиться в доступі, а точніше має доступ в інтернет та знаходиться в середовищі production. Запуск веб серверу продемонстровано на рисунку 3.1.

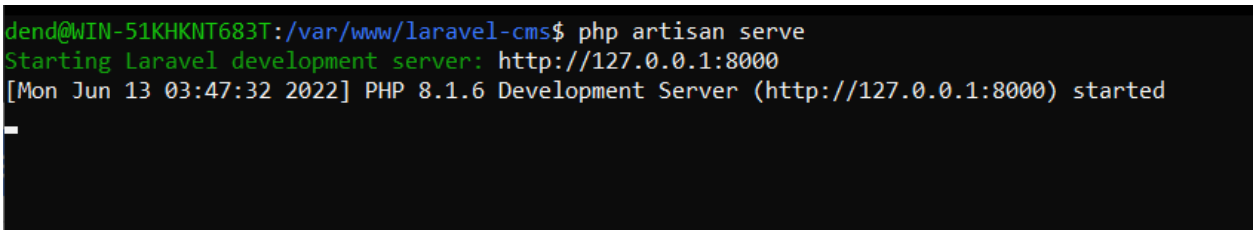

Рисунок 3.1 – Запуск локального веб серверу

Далі необхідно встановити всі пакети Composer за допомогою команди composer install -vvv, де атрибут -vvv буде відображати хід виконання встановлення, оскільки іноді через конфлікти пакетів встановлення може тривати довго і користувач цього не побачить. Встановлення пакетів Composer продемонстровано на рисунку 3.2.

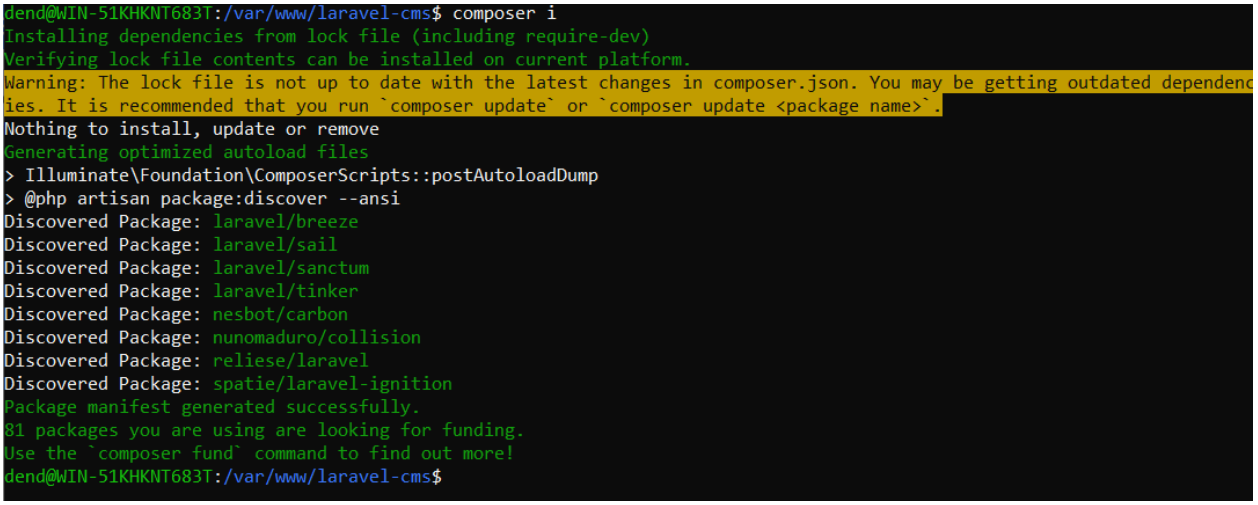

Рисунок 3.2 – Запуск встановлення пакетів Composer

Далі необхідно встановити всі пакети npm за допомогою команди npm install. Встановлення пакетів Npm продемонстровано на рисунку 3.3.

Встановлення пакетів Composer та Npm також мають поліфіл замість повної команди, а саме:

– composer i -vvv замість composer install -vvv;

– npm i замість npm install.

Наступним кроком буде налаштування файлу середовища .env, а саме доступів до бази даних. Продемонстровано на рисунку 3.4.

