Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Навчально-науковий
інститут електроенергетики
(інститут)
Факультет інформаційних технологій
(факультет)
Кафедра інформаційних технологій та комп'ютерної інженерії
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня бакалавра

студента **Литвиненка Дмитра Андрійовича**
(ПІБ)

академічної групи 123-20-2
(шифр)

спеціальності 123 Комп'ютерна інженерія
(код і назва спеціальності)

за освітньо-професійною програмою 123 Комп'ютерна інженерія
(офіційна назва)

на тему «Комп'ютерний комплекс компанії "TopEuroGoods" з детальним опрацюванням побудови CMS та корпоративної мережі»

(назва за наказом ректора)

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	проф. Цвіркун Л.І.			
спеціальної частини	проф. Цвіркун Л.І.			
розділів:				
розробка апаратної частини	доц. Ткаченко С.М.			
розробка корпоративної мережі	ас. Бешта Л.В.			

Рецензент				

Нормоконтролер	проф. Цвіркун Л.І.			

Дніпро
2024

ЗАТВЕРДЖЕНО:

завідувач кафедри
інформаційних технологій
та комп'ютерної інженерії
(повна назва)

_____ Гнатушенко В.В.
(підпис) (прізвище, ініціали)
« 25 » січня 2024 року

ЗАВДАННЯ
на кваліфікаційну роботу
ступеня бакалавр

студента Литвиненка Д.А. **академічної групи** 123-20-2
(прізвище та ініціали) (шифр)

спеціальності 123 Комп'ютерна інженерія

за освітньо-професійною програмою 123 Комп'ютерна інженерія
(офіційна назва)

на тему «Комп'ютерний комплекс компанії "TopEuroGoods" з детальним опрацюванням побудови CMS та корпоративної мережі»

затверджену наказом ректора НТУ «Дніпровська політехніка» від 23.05.2024 № 469-с

Розділ	Зміст	Термін виконання
Стан питання та постановка завдання	Внаслідок пандемії Covid-19 та російської агресії створити зручну систему керування контентом для компанії "TopEuroGoods" з детальним опрацюванням створення високо-функціональної CMS та веб-сайту електронної комерції. Розробити корпоративну мережу.	10.05.2024
Розробка апаратної частини	Зробивши аналіз підприємства сформувати технічні вимоги до комп'ютерної мережі та апаратної частини мережі.	17.05.2024
Розробка корпоративної мережі	Побудова моделі корпоративної мережі компанії у програмі Cisco Packet Tracer, виконати налаштування та перевірку роботи системи.	24.05.2024
Розробка компонента системи	Зробивши аналіз, знайти оптимальні інструменти для виконання завдання. Розробити систему адміністрації контенту для відповідного бізнесу та побудувати веб-сайт для того ж бізнесу на основі створеної системи.	31.05.2024

Завдання видано _____ проф. Цвіркун Л.І.
(підпис керівника) (прізвище, ініціали)

Дата видачі 25.01.2024

Дата подання до екзаменаційної комісії 27.06.2024

Прийнято до виконання _____ Литвиненко Д.А.
(підпис студента) (прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 109 с., 62 рис., 6 табл., 1 дод., 18 джерел.

CMS, FRONTEND, ЕЛЕКТРОННА КОМЕРЦІЯ, УПРАВЛІННЯ КОНТЕНТОМ, АДМІНІСТРАЦІЯ КОНТЕНТУ, BACKEND, БАЗА ДАНИХ, CSS ФРЕЙМВОРКИ, CISCO, МЕРЕЖА, ПОТРЕБИ БІЗНЕСУ, B2C.

Об’єкт розробки – комп’ютерний комплекс з системою керування контентом для малого та середнього бізнесу та веб-сайт побудований за допомогою розробленої системи для компанії "TopEuroGoods". Побудована корпоративна мережа.

Мета роботи – побудова корпоративної мережі та системи керування контентом компанії "TopEuroGoods" з детальним опрацюванням створення високо-функціональної CMS та веб-сайту електронної комерції.

Платформа управління контентом разом з корпоративною мережею надає можливість швидко та надійно створювати інтернет-магазини, розпоряджатися ними одночасно; керувати контентом, починаючи від налаштування банерів, завершуючи іншими характеристиками продуктів, що буде продавати магазин в мережі Інтернет. Технічну та програмну модернізацію; зручно вести фінансову звітність через зручну систему звітності реалізовану у вигляді таблиць. Веб-сайт побудований як SPA надає чудово налагоджує взаємодію між покупцем та продавцем. Система керування контентом та відповідний веб-сайт побудований за допомогою цієї системи було розроблено відповідно до завдань, поставлених для кваліфікаційної роботи бакалавра.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

CMS (Content management system) – система управління вмістом;

SEO (Search engine optimization) – пошукова оптимізація;

DXP (Digital experience platform) – платформа цифрового досвіду;

CRM (Customer relationship management) – управління відносинами з клієнтами;

ERP (Enterprise resource planning) – планування ресурсів підприємства;

ВСТУП

Більшість людей, а саме 66% має доступ до всесвітньої мережі Інтернет [1]. Тож, це свідчить лише про те, що веб-застосунки матимуть лише більший вплив у майбутньому, адже третина людства ще не використовує “Всесвітню павутину”. У той же час комп’ютери та телефони у двадцять першому сторіччі починають використовуватись все більше, починаючи з бухгалтерського обліку, програмування, закінчуючи лінгвістикою, юридичною справою. І справді, людське життя вже тісно пов’язано з цими обчислювальними машинами, котрі використовуються для доступу в Інтернет та стали помічниками людства у багатьох галузях.

З початком пандемії Covid-19, що прокотилася руйнівною “хвилею” по світовій економіці та спричинила найбільшу глобальну економічну кризу за понад століття [2]. Криза призвела до різкого зростання нерівності як всередині країн, так і між ними. Попередні дані свідчать про те, що відновлення після кризи буде таким же нерівномірним, як і її початкові економічні наслідки: країнам з економікою, що розвивається, та економічно незахищеним групам населення знадобиться набагато більше часу, щоб відновити спричинені пандемією втрати доходів і засобів до існування. Якщо згадати 2020-2021 роки, то велика кількість країн влаштували так званий локдаун, а отже, магазини не працювали, люди не могли нормально купувати товар, що призвело до проблем малого та середнього бізнесу. Саме через це почала з’являтись все більша потреба у електронній комерції. У 2023 році 71% компаній мали веб-сайт [13]. Це більше, ніж у попередні роки, і значною мірою може бути пов'язано з впливом пандемії Covid-19. З переходом до онлайн-комерції та віддаленої роботи компанії усвідомили важливість присутності в Інтернеті, щоб охопити ширшу аудиторію та залишатися конкурентоспроможними в сучасному цифровому ландшафті. Пандемія Covid-19 прискорила тенденцію до створення компаніями веб-присутності, і цілком ймовірно, що ця тенденція збережеться і в майбутньому. Конструктори веб-сайтів, що не містять коду, також спростили для

бізнесу створення присутності в Інтернеті як ніколи.

З початком же повномасштабного вторгнення на територію України виникла ще більша потреба у створенні зручного інструменту менеджменту контента для електронної комерції, адже через постійні ракетні обстріли малий та середній бізнес почав страждати, а мікро-, малі та середні підприємства є основою економіки України, складаючи 99,98% усіх суб'єктів господарювання, забезпечуючи 74% усіх робочих місць та створюючи 64% доданої вартості [3]. З початку повномасштабного вторгнення 64% тимчасово призупинили або закрили свою господарську діяльність [3]. Однак переважна більшість відновила свою діяльність, і в жовтні 2023 року лише 9,6% компаній, які призупинили свою діяльність, перебувають під загрозою закриття [3].

У світлі цих викликів стає очевидною необхідність впровадження CMS для бізнесу. Системи адміністрування контенту надають бізнесу можливість швидко та ефективно адаптуватися до нових умов, оптимізуючи процеси створення, управління та публікації цифрового контенту. Тож, системи управління контентом є необхідними для сучасного бізнесу через:

Спрощення управління контентом: дозволяють легко створювати, редагувати та публікувати контент без потреби в глибоких технічних знаннях. Це особливо важливо для малих та середніх підприємств, які часто не мають великих IT-відділів, і мають змогу лише винаймати промоутерів, або UGC creators.

1. Підвищення ефективності роботи: автоматизація багатьох процесів у таких системах допомагає зменшити навантаження на співробітників та прискорити робочі процеси. Це дає змогу бізнесу зосередитися на стратегічних завданнях та розвитку.
2. Адаптивність та гнучкість: гнучкість у налаштуванні та масштабуванні ресурсів є складовою успіху бізнеса, а в умовах війни це ще більш корисно. Тож, це дозволяє швидко реагувати на зміни ринкових умов.
3. Покращення користувацького досвіду: створення власної, сучасної системи

менеджменту контенту з привабливим дизайном застосунку, створить чудову та просту середу для контент-мейкерів, а швидкість роботи покращить досвід для користувачів. Це особливо важливо для електронної комерції, де зручність та швидкість користування сайтом може значно вплинути на рівень продажів.