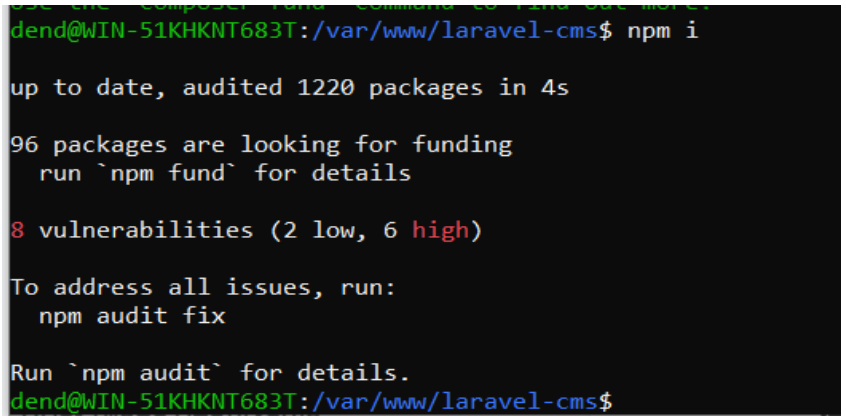

Рисунок 3.3 – Запуск встановлення пакетів Npm

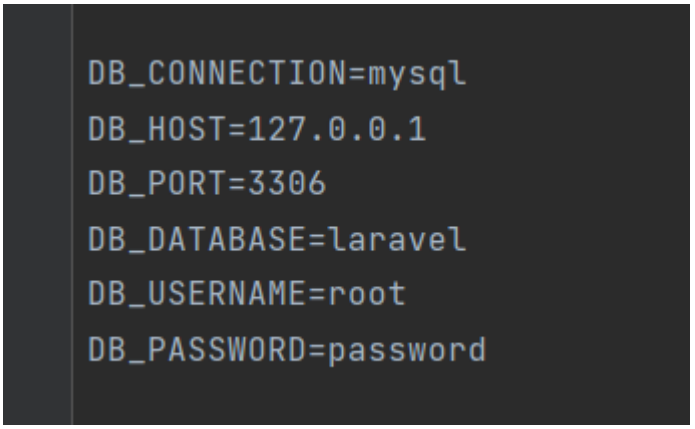

Рисунок 3.4 – Налаштування доступів до бази даних

Далі потрібно виконати запуск міграцій які раніше було створено за допомогою команди php artisan migrate:install, після чого виконати запуск внесення готових початкових даних за допомогою команди php artisan db:seed.

Після чого потрібно очистити кеш додатку за допомогою команди php artisan optimize. Після виконання всіх цих кроків додаток готовий до початку розробки, а саме безпосереднього написання коду та логіки додатку.

3.3 Використання готового продукту

В плані використання готового продукту додаток має розподіл на використання майбутнім розробником, використання користувачами клієнтської частини та використання адміністраторами.

Почнемо з використання адміністраторами. Серед можливостей такого користувача наступне:

– створення, видалення, редагування та перегляд товарів (зображено на рисунку Б.5 - Б.7);

– створення, видалення, редагування та перегляд категорій (зображено на рисунку Б.2 - Б.4);

– створення, видалення, редагування та перегляд постів (зображено на рисунку Б.8 - Б.10);

– створення, видалення, редагування та перегляд замовлень;

– видалення коментарів;

– аутентифікація (зображено на рисунку 3.5);

– створення, видалення, редагування та перегляд налаштувань системи.

Серед можливостей користувача клієнтської частини наступне:

– перегляд та замовлення товарів(зображено на рисунку 3.6);

– перегляд постів(зображено на рисунку 3.7);

– створення та перегляд замовлень(зображено на рисунку 3.8);

– створення, редагування, перегляд та видалення коментарів;

– аутентифікація;

– реєстрація(зображено на рисунку 3.9).

Серед можливостей майбутнього розробника на базі готової CMS майже будь-які, оскільки в даної системи наслідується принцип відкритості коду, що дозволить їй більш швидко розвиватись за рахунок користувачів. Даний розробник в праві дороблювати та переробляти будь-який функціонал системи.