Таким чином, впровадження CMS-систем дозволяє бізнесу залишатися конкурентоспроможним, швидко реагувати на виклики часу та ефективно управляти своїм цифровим контентом, що є ключовим фактором у забезпеченні стійкості та розвитку в умовах сучасних економічних і соціальних змін.

Також важливо згадати про світові тренди. Двадцять дев’ять відсотків бізнесу ведеться онлайн. Згідно з останніми статистичними даними, 28% всієї ділової активності зараз ведеться онлайн [15]. Цей перехід до онлайн-комерції відображає зростаючу важливість Інтернету в сучасному діловому світі та тенденцію до розвитку електронної комерції. Простота та зручність онлайн-транзакцій зробили їх популярним вибором серед споживачів, і бізнес користується цим зрушенням, розширюючи свою присутність в Інтернеті.

Створення переконливого контенту – це одна з складових переваги ресурсу над іншим веб-ресурсом в Інтернеті. Однак лише за допомогою правильних інструментів зусилля можуть досягти максимальної ефективності. Інвестиції в систему управління контентом приносять величезні творчі та маркетингові можливості. Підприємства, які прагнуть заявити про себе в сфері електронної комерції, все частіше обирають рішення під назвою CMS. Тож, ціллю є створення продукту, котрий допоможе бізнесу зосередитись на своїй основній діяльності, доручивши управління контентом надійному цифровому інструменту. Використовуючи високо-функціональну систему управління контентом, компанії можуть координувати створення веб-сайту та світу цифрового контенту, не вдаючись до передових технологічних ноу-хау.

В результаті, системи адміністрування контенту відіграють вирішальну роль

у створенні стійких та адаптивних бізнес-моделей, що можуть швидко реагувати на нові виклики та можливості. Вони надають бізнесам необхідні інструменти для управління своїм контентом, забезпечуючи тим самим їхню конкурентоспроможність та здатність розвиватися в умовах постійних змін. У сучасному світі, де цифровий простір стає все більш важливим, інвестування у високоякісну CMS є не лише розумним рішенням, але й необхідністю для довготривалого успіху.

1 СТАН ПИТАННЯ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1 Стисла характеристика галузі та умов застосування системи керування контентом, що проєктується.

Галузь електронної комерції відноситься до сфери бізнесу, яка зосереджена на проведенні комерційних операцій в Інтернеті. Вона включає в себе купівлю та продаж продуктів або послуг через Інтернет за допомогою електронних засобів. Електронна комерція зробила революцію в способі ведення бізнесу і стає все більш популярною з широким розповсюдженням Інтернету. Важливо зазначити, що у 2024 році цифровий всесвіт налічує вже близько 1,09 мільярда веб-сайтів [14]. Щодня створюється 252 000 нових сайтів, що показує невпинну еволюцію та розширення інтернету і як наслідок бізнесу. Ключові компоненти домену електронної комерції:

1. Платформи для онлайн-покупок: це веб-сайти або додатки, які слугують цифровою вітриною, де клієнти можуть переглядати та купувати товари чи послуги. Прикладами є Prom, Amazon, eBay, OLX, сайти побудовані на Shopify.
2. Каталог продуктів: каталог товарів містить детальну інформацію про товари чи послуги, доступні для придбання, зокрема описи, зображення, ціни та наявність. Це дозволяє клієнтам приймати обґрунтовані рішення щодо своїх покупок.
3. Кошик: функція кошика дозволяє клієнтам обирати та збирати товари, які вони бажають придбати, продовжуючи перегляд сайту. Це зручний спосіб керувати кількома продуктами перед тим, як перейти до процесу оформлення замовлення.
4. Платіжні шлюзи: це онлайн-сервіси, які сприяють безпечним транзакціям шляхом шифрування конфіденційної фінансової інформації. Вони дозволяють клієнтам здійснювати платежі за допомогою різних методів, таких як кредитні картки, дебетові картки, цифрові гаманці або банківські перекази.

5. Управління замовленнями: цей компонент передбачає обробку та виконання замовлень клієнтів. Він включає такі завдання, як підтвердження замовлень, управління кількістю доданих продуктів, відстеження замовлень і логістику відвантаження для забезпечення своєчасної доставки.
6. Облікові записи клієнтів: платформи електронної комерції часто пропонують функції облікових записів клієнтів. Клієнти можуть створювати облікові записи, щоб зберігати свої вподобання, відстежувати історію замовлень, керувати способами оплати та отримувати персоналізовані рекомендації.
7. Безпека: платформи електронної комерції повинні надавати пріоритет безпеці, щоб захистити інформацію про клієнтів і фінансові дані. Такі заходи, як шифрування, безпечні механізми входу та дотримання стандарту PCI-DSS (стандарт безпеки даних індустрії платіжних карток) допомагають захистити конфіденційну інформацію.
8. Логістика та доставка: ефективне управління логістикою та доставкою має важливе значення для успішних операцій електронної комерції. Це включає в себе координацію інвентаризації, пакування, перевізників, відстеження та обробку повернень або обмінів.
9. Підтримка клієнтів: надання підтримки клієнтам має вирішальне значення в сфері електронної комерції. Це може включати такі канали, як живий чат, електронна пошта або телефонна підтримка, щоб відповідати на запити клієнтів, вирішувати проблеми та надавати допомогу протягом усього процесу купівлі. Також для цього має бути налагоджена дуже добре комунікація між IT та відділом підтримки підприємства.
10. Маркетинг і просування: підприємства електронної комерції використовують різні маркетингові методи для залучення клієнтів, підвищення популярності та стимулювання продажів. Це може включати пошукову оптимізацію, цифрову рекламу, маркетинг у соціальних мережах і персоналізовані рекомендації, засновані на поведінці клієнтів. Особливо популярними для малого та середнього бізнесу зараз стали платформи з короткими роликами, котрі використовують такі алгоритми, що навіть непопулярний ролик на

перший погляд має змогу просуватись у тренди.

11. Аналітика: платформи електронної комерції збирають та аналізують дані, щоб отримати уявлення про поведінку, вподобання та тенденції клієнтів. Ця інформація допомагає оптимізувати маркетингові стратегії, покращити користувацький досвід та приймати бізнес-рішення на основі даних.
12. Мобільна комерція: з розвитком мобільних пристроїв мобільна комерція набуває все більшого значення. Платформи електронної комерції повинні надавати зручні для мобільних пристроїв інтерфейси та додатки, щоб задовольнити потреби користувачів, які отримують доступ до їхніх послуг через смартфони та планшети.

Важливо уточнити, що сфера електронної комерції охоплює широкий спектр бізнесів, включаючи моделі B2C (бізнес-споживач) і B2B (бізнес-бізнес), а також різні сектори, такі як роздрібна торгівля, подорожі, готельно-ресторанний бізнес і сфера послуг.

Дуже корисним інструментом у сфері електронної комерції є ERP, він же планування ресурсів підприємства, що допомагає налагодити взаємодію між усіма частинами підприємства, включаючи і доставку.

У цьому ж випадку “TopEuroGoods” є B2C платформою, адже продукти, що продає компанія потрапляють одразу до споживача.

1.2 Характеристика і структура компанії

Малий бізнес, такий як “TopEuroGoods” є важливою частиною сучасної економіки України, тож обов’язково треба розглянути характеристику і структуру компанії, адже це дозволить краще зрозуміти контекст, в якому розробляється система менеджменту контенту та її важливість для підвищення працездатності бізнесу.

Структура компанії включає в себе:

- керівника, котрий є головним та відповідає за стратегічне планування та

розвиток бізнесу.

- Фінансовий відділ, котрий займається сплатою податків, звітністю, оптимізацією витрат, аналізом і бюджетуванням.
- Відділ підтримки клієнтів, котрий обробляє скарги або побажання клієнтів, надаючи консультаційні послуги, котрі мають задовільнити кожного.
- Промоутерський відділ, котрий займається рекламою, маркетингом продукції. Задача цього відділу своєчасно поставляти та створювати контент на різноманітні платформи.
- Відділ логістики, котрий відповідає за планування, виконання та контроль ефективного і економічного руху товарів, послуг та інформації від пункту походження до пункту споживання з метою задоволення вимог клієнтів. Цей відділ займається організацією транспортування, зберігання та контролю запасів.
- Відділ кадрів, котрий займається управлінням персоналом компанії. Основні функції відділу включають підбір і наймання нових працівників, навчання та розробляння таргетованої стратегії розвитку для кожного співробітника, управління трудовими відносинами, забезпечення відповідності трудового законодавства та розробка мотиваційних програм.

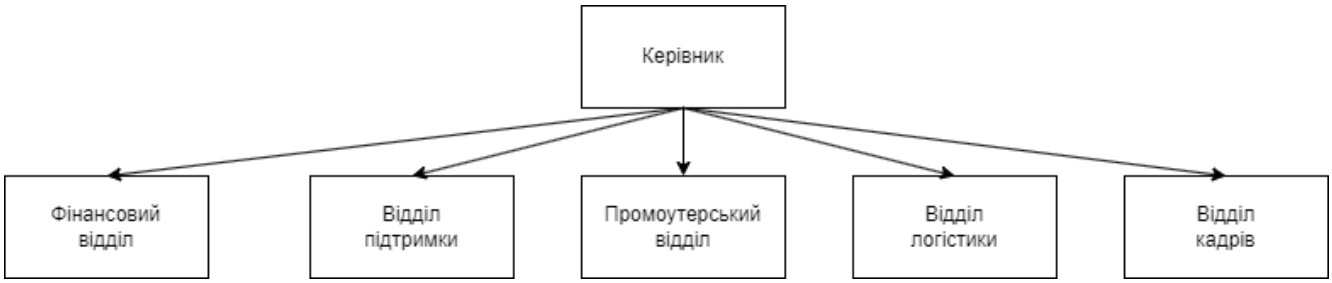


Рисунок 1.1 – Структура компанії

1.3 Мережі. Мережі у бізнесі

Комп'ютерні мережі стали невід'ємною складовою сучасного суспільства і відіграють вирішальну роль у функціонуванні бізнесу, урядів та приватних осіб. З розвитком технологій комп'ютерні мережі продовжують розвиватися і відіграватимуть все більш важливу роль у повсякденному житті індивідів. У сучасному світі комп'ютер став невід'ємною частиною ділового сектору не тільки для професійної діяльності, але й для особистої діяльності. З розвитком технологій з'явилася мережа, і поступово від початкової дротової технології почали з’являтись і бездротові технології. Якщо замислитись, то стане зрозуміло, що мережеві технології впливають на все, адже це те, що дозволяє встановити взаємозв'язок між комп'ютерами і налагодити процеси бізнесу. Мережі є основою комунікації в інформаційних технологіях.

Найперші типи мереж з'явилися в 1950-х роках, передаючи інформацію по телефонних лініях, в основному для використання в оборонній та військовій сферах. Перший комерційний модем для комп'ютерів був випущений в 1958 році американською телекомунікаційною компанією AT&T, що дозволило передавати дані зі швидкістю всього 110 біт в секунду [11]! Відтоді, з появою інтернету, комп'ютерні мережі стали неймовірно поширеними, потужними, функціональними та необхідними. Вони мають вирішальне значення для сучасних бізнес-операцій і стали більш безпечними, програмованими та автоматизованими. Звісно ж змінились і вимоги, сучасна мережа має:

- Захищати дані – сучасні мережеві рішення часто мають вбудовані функції безпеки і можуть бути інтегровані зі сторонніми рішеннями для захисту від шкідливого програмного забезпечення.
- Адаптивними – сучасні комп'ютерні мережі, як правило, є програмно-визначеними, що означає, що ними можна швидко керувати централізовано.
- Масштабованість – мережеві послуги можуть бути оптимізовані, збільшені або зменшені в залежності від потреб. Вони можуть бути інтегровані у великих масштабах.
- Віртуальні – фізична мережева інфраструктура може бути розділена, створюючи "накладні" мережі, в яких вузли віртуально пов'язані між собою і

можуть обмінюватися даними. Багато великомасштабних мереж накладаються в Інтернеті.

Звісно ж в контексті поставленої задачі потрібно обов’язково розглянути які існують типи бізнес-комп'ютерних мереж. Комп'ютерні мережі для бізнесу можна класифікувати багатьма способами, найчастіше за географічною ознакою. Невеликі компанії зазвичай використовують локальні мережі. Це сукупність пристроїв в одному фізичному місці, наприклад, в офісі, на складі чи в магазині. Традиційно приміщення з'єднують кабелем.

Існує два типи локальних мереж: однорангові та клієнт-серверні. Технологія цих мереж може бути дротовою або бездротовою. Однорангові мережі включають два або більше комп'ютерів, об'єднаних разом, із загальними ресурсами, доступними кожному в мережі. Комп'ютери мають однакові привілеї і діють як клієнт і сервер, спілкуючись безпосередньо один з одним. Наприклад: принтером на одному комп'ютері може користуватися будь-хто інший в мережі. Переваги – дуже простий і легкий у налаштуванні. Мінуси – пристрої, до яких надає спільний доступ комп'ютер, залежать від працездатності цього комп'ютера. Якщо раптом вимкнеться комп'ютер, то доступ буде втрачений до всіх підключених пристроїв, файлів і папок; низький рівень безпеки та масштабованості.

Клієнт-серверні мережі включають кілька робочих станцій, підключених до одного або декількох серверів, на яких зберігаються файли та ресурси, до яких надається спільний доступ. Клієнти – отримують доступ до них з сервера. Сервер виконує роботу, даючи клієнтам інформацію, необхідну для їхньої роботи. Це може бути інформація про безпеку, дозволи користувачів, доступ до файлів, принтерів, електронної пошти, оновлення програмного забезпечення, тощо. Плюси – надійна система, яка є масштабованою та ефективною. Мінуси – початкові витрати можуть бути дорогими; потребує більше досвіду для налаштування, управління та обслуговування; можуть бути досить ізольованими від бізнесу.

Виходячи з отриманої інформації, клієнт-серверна мережа, безумовно є найкращим вибором для бізнесу.

Але на останок потрібно згадати про хмарні комп'ютерні мережі. Традиційна

мережа клієнт-сервер стала надто дорогою та застарілою. Дистанційна та гібридні форми роботи різко зросли в популярності через вже згадані вище Covid-19 та через вторгнення росії на територію України. Наразі 12,7% штатних працівників працюють з дому, що свідчить про швидку нормалізацію віддаленого робочого середовища. Водночас, значні 28,2% працівників адаптувалися до гібридної моделі роботи. Ця модель поєднує роботу вдома та в офісі, пропонуючи гнучкість та підтримуючи рівень фізичної присутності на робочому місці [12]. Також на це вплинула тенденція бізнесу працювати на національному та міжнародному рівнях. Для таких компаній, з'явився третій тип мережі, доступний для вибору. Цей тип мережі можна розглядати як суміш локальних і глобальних мереж. Різні хмарні сервіси по типу Microsoft SharePoint і Google Workspace надають компаніям інструменти і забезпечують функціональність локальної мережі без прив'язки до одного місця.

1.4 Історія створення систем менеджменту контенту

Системи управління контентом позначають першим етапом розвитку “Всесвітньої павутини”, що характеризується простими статичними веб-сайтами. Перші системи управління контентом почали з’являтись у 1989 році, коли Тім Бернерс-Лі запропонував гіпертекстову мову HTML, а наприкінці 1990 року написав браузер і серверне програмне забезпечення. HTML походить від SGML, стандартної узагальненої мови розмітки, яку створили в IBM Чарльз Ф. Голдфарб, Ед Мошер і Рей Лорі в 1970-х роках. Перші веб-сайти були простими текстовими файлами HTML. У 1993 році браузери Mosaic почали підтримувати зображення, які могли з'являтися разом з текстом, а статичні сайти, схожі на брошури, ділилися інформацією про компанії та продукти. На початку 1990-х років першим кроком до керування вмістом веб-сторінки стала технологія Server Side Includes (SSI). Server Side Includes дозволяє користувачу зберігати частини сайту окремо від основного контенту, наприклад, меню сайту або нижній колонтитул. Приблизно в той же час на сцену вийшов Common Gateway Interface, який дозволив створювати

інтерактивні форми.

Перші технології, подібні до CMS, полягали у використанні сценаріїв на стороні сервера для створення контенту, що надсилається з сервера до веб-браузера. Відомі мови програмування для веб-розробки, такі як PHP та Active Server Pages (ASP), з'явилися в 1995-1997 роках, а в 1999 році до них додалася Java Server Pages. Ці серверні скриптові механізми спростили створення динамічно згенерованих веб-сторінок, і з цього моменту було створено передумови для появи інструментів менеджменту контенту.

Динамічна революція відбулася в 1997 році з появою об'єктної моделі документа (Document Object Model, або ж DOM), API для документів HTML і XML, що дозволяє ідентифікувати і контролювати частини документа. DOM дозволила користувачам отримувати доступ і маніпулювати стилями елементів HTML, таких як тіло або розбиття сторінки. Кілька років по тому впровадження асинхронних JavaScript і XML (Ajax) додало динамічної революції, коли розробники могли запитувати і отримувати дані для оновлення веб-сторінки, тобто без перезавантаження її.

У період між 1997 та 2002 роками багато організацій та компаній створили свої власні системи керування контентом з використанням існуючих технологій. Виробники, агентства, компанії, що займаються просуванням подкастів, чи газети лише розростались, і їм потрібно було розміщувати там свій контент, і як наслідок розросталась мережа Інтернет. Характерною рисою CMS тієї епохи було те, що вони часто були адаптовані до конкретних потреб організації, яка стояла за ними. Кілька корпоративних CMS почали з'являтися в середині та наприкінці 1990-х років, визнаючи ринок для уніфікованих і професійно розроблених систем управління контентом. Прикладами корпоративних систем адміністрації контенту того періоду є FileNet, StoryBuilder, Interwoven, Documentum, FatWire, FutureTense та Inso. У той самий період служба веб-хостингу GeoCities досягла свого апогею, ставши третім найбільш відвідуваним сайтом в мережі Інтернет. Придбаний Yahoo! у 1999 році, GeoCities став однією з перших веб-орієнтованих CMS, що дозволила користувачам керувати своїми веб-сайтами. В результаті купа персональних, хобі-

сайтів та інших видів веб-сайтів різних доменів почали з'являтись на GeoCities та стали основним продуктом кінця 1990-х - початку 2000-х років.