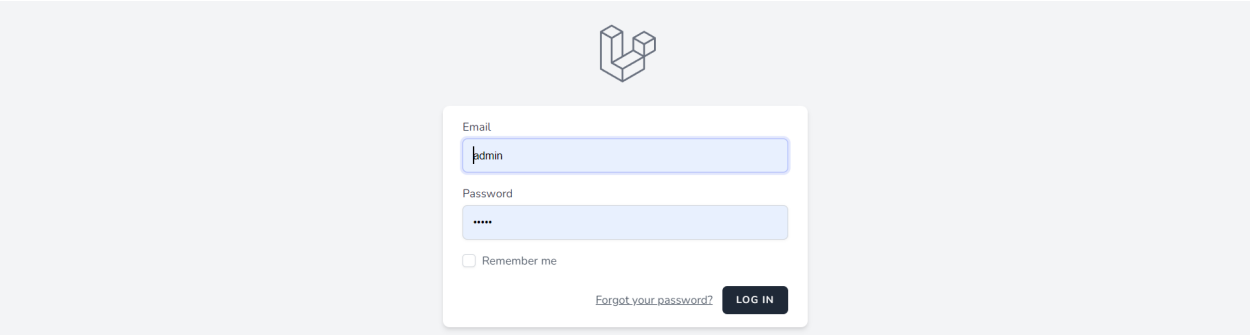

Рисунок 3.5 – Аутентифікація

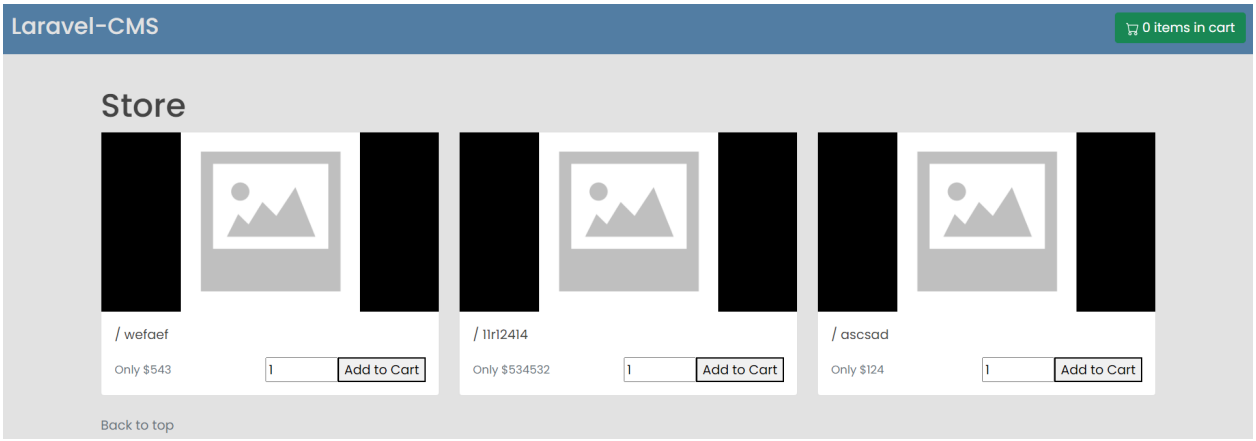

Рисунок 3.6 – Перегляд та замовлення товарів

Рисунок 3.7 – Перегляд постів

Рисунок 3.8 – Створення та перегляд замовлення

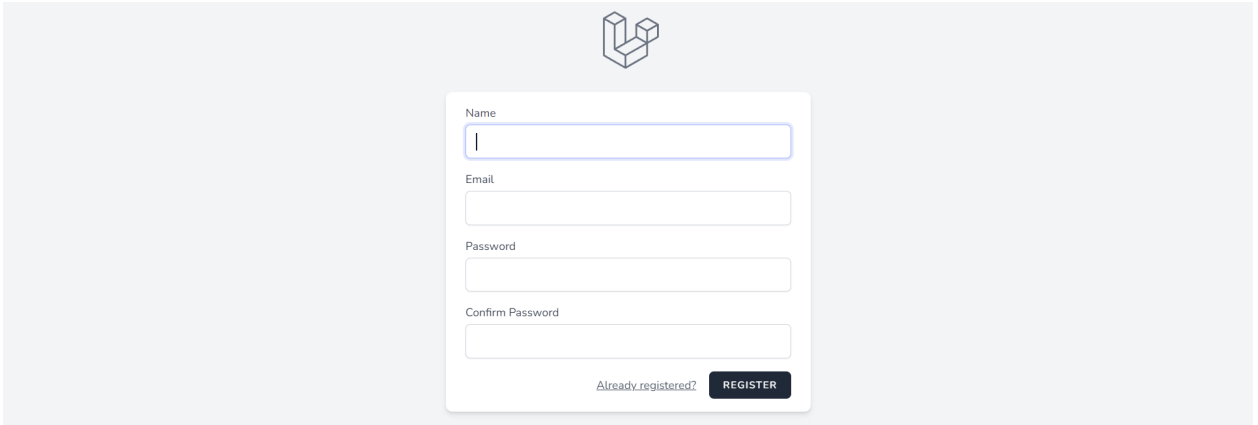

Рисунок 3.9 – Реєстрація

Огляд існуючого функціоналу завершено, далі підведемо підсумки у висновках.

ВИСНОВКИ

1. У ході виконання дипломної роботи було детально розібрано фреймворк Laravel, основні принципи розробки на ньому, а також аналоги даного фреймворку такі як Symfony.

2. Робота виконана на базі фреймворку Laravel для електронної комерції.

3. В процесі виконання дипломної роботи було розглянуто та вирішено:

- Розробка додатків на базі Laravel;
- ознайомлення з принципами проектування CMS;
- порівняння різних фреймворків;
- система управління контентом було запущено в розробку.

4. Сформульовано технічне завдання до додатку з описанням таких пунктів, як призначення CMS, мета розробки, аудиторія.

5. Корисність дана робота несе саме для розробників в сфері електронної комерції тим що в майбутньому спростить розробку блогів та інтернет магазинів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Фреймворк. [Електронний ресурс] – Режим доступу до ресурсу: https://laravel.com/docs/9.x/.

2. Laravel. [Електронний ресурс] – Режим доступу до ресурсу: https://laravel.com/docs/9.x/.

3. Symfony. [Електронний ресурс] – Режим доступу до ресурсу: https://symfony.com/doc/current/index.html.

4. YAML [Електронний ресурс] – Режим доступу до ресурсу: https://yaml.org/.

5. MySQL [Електронний ресурс] – Режим доступу до ресурсу: https://www.mysql.com/.

6. Composer [Електронний ресурс] – Режим доступу до ресурсу: https://getcomposer.org/.

7. Node [Електронний ресурс] – Режим доступу до ресурсу: https://nodejs.org/en/.

8. PHP [Електронний ресурс] – Режим доступу до ресурсу: php.net/manual/ru/intro-whatis.php.

9. Weblium [Електронний ресурс] – Режим доступу до ресурсу: https://ua.weblium.com/.

10. Nginx [Електронний ресурс] – Режим доступу до ресурсу: https://nginx.org/en/.

11. Bootstrap [Електронний ресурс] – Режим доступу до ресурсу: https://getbootstrap.com/.

12. ERD [Електронний ресурс] – Режим доступу до ресурсу: https://www.lucidchart.com/pages/er-diagrams.

13. Хостинг [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/%D0%A5%D0%BE%D1%81%D1%82%D0%B8%D0%BD%D0%B3.

14. Що таке сервер? Які види бувають? [Електронний ресурс] – Режим доступу до ресурсу: https://hyperhost.ua/uk/wiki/chto-takoe-server-kakie-vidy-byvayut.

ДОДАТОК А. ТЕХНІЧНЕ ЗАВДАННЯ

1 ПРИЗНАЧЕННЯ Й МЕТА СТВОРЕННЯ CMS

1.1 Призначення CMS

Розробка інтернет магазинів та блогів на базі підтримуємої та зрозумілої системи управління контентом.

1.2 Мета створення CMS

Мета створення CMS це спрощення розробки веб-сайтів таких як інтернет магазини та блоги. Оптимізація розробки за рахунок популярного фреймворку та створення готових рішень.

1.3 Цільова аудиторія

Цільова група даного продукту є розробники програмного забезпечення та веб додатків, а також клієнти яким необхідно мати інтернет магазин чи блок.

2 ВИМОГИ ДО CMS

2.1 Вимоги до структури та функціонування інформаційної системи

Система має мати структуру яка наслідує структура фреймворку Laravel та мати основні сутності для роботи як система управління контентом.

2.2 Вимоги до персоналу

Вимоги до персоналу мають в собі середній або високий рівень навичок програмування на мовах PHP, JavaScript. Розуміння принципів MVC та OOP, а також вміння працювати з фреймворком Laravel на середньому або високому рівнях.

2.3 Вимоги до стилізації CMS

Дизайн системи управління контентом Laravel-CMS має бути мінімалістичним та зрозумілим, а також наслідувати принципи дизайну Usability.

3 ОСНОВНІ ВИМОГИ

3.1 Структура системи управління контентом

У даній системі мають бути реалізовані компоненти управління основними сутностями в адміністративній панелі, а також уявлення цих сутностей на клієнтській частині сайту. До того ж мають бути реалізовані компоненти аутентифікації та реєстрації.

3.2 Навігація

Розроблена система управління контентом буде мати: адміністративну та клієнтську частини.

Клієнтська частина призначена для перегляду сутностей користувачем сайту та здійснення замовлень.

Адміністративна частина призначена для керування та обробки сутностей адміністратором даної системи.

Навігація всієї системи повинна мати повний CRUD, тобто маршрути до сутностей на читання, видалення, створення та редагування.

3.3 Вимоги до програмного забезпечення

Вимога до користувача системи тільки одна, а саме підтримка протоколу TLS 1.3. Серед вимог до розробників:

- MySQL;
- Nginx;
- PHP 8.x;
- Node 14.x;
- Composer 2.

3.4 Функціональні вимоги

Функціональні вимоги:

- можливість редагування, створення, видалення, перегляд товарів;
- можливість редагування, створення, видалення, перегляд постів;
- можливість додавання товарів до кошику;
- можливість виконати замовлення;
- можливість залишати коментарі.

3.5 Наповнення сайту

Наповнення сайту буде виконуватись безпосередньо кінцевими користувачами даної системи управління контентом або в тестових цілях за допомогою генерації фейкових даних.

ДОДАТОК Б. ЗОБРАЖЕННЯ ОСНОВНИХ СТОРІНОК САЙТУ

Головну сторінку адміністративної частини системи зображено на рисунку Б.1

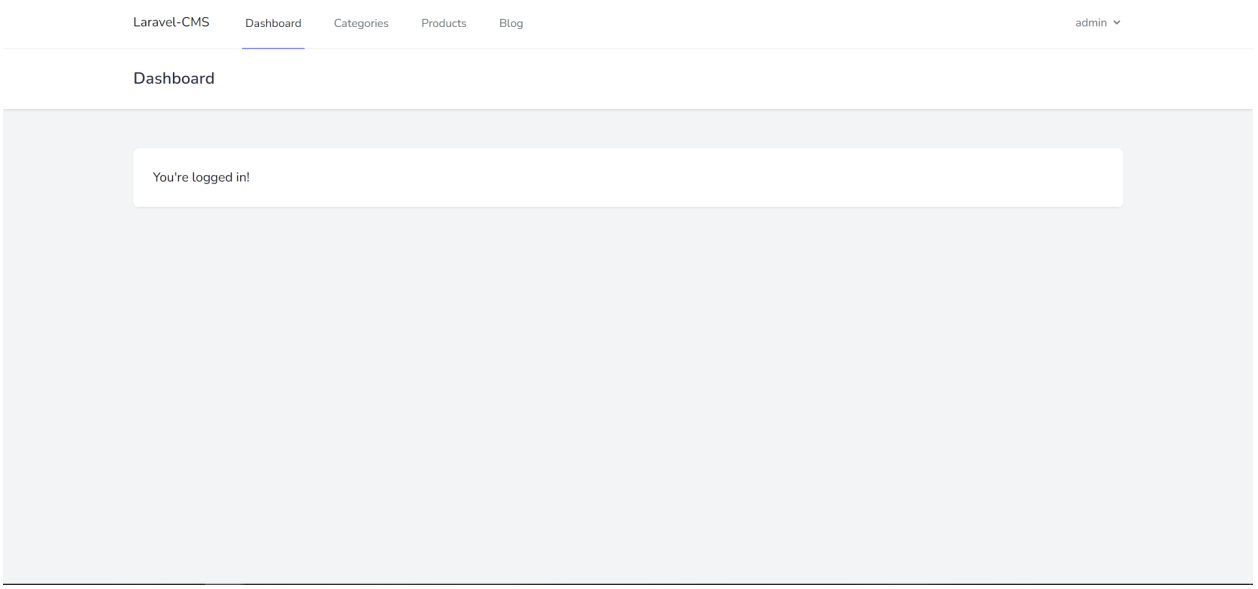

Рисунок Б.1 – Головна сторінка адміністративної частини системи

Сторінку адміністративної частини системи сутності Категорії зображено на рисунку Б.2