Початок і середина 2000-х років стали свідками дедалі більшої професіоналізації та спеціалізації систем управління контентом. У цей період почав з'являтися термін "WCM", що означає "управління веб-контентом". У той час як CMS є найширшим терміном, що позначає управління контентом також для інтрамереж, архівів і бізнес-операцій, система управління веб-контентом орієнтована виключно на Інтернет. На початку та середини 2000-х років системи адміністрування контенту почали задовольняти потреби підприємств та бізнесу у все більш професійний спосіб. Одночасно з цим з'явилися CMS з відкритим кодом, такі як Drupal, WordPress і Joomla, що і досі є доволі популярними рішеннями при створені веб-ресурсу, котрий потребує адміністрування його контенту у зручний спосіб через графічний інтерфейс. Більшість CMS містили як внутрішню, так і зовнішню технологію веб-сайту і могли працювати з текстом, зображеннями та іншими файлами для зберігання, відображення та завантаження.

Завдяки технологіям, що дозволяють динамічно надавати контент, світ став свідком появи так званого "WEB 2.0". З плином того, як веб-сайти переходили від статичних брошур до більш інтерактивного досвіду, виникла потреба в частішому оновленні контенту та управлінні ним. Таким чином, однією з центральних ролей систем менеджменту контенту стало забезпечення можливості для різних ролей користувачів і дозволів у наданні контенту. Крім того, технологічні інновації призвели до постійного зростання інших функцій CMS, включаючи попередній перегляд, реалізацію адаптивного дизайну, коментарі відвідувачів, системи відстеження, системи дозволів, перетягування, візуальні редактори, шаблони, інтеграцію з електронною комерцією, аналітикою, CRM і ERP, тощо.

Ось найпопулярніші функції в системах адміністрування контенту з року в рік (завдяки Дрісу Буйтаерту, засновнику Drupal) [4]:

Таблиця 1.1 – Найпопулярніші функції в системах адміністрування контенту за версією засновника Drupal

Роки	**2000**	**2005**	**2010**	**2015**
Функції	Статичний контент	Що бачиш, те й отримуєш	Інтеграція з соціальними мережами	Клієнтська аналітика
	Відокремлений контент від дизайну	Динамічний контент	Принцип що бачиш, те й отримуєш у дизайні сторінок	Контекстна орієнтованість
	Анімовані GIF-файли	Робочі процеси публікації	Інструменти для співпраці	Підтримка різних пристроїв
		Вміст, створений користувачами	Інтеграція мультимедійних даних	Сервіси з підтримкою
		Модульна архітектура	Інструменти впорядкування складних даних	Управління платформою
		Синдикація		Платформи як сервіс
		Інструменти пошуку		Інтеграція сервісів як програмного забезпечення

Наступним кроком стала платформа цифрового досвіду, вона являє собою еволюцію систем управління контентом у повноцінні маркетингові пакети з кінця 2000-х і до 2010-х років. У світі цифрової трансформації така платформа мала на меті забезпечити цілісний досвід для брендів у різних цифрових точках контакту. В першу чергу вона була орієнтована на підприємства і може включати класичну

CMS разом з аналітикою, електронною комерцією, машинним навчанням, інструментами персоналізації, A/B-тестуванням, SEO і - можливо, найважливіше в контексті сьогоднішнього багатоканального світу - API для доставки контенту на різні канали. Хоча багато хто вважає, що переваги DXP переважують недоліки, але є великі “але”, адже будучи монолітними гігантами, DXP можуть бути повільними в роботі для редакторів контенту і, як правило, незручними для розробників. Це призвело до наступного кроку в еволюції - чогось легшого та гнучкішого, а саме так званого “безголового CMS”.

Наближаючись до 2000-х років, зв'язок між рівнем доставки та CMS став тіснішим. Раніше це було зручно як для редакторів, так і для розробників: перші могли використовувати всі багаті можливості CMS, тоді як другі могли кодувати, тестувати і розгортати як редакторську, так і призначену для кінцевого користувача функціональність в одному пакеті. Певний час це було дуже зручно, але дві нові тенденції 2010-х років порушили цю картину:

1. Впровадження нових каналів у вигляді мобільних телефонів, Інтернету речей, кишенькових пристроїв, тощо.
2. Інновації інтерфейсних фреймворків.

Отже, редактори контенту хотіли, щоб їхній контент був у всіх каналах, тоді як розробники хотіли створювати рішення за допомогою своїх улюблених frontend-інструментів. Разом ці дві тенденції змусили постачальників систем менеджменту контенту побачити необхідність безголового підходу. Безголова CMS розриває тісний зв'язок між контентом і презентацією, який спостерігається в традиційних CMS, натомість стає базою даних зі структурованим контентом, готовим до доставки на кілька каналів - чи то класичний десктоп, чи то мобільний додаток, чи то пристрій Інтернету речей, чи то щось інше. Хоча це не нова концепція, але сучасні безголові суттєво відрізняються від старих. У минулому клієнтом найчастіше був лише веб-сайт, тоді як зараз ідея полягає в тому, щоб доставляти контент на абсолютно різні канали та пристрої.

Тож було вирішено розробляти безголову системи менеджменту контенту, бо це є найбільш актуально і перспективно для зростання бізнесу.

1.5 Аналіз існуючих систем менеджменту контенту

На цей момент існує більше 1000 різних безкоштовних і платних систем адміністрування контенту. Вони постійно розвиваються та вдосконалюються. Періодично з'являються нові системи управління контентом. Список одних з найпопулярніших на даний момент:

- Joomla
- Drupal
- WordPress
- Shopify
- Magento
- Wix
- Contentful
- Strapi
- Webflow

Знайти правильну статистику найпопулярніших CMS складно, оскільки відсоток користувачів в різних джерелах варіюється в залежності від регіону дослідження. Проаналізувавши кілька різних джерел стає зрозуміло, що 3 найпопулярніші безкоштовні CMS на цей момент це WordPress, Joomla та Drupal [5].

Most popular content management systems

© W3Techs.com	usage	change since 1 April 2024	market share	change since 1 April 2024
1. WordPress	43.3%		62.7%	-0.1%
2. Shopify	4.4%	+0.1%	6.4%	+0.1%
3. Wix	2.7%	+0.1%	3.9%	+0.1%
4. Squarespace	2.1%		3.0%	
5. Joomla	1.7%		2.4%	-0.1%

percentages of sites

Рисунок 1.2 – Статистика використання CMS на одному з веб-ресурсів [5]

	WordPress	PrestaShop	Joomla!	Drupal
Popularity (average downloads/week)	1,000,000	N/A	113,000	34,000
CMS market share	63.5%	3.9%	2.6%	0.8%
Number of free themes	over 2,000	N/A	over 900	over 1,800
Number of free plugins	over 27,000	over 3,000	over 7,000	over 24,000
Frequency of updates	42 days	N/A	36 days	51 days
Level of skill required	Amateur	Standard	Standard	Standard

Рисунок 1.3 – Статистика використання CMS використовуючи інфомацію з w3techs, Wordpress, Journal du net, Prestashop, Développez.com, My Guroo [20]

Тож, перейдемо до порівняння CMS. Першою буде WordPress як найбільш широко визнана LAMP CMS з відкритим вихідним кодом. Також потрібно згадати про те, що значить LAMP. LAMP – це стек для створення продукта і є абревіатурою, він включає в себе:

1. Linux – операційна система Linux;
2. Apache – веб-сервер;
3. MariaDB / MySQL – СУБД;
4. PHP – мова програмування, що використовується для створення веб-додатків.

WordPress спочатку був створений у відповідь на тренд блогерства. За 15 років він пройшов шлях від конкурента Bloggers, OverBlog та інших платформ до беззаперечного лідера серед CMS. Понад 60% веб-сайтів, заснованих на CMS з відкритим кодом, використовують його у всьому світі! Без сумніву, це найпростіше рішення для встановлення та управління з панелі управління. Він пропонує користувачам дуже широкий вибір готових до використання шаблонів і плагінів, які можуть швидко задовольнити більшість конфігурацій веб-сайтів, але не потрібно забувати, що плагіни можуть бути як безкоштовні, так і платні. Також плагіни – це є не завжди рішення проблеми, адже вони суттєво сповільнюють роботу веб-сайту, котрий і так є доволі не швидким. Простіше кажучи, WordPress чудово підходить для тих, хто хоче отримати просте та швидке рішення для налаштування, яке не потребує глибоких технічних знань. Більше того, він також

пропонує сильні гарантії SEO. Однак WordPress швидко демонструє свої обмеження, коли справа доходить до складних проєктів, на які він не розрахований. CMS також вимагає певного рівня пильності щодо безпеки - її популярність означає, що вона є головною мішенню для кіберзлочинців.