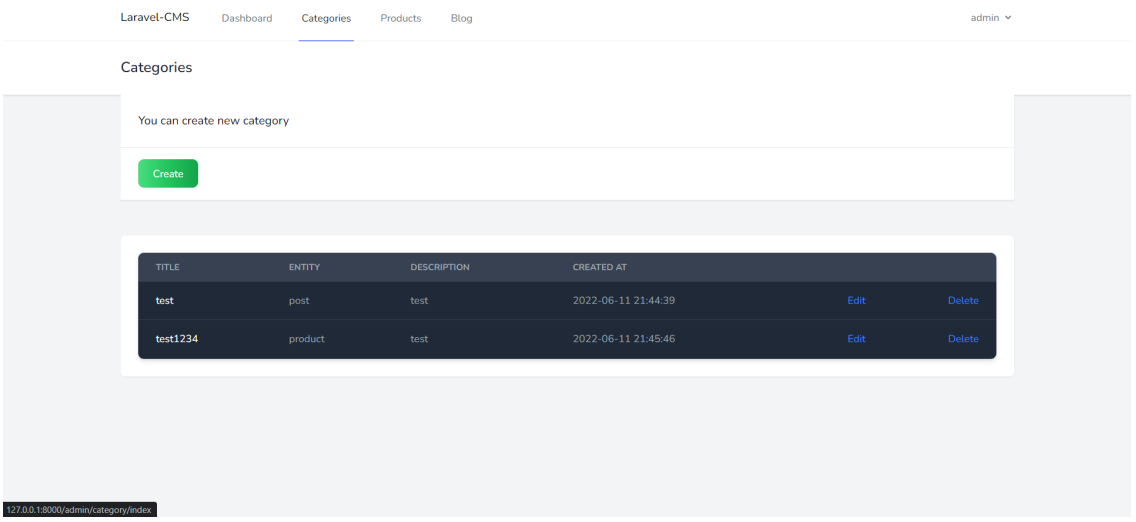

Рисунок Б.2 – Сторінка адміністративної частини системи сутності Категорії

Сторінку адміністративної частини системи створення сутності Категорії зображено на рисунку Б.3

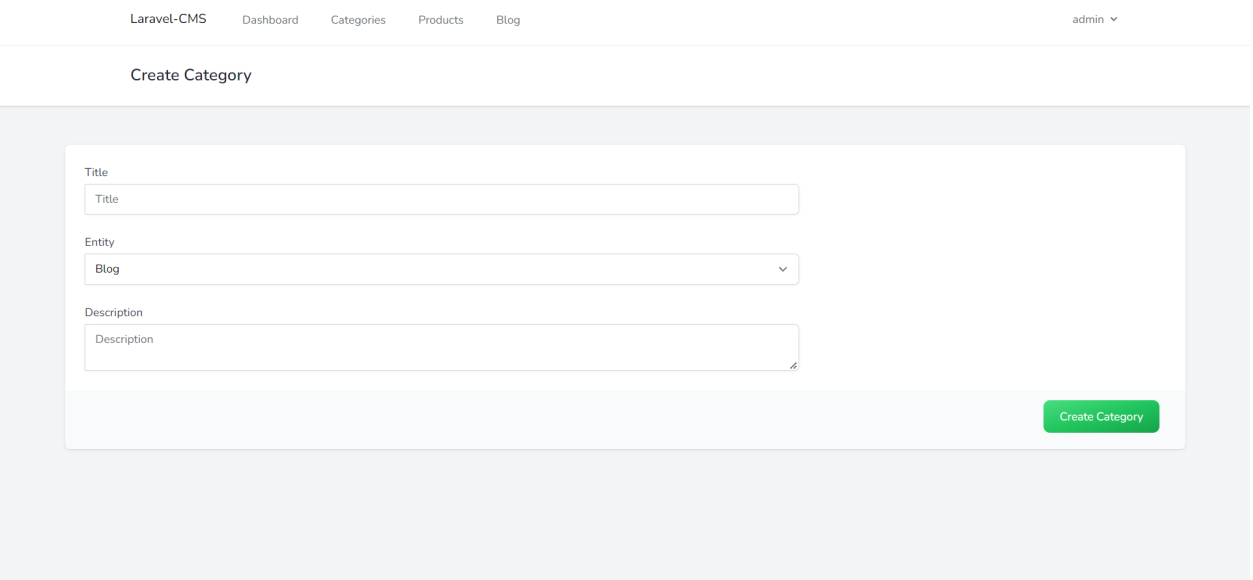

Рисунок Б.3 – Сторінка адміністративної частини системи створення сутності Категорії

Сторінку адміністративної частини системи редагування сутності Категорії зображено на рисунку Б.4

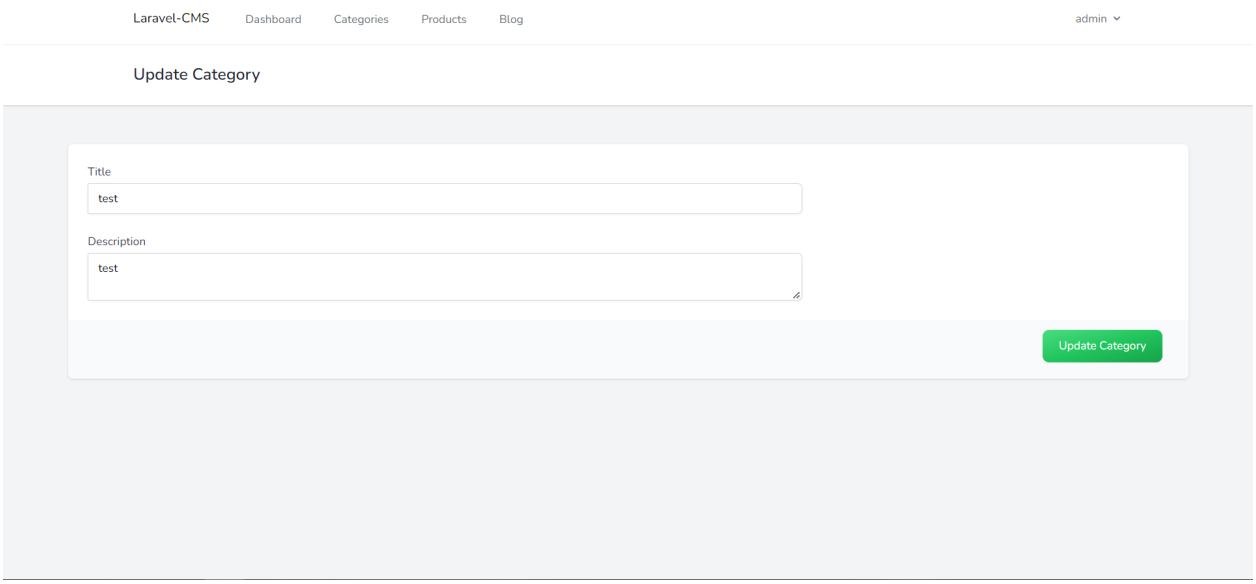

Рисунок Б.4 – Сторінка адміністративної частини системи редагування сутності Категорії

Сторінку адміністративної частини системи сутності Товари зображено на рисунку Б.5

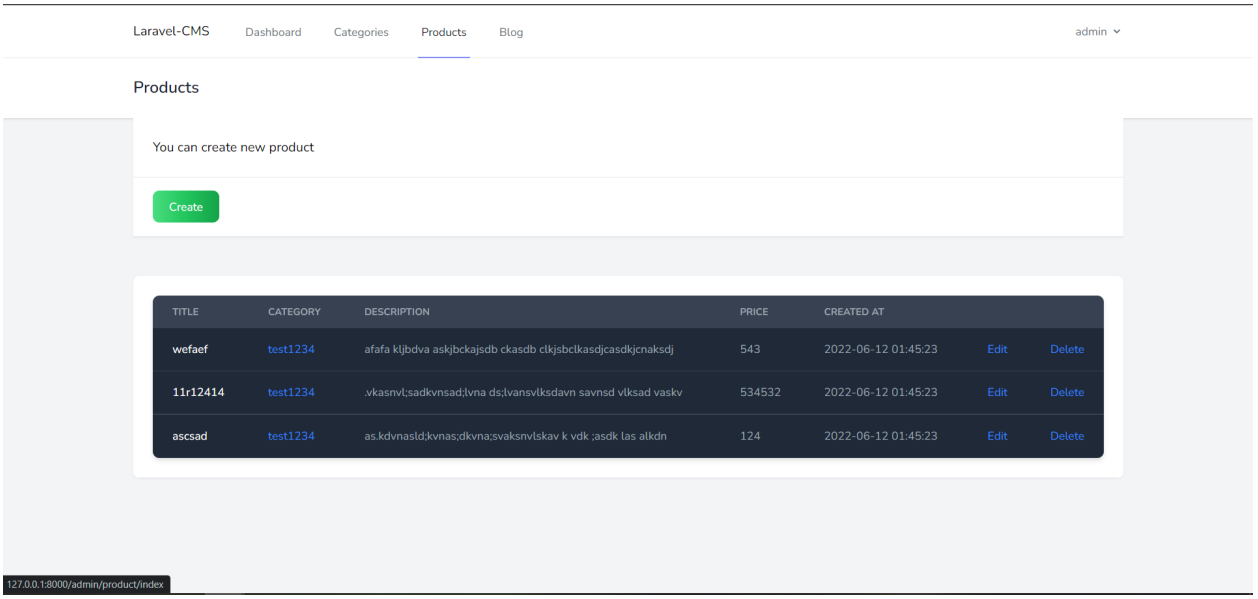

Рисунок Б.5 – Сторінка адміністративної частини системи сутності Товари

Сторінку адміністративної частини системи створення сутності Товари зображено на рисунку Б.6

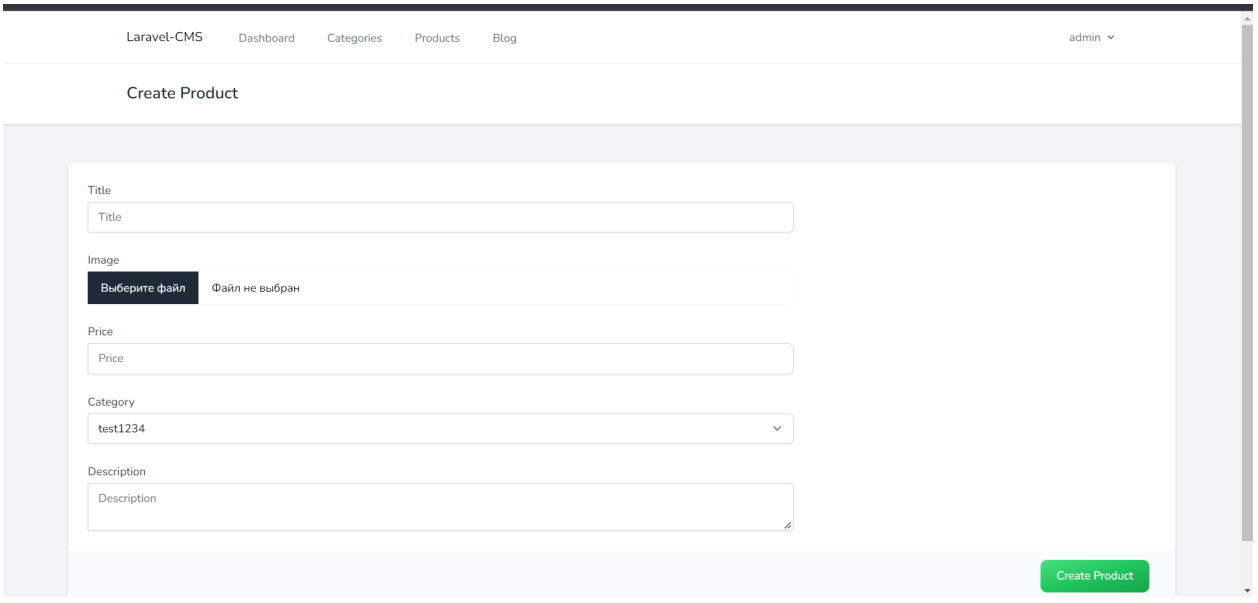

Рисунок Б.6 – Сторінка адміністративної частини системи створення сутності Товари