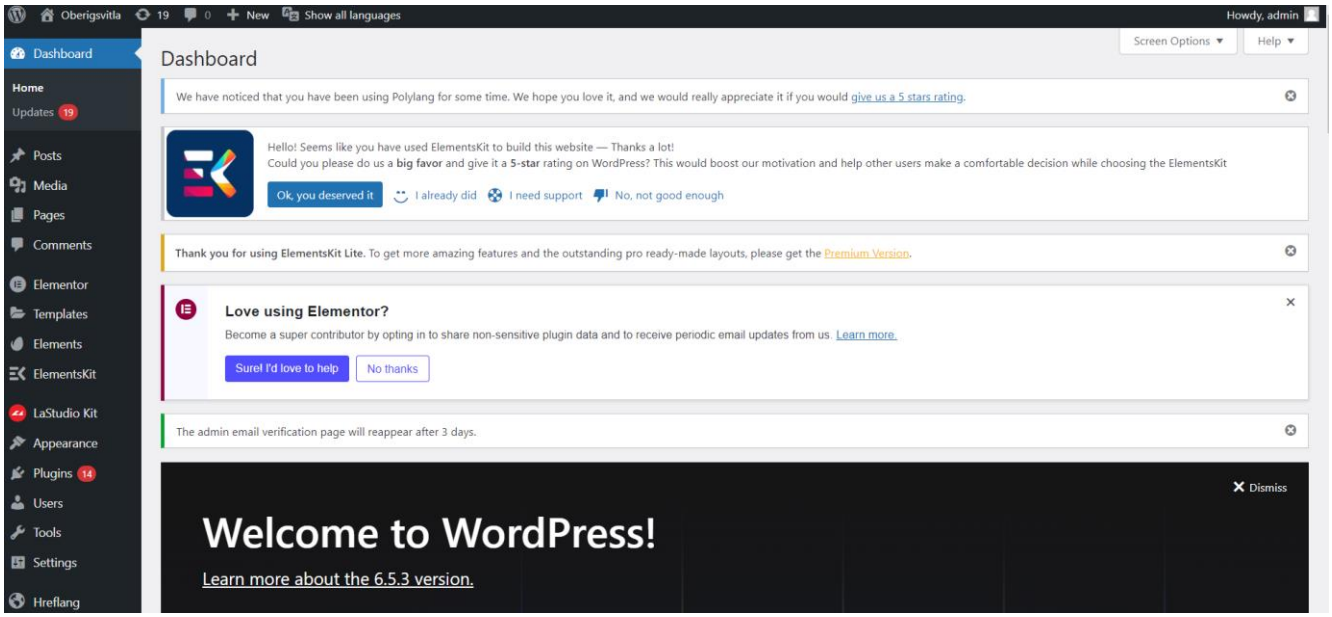


Рисунок 1.4 – Адмін-панель WordPress

Наступним до порівняння буде Joomla!. Joomla була створена з вихідного коду існуючого програмного забезпечення CMS Mambo. Вона довгий час була в списку найпопулярніших систем CMS з відкритим вихідним кодом на ринку, поки не з'явився WordPress. Joomla досить проста у встановленні та налаштуванні, і може використовуватися для швидкого проєктування та створення веб-сайтів, без необхідності мати будь-які спеціалізовані технічні навички. З моменту виходу третьої версії, Joomla зробила крок вперед у сфері безпеки. Вона також пропонує цікаві сервіси для налаштування прав і профілів або управління декількома сайтами. Це дозволяє створювати багаті платформи, включаючи приватні простори, якщо це необхідно. Досі ці функції були доступні лише у більш складних рішеннях.

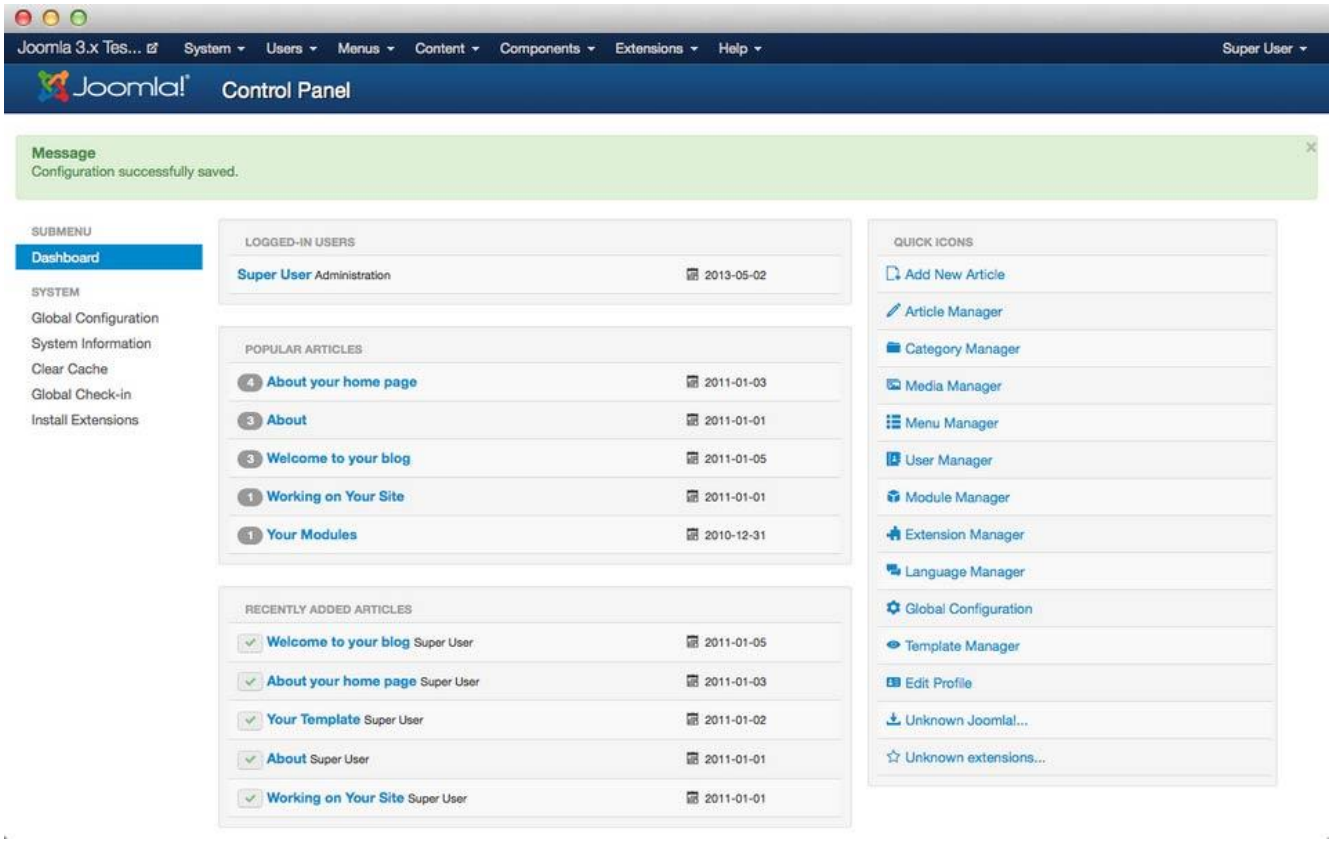


Рисунок 1.5 – Приклад адмін-панелі Joomla

Наступним буде Drupal. Цей гравець на ринку CMS з відкритим вихідним кодом став одним з найкращих. Розроблений як платформа для створення соціальних мереж, Drupal швидко зарекомендував себе як надійна та легко налаштовувана CMS. В основному це пов'язано з інтеграцією фреймворку Symfony (група багаторазових компонентів), на якому вона побудована. Така гнучкість робить її фаворитом серед веб-дизайнерів, коли вони стикаються з амбітними проєктами на замовлення. Завдяки своєму творцеві, Дрісу Буйтаерту, Drupal також популяризував концепцію "відокремленої CMS", яка зараз дуже популярна.