Сторінку адміністративної частини системи редагування сутності Товари зображено на рисунку Б.7

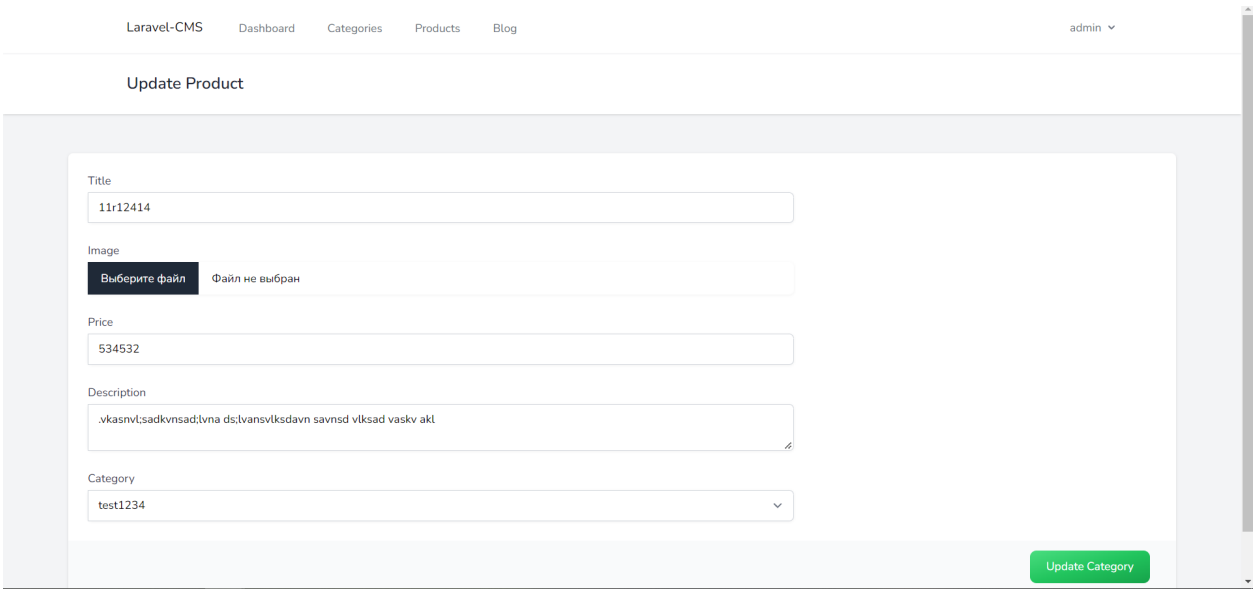

Рисунок Б.7 – Сторінка адміністративної частини системи редагування сутності Товари

Сторінку адміністративної частини системи сутності Пости зображено на рисунку Б.8

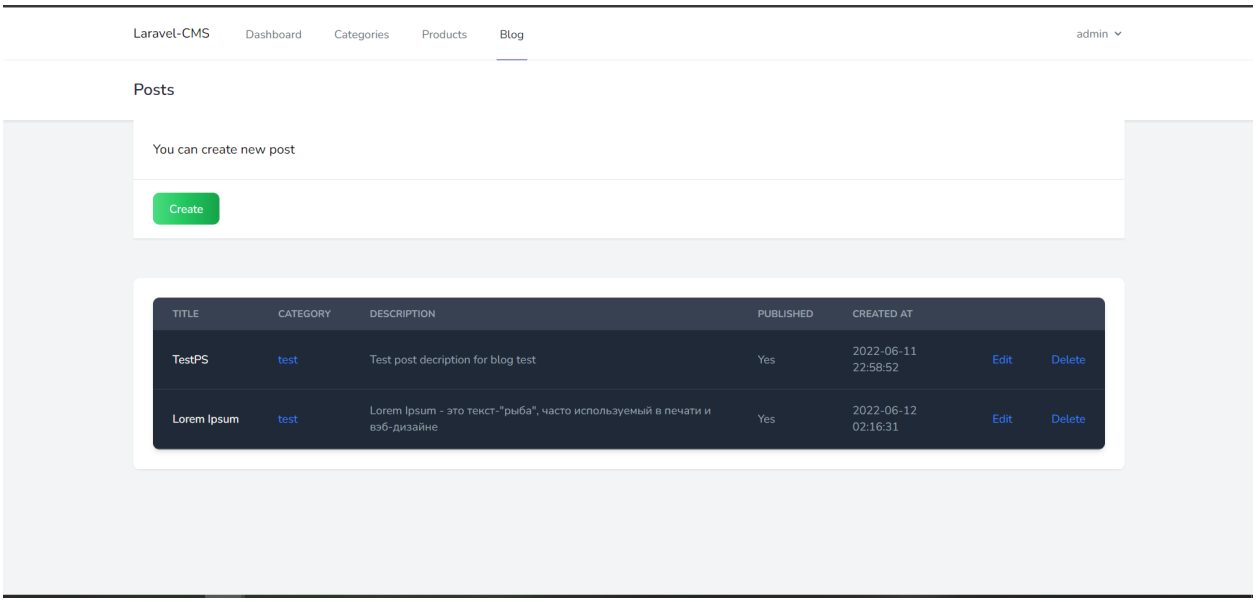

Рисунок Б.8 – Сторінка адміністративної частини системи сутності Пости

Сторінку адміністративної частини системи створення сутності Пости зображено на рисунку Б.9

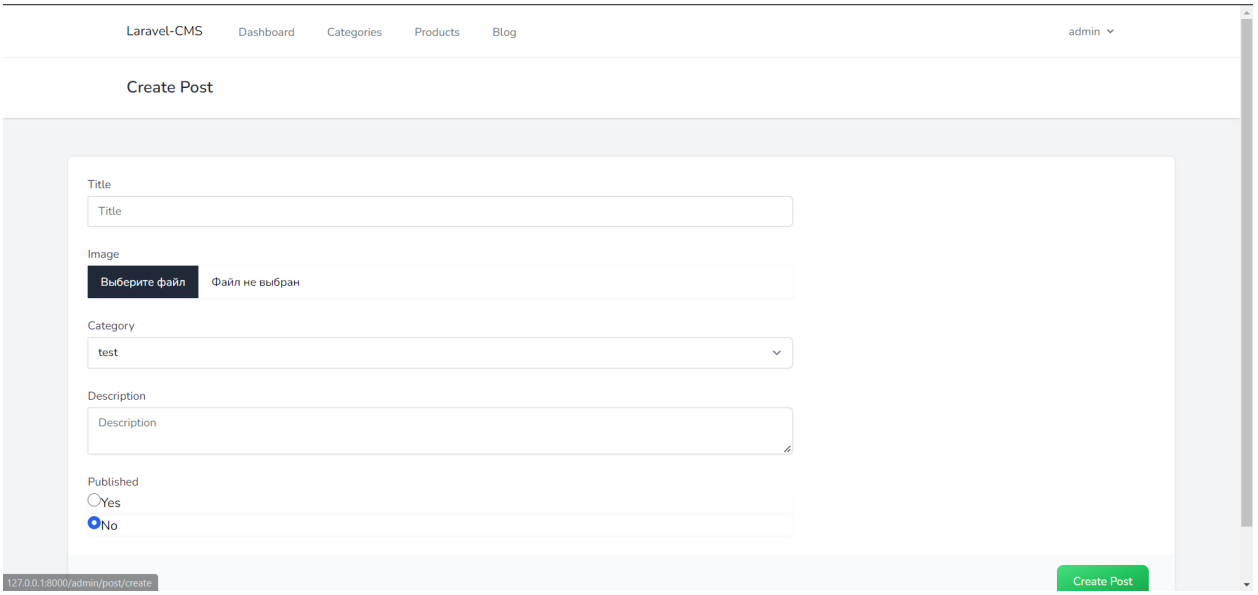

Рисунок Б.9 – Сторінка адміністративної частини системи створення сутності Пости

Сторінку адміністративної частини системи редагування сутності Пости зображено на рисунку Б.10

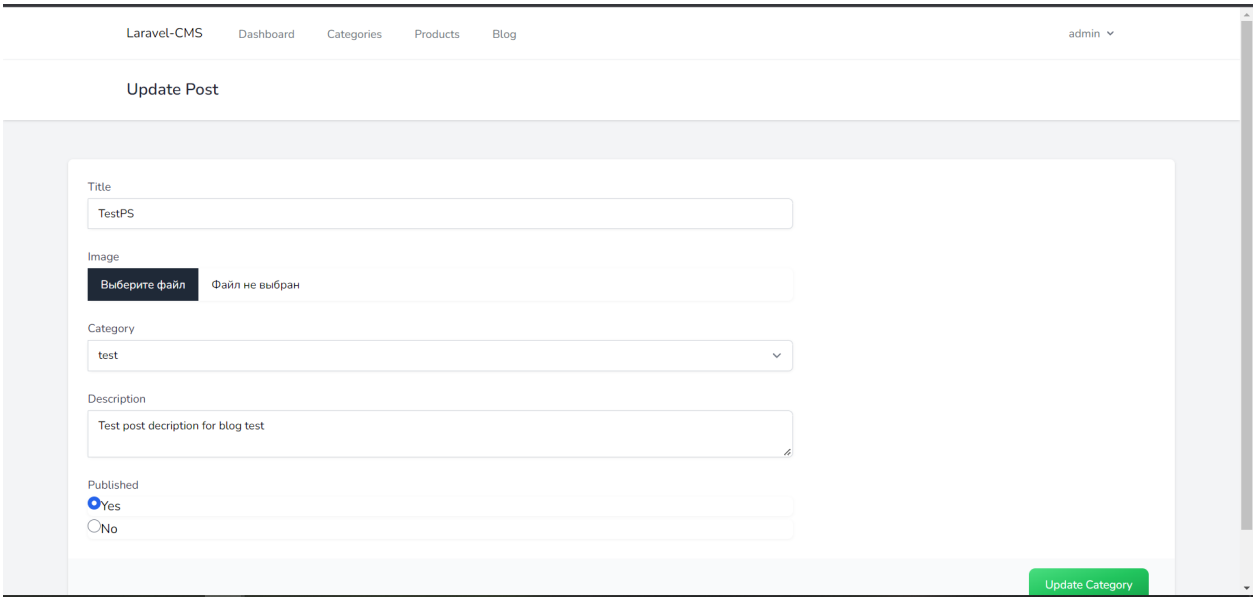

Рисунок Б.10 – Сторінка адміністративної частини системи редагування сутності Пости