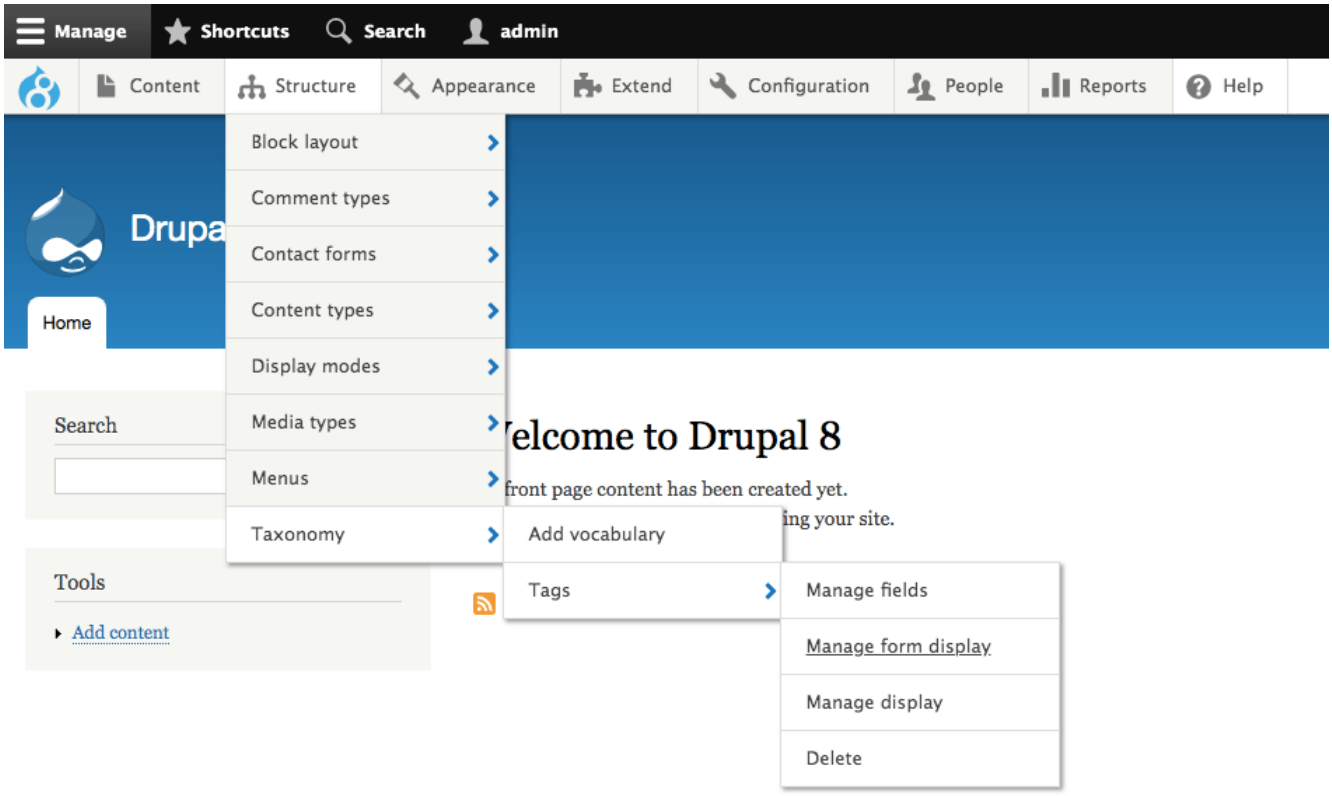


Рисунок 1.6 – Приклад адмін-панелі Drupal

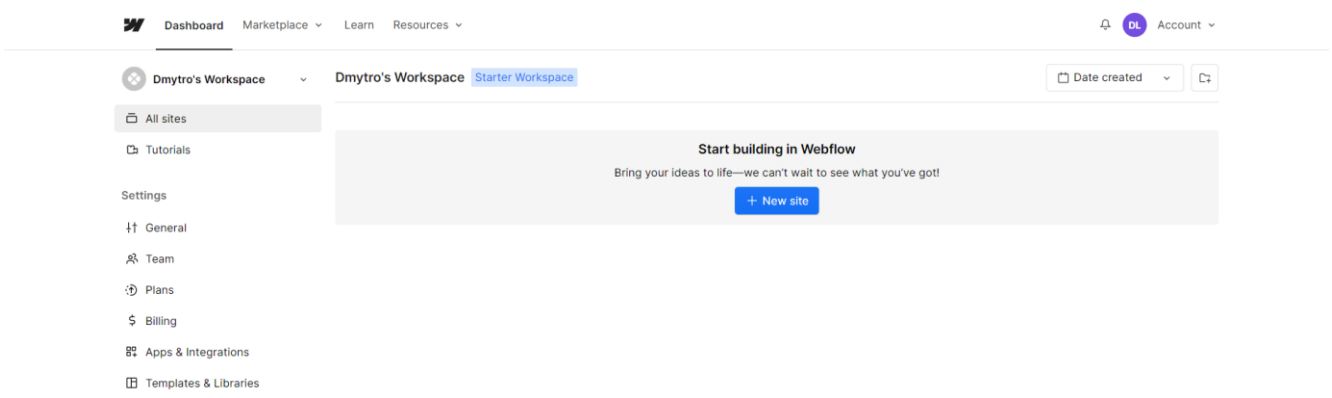


Рисунок 1.7 – Приклад адмін-панелі Webflow

Оскільки “TopEuroGoods” є електронною комерцією, то потрібно розглядати платформи адміністрування контенту саме для електронної комерції.

1. PrestaShop. Улюблене рішення для багатьох інтернет-магазинів. PrestaShop, безсумнівно, є одним з найвідоміших рішень для електронної комерції на

ринку. Він відносно простий у встановленні та налаштуванні, а також вирізняється багатим каталогом доповнень. Ця французька платформа пропонує можливість створити інтернет-магазин за низькою ціною і в рекордно короткі терміни. Хоча вона вирізняється своїми індивідуальними проєктами, її внутрішні якості зуміли залучити провідні бренди (fujitsu, eBay Inc., тощо).

2. WordPress. WordPress, в першу чергу продається як інструмент для створення блогів, пропонує набагато більш просунуті веб-рішення з безліччю плагінів, доступних для інтеграції в систему управління контентом. У поєднанні з іншими розширеннями, необхідними для управління інтернет-магазином, такими як WooCommerce дозволяє розгорнути веб-сайт електронної комерції з високим рівнем свободи та кастомізації. Це рішення особливо добре підходить для веб-сайтів, основною діяльністю яких є пропонування контенту з підключеним елементом електронної комерції. Такий тип використання вимагає певних технічних знань для управління додатковими модулями, обслуговування та безпеки платформи електронної комерції. Це рішення поєднує в собі простоту PrestaShop з технічністю таких рішень, як Magento.
3. Magento. Це рішення для електронної комерції швидко стало лідером на міжнародному рівні. Надійне і легко конфігурується, воно в основному підходить для складних і амбітних проєктів, які вимагають високого ступеня кастомізації і певного технічного досвіду. Однак, інтерфейс адміністрування вимагає певного часу на освоєння. Для повної реалізації його потенціалу також потрібна потужна інфраструктура. На відміну від рішень з керованого веб-хостингу, які постачаються з установкою CMS, таких як PrestaShop або WordPress, в один клік, встановлення рішення Magento вимагає певних технічних знань.
4. Shopify. Shopify стає все більш популярним серед онлайн-продавців по всьому світу. Канадська система магазинів постійно покращує свою присутність на ринку і стала невід'ємною частиною індустрії електронної

комерції. Проста у використанні платформа електронної комерції – це, безумовно, найбільша причина, чому продавці люблять Shopify. Shopify створений для пересічного користувача, а не для досвідчених розробників (хоча можливість кастомізації так званих тем присутня за допомогою liquid.js). Додавати товари, створювати знижки та обробляти замовлення дуже просто. Веб-дизайн є зручним для користувача і навіть легким завдяки інструменту редагування Shopify. Фактично, інтернет-магазин можна створити за 4 хвилини. Сама платформа Shopify надає програмне забезпечення та хостинг, тобто технічну інфраструктуру, необхідну для запуска інтернет-магазину. Shopify підтримує багато різних типів підприємств електронної комерції. Користувачі Shopify можуть продавати фізичні товари, цифрові товари або товари, що постачаються за дропшиппінгом. Продавці Shopify використовують програмне забезпечення для продажу найрізноманітніших товарів, таких як: товари ручної роботи, косметичні товари, товари для дому, спорядження для активного відпочинку, одяг, цифрові продукти, різноманітні події, електронні книги, тощо. Також Shopify надає круту можливість створити як і headled магазин, тобто котрий користувач може розробити на сайті за допомогою відповідного конструктора, так і headless та використати API Shopify у себе в застосунку. Ще однією важливою можливістю є створення як звичайного B2C магазину, так і B2B магазину. В останньому оновлені Shopify навіть додав нову безкоштовну тему, розроблену спеціально для B2B магазинів, та їхніх потреб. І звісно ж неможливо не згадати про величезну кількість плагінів, та можливість самостійно розробляти плагіни, котрі можливо буде й надалі використовувати у інших магазинах створених за допомогою цієї платформи.

5. Wix. Wix – це платформа за принципом "що бачиш, те й отримуєш". Коли користувач реєструється як власник інтернет-магазину, йому буде поставлено кілька запитань. По-перше, він запитає, який тип інтернет-магазину хоче створити користувач, і запропонує кілька шаблонів на основі вибору. Потім майбутнього власника магазину буде перенаправлено на

інформаційну панель, де потрібно буде відповісти на кілька інших запитань, зокрема про те, який тип товарів буде продавати користувач. Wix також дає багато свободи для початківців. На вибір є понад 800 шаблонів, які надаються безкоштовно згідно тарифного плану у Wix. Шаблони поділені на категорії для різних галузей. Або ж можна скористатися конструктором Wix, щоб швидко опублікувати свій сайт з кастомним дизайном. Не менш важливим є доставка. Wix має такі можливості, вони є досить широкими і охоплюють майже все. Користувач зможе налаштувати доставку за регіонами та різними типами доставки. Звісно ж невід'ємною частиною є і просування. І Wix доволі круто просунувся порівняно з іншими, адже за допомогою адмін-панелі можна налаштувати рекламу та шеринг контенту у Google, Instagram, Facebook, тощо. І останнім про що б треба було б сказати у контексті створення сайту електронної комерції це інтеграція платіжних сервісів. Wix Payments – це власний платіжний сервіс Wix. Він дозволяє приймати платежі кредитними картками та іншими способами, такими як Apple Pay, Google Pay, iDEAL, Pay Now, тощо, залежно від місцезнаходження. Платежі можливо приймати з усіх основних кредитних карток: Visa, Mastercard, Discover, CUP, JCB, Maestro, тощо. Крім того, налаштування навіть припускають, що власник може приймати платежі готівкою, та навіть реалізована функція накладеного платежу.

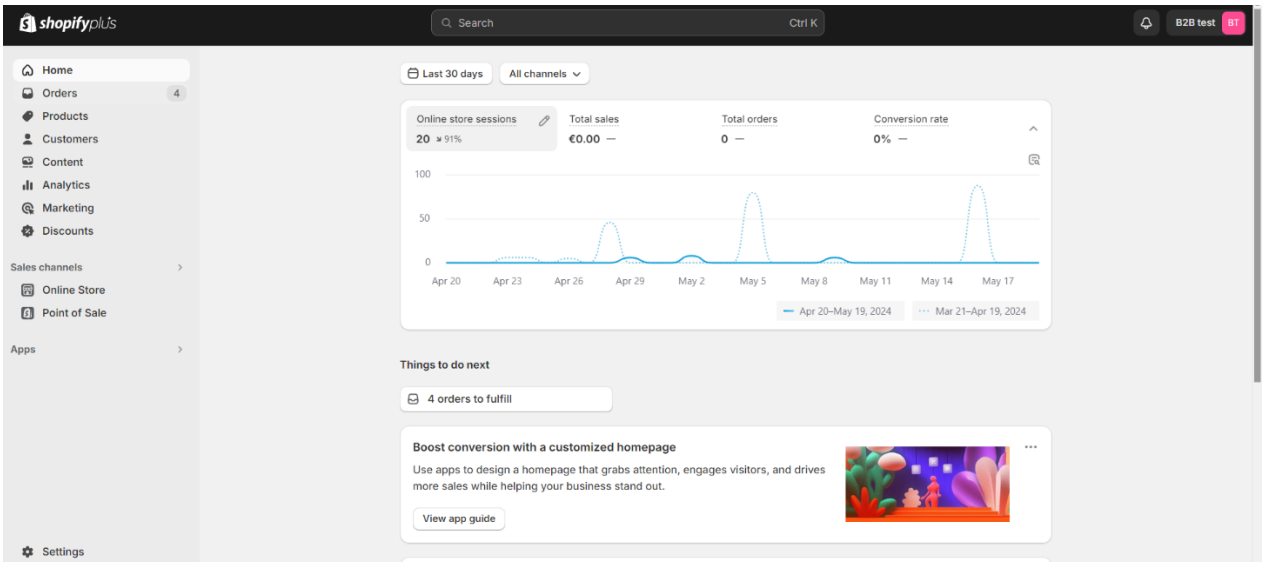


Рисунок 1.8 – Адмін-панель Shopify

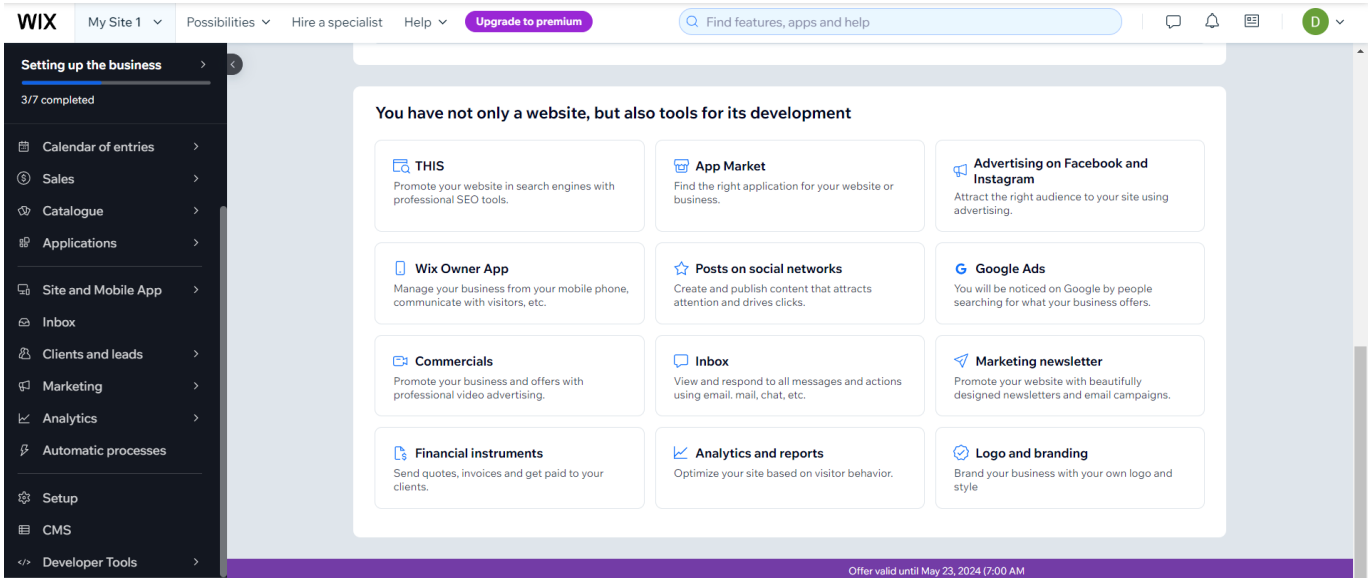


Рисунок 1.9 – Адмін-панель Wix

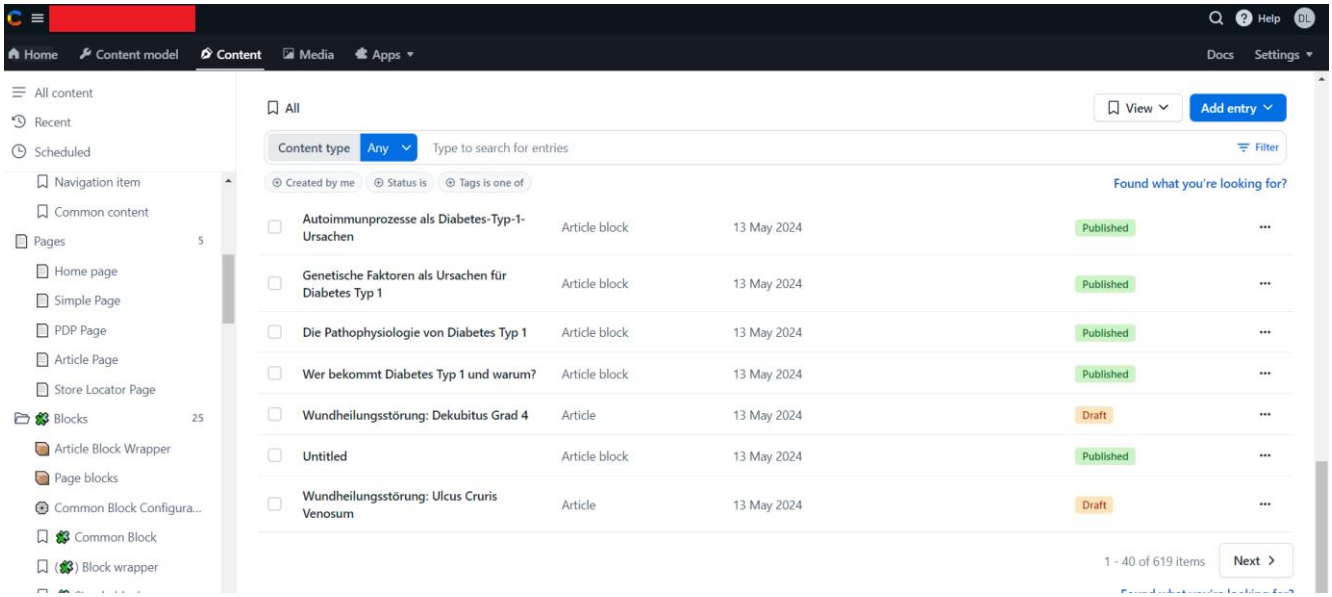


Рисунок 1.10 – Приклад адмін-панелі Contentful

1.6 Завдання і мета роботи

Метою роботи є побудова безголової системи менеджменту контенту для компанії “TopEuroGoods” з детальним опрацюванням розробки CMS та відповідного веб-сайту для електронної комерції. Застосунок має працювати

швидко, мати гарний дизайн для того, щоб користувачі з відділу, котрий відповідає за контент мали змогу швидко та зручно поставляти новий контент, таким чином підвищуючи продажі бізнесу. Також система має мати зручний інструмент для адміністрування користувачів веб-сайту, для перегляду даних для доставки, статусу оплати. Звісно ж потрібно створити зручний інструмент з графіками для відстеження росту продажів.

Для досягнення поставленої мети необхідно виконати наступні завдання:

1. Ознайомитися з існуючими CMS та спектром їх можливостей.
2. Ознайомитись з системами менеджменту контенту для електронної комерції.
3. Провести аналіз потреб до власної системи адміністрування контенту, враховуючи специфіку використання.
4. Ліквідувати недоліки найбільш відомих систем керування контенту при розробці власного продукту.
5. Розробити архітектуру CMS, включаючи модулі, компоненти та їх взаємозв'язки.
6. Обрати технології для створення системи менеджменту контенту.
7. Провести мануальне тестування усіх компонентів розробленої системи.
8. Оцінити працездатність та зручність користування розробленою CMS.
9. Розгорнути створену систему керування контентом у мережі Інтернет.
10. Розробити архітектуру сайту електронної комерції.
11. Розробити веб-додаток використовуючи створену CMS.
12. Мануально протестувати створений веб-сайт за допомогою системи менеджменту контенту.
13. Розгорнути створений веб-додаток в мережі Інтернет.
14. Наповнити контентом веб-сайт використовуючи розроблену раніше CMS.

В результаті виконання дипломної роботи очікується отримати:

- Глибокі знання про існуючі CMS та їх можливості.
- Чітко визначені вимоги до власної системи адміністрування контенту.
- Розроблену архітектуру та основні модулі власної CMS системи.
- Працюючий продукт у вигляді власної CMS системи.
- Результати тестування та оцінки працездатності власної CMS системи.
- Результати тестування та оцінки працездатності створеного веб-додатку на базі розробленої системи адміністрування контенту.

2 РОЗРОБКА АПАРАТНОЇ ЧАСТИНИ

2.1 Огляд структури і системи

Як відомо комп’ютерна мережа це практика з’єднання декількох пристроїв для спільного використання ресурсів та інформації. Вона передбачає використання протоколів, обладнання та програмного забезпечення для встановлення та підтримки цих з’єднань. І безумовним фактом є те, що ні одна комп’ютерна мережа не може обійтись без апаратури, адже це є її основою. Теперішні комп’ютерні мережі мають виконувати неабияку кількість функцій для бізнесу, як малого або середнього, так і для великого.

Створення комп’ютерної мережі вимагає великої підготовки та знання правильних компонентів, що використовуються. Одним з перших кроків у створенні комп’ютерної мережі є визначення того, яке обладнання використовувати і де його розмістити. Сюди входить вибір відповідного обладнання, такого як маршрутизатори, комутатори концентратори, тощо. Також важливим є і відповідне програмне забезпечення, таке як операційна системи, програми та мережеві служби.

Згідно зі структурою компанії (див. рис. 1.1), комп’ютерна мережа буде складатися з п’яти підмереж, а саме:

1. Підмережа фінансового відділу, задачами котрої буде контроль фінансів компанії, складання планів фінансування, проведення аудитів, перевірка і аналіз точності фінансових звітів, фінансова аналітика, розробка кредитної політики, банківські операції, нарахування заробітної плати.
2. Підмережа промоутерського (маркетингового) відділу, задачами котрого буде просування аккаунтів компанії у соціальних мережах, створення креативів, щоб привертати більше уваги користувачів, пошук блогерів для колаборації.
3. Відділ кадрів також має свою підмережу та його задачами є: керівництво персоналом, розповсюдження навчальної інформації щодо цінностей компанії, звітність по наданих послугах, прийом на роботу спеціалістів.

4. Підмережа логістичного відділу.
5. Підмережа відділу підтримки.

Кожна з цих локальних підмереж повинна мати доступ до всесвітньої мережі Інтернет. Також потрібно зауважити, що згідно з поставленим завданням кваліфікаційної роботи, підмережі повинні мати наступну кількість доступних IP-адрес: LAN1 – 36, LAN2 – 81, LAN3 – 164, LAN4 – 84, LAN5 – 58.

2.2 Вимоги до патентної чистоти

Перед тим як говорити про патентну чистоту компанії потрібно розібратись із цим поняттям. Патентна чистота – це юридична властивість технічного об’єкта, яка вказує на те, що він не містить елементів, які мають чинні патенти на винаходи, корисні моделі або промислові зразки, а також чинні свідоцтва на торговельні марки, що належать іншим особам [18].

Вимоги до патентної чистоти компанії, яка у цьому випадку торгує одягом з Європи, мають бути ретельно опрацьовані, щоб забезпечити повний захист інтелектуальної власності та дотримання всіх відповідних нормативних актів. У цьому контексті необхідно розглянути декілька ключових аспектів.

По-перше, компанія повинна визначити перелік країн, відносно яких забезпечуватиметься патентна чистота системи та її частин. Це включає країни Європейського Союзу, такі як Німеччина, Франція, Італія, Іспанія та Польща, а також Великобританію, котра покинула ЄС і Україну, адже саме з країн блоку Європейського союзу будуть завозитись відповідні товари до України.

Важливо також переконатися, що кожен елемент системи, включаючи програмне забезпечення, обладнання та бізнес-процеси, не порушують патентних прав в Україні. Для слідкування за цими пунктами, потрібно здійснювати закупку сертифікованого обладнання. Стосовно програмного забезпечення, воно має також бути ліцензійним, або ж дозволяється використання програм з відкритим вихідним кодом, якщо це перевірене рішення, що не несе за собою ризиків.

2.3 Вимоги до функцій, що має виконувати комп’ютерний комплекс

При розробці комп’ютерної системи необхідно враховувати всі вимоги до функцій, які повинна виконувати система. Комп’ютерний комплекс має відповідати ряду критичних вимог для забезпечення своєї ефективності та надійності. Першим і основним є швидкість відгуку на запити користувача, вона повинна бути високою, що дозволить користувачам швидко отримувати доступ до необхідної інформації та ресурсів. Це критично для підтримки продуктивності та задоволення користувачів. Наступним є швидкість передачі даних, особливо для великих обсягів інформації та високих навантажень на мережу. Це забезпечує плавний та безперервний обмін даними між усіма компонентами системи.

Конфіденційність даних є напевно найважливішою складовою та повинна бути забезпечена на всіх рівнях системи, щоб захистити особисту інформацію клієнтів та комерційну таємницю компанії. Це включає використання сучасних методів шифрування та інших технологій безпеки.

Надійність системи повинна бути також високою, що означає мінімізацію часу простою та забезпечення безперебійної роботи, що особливо актуально для бізнесу в умовах відключення електроенергії і повномасштабного вторгнення росії на територію України. Це досягається за допомогою резервування важливих компонентів та постійного моніторингу стану системи.

Основними вимоги до функціоналу комп’ютерної мережі є:

- Збір даних про клієнтів, що дозволяє накопичувати та аналізувати інформацію для покращення обслуговування та маркетингових стратегій. Це важливо для побудови довгострокових відносин з клієнтами та підвищення їхньої задоволеності.
- Збір даних про співробітників, що забезпечує управління кадровими ресурсами та оптимізацію робочих процесів. Ця функція сприяє ефективному керуванню персоналом та підвищенню продуктивності праці.
- Зберігання баз даних має бути надійним і захищеним, що гарантує збереження та швидкий доступ до великої кількості даних. Це включає

використання сучасних серверів та технологій для забезпечення цілісності даних.

- Доступ до міжнародної мережі Інтернет є необхідним для забезпечення глобальної комунікації та обміну інформацією. Це дозволить співробітникам отримувати доступ до зовнішніх ресурсів, необхідних для виконання їхніх завдань.
- Обмін даними між співробітниками повинен бути швидким та надійним, що сприяє ефективній командній роботі та оперативному вирішенню завдань. Це включає використання внутрішніх комунікаційних засобів та мережевих сервісів.
- Доступ до спільних файлів забезпечить зручність у спільній роботі над проектами та документацією. Це включає можливість спільного редагування, зберігання та обміну файлами у реальному часі.

Таким чином, комп'ютерний комплекс компанії має відповідати всім зазначеним вимогам для забезпечення ефективної, надійної та безпечної роботи.

2.4 Вимоги до програмного забезпечення

При розробці та впровадженні програмного забезпечення для компанії, котра торгує одягом з Європи, необхідно враховувати різноманітні функціональні та нефункціональні вимоги для забезпечення ефективної роботи всієї системи. Основною вимогою до програмного забезпечення звісно ж буде функціональність, адже програмне забезпечення повинно відповідати всім функціональним потребам бізнесу, включаючи управління замовленнями, облік товарів, обробку платежів, управління клієнтами, логістику та звітність. Інтерфейс користувача повинен бути інтуїтивно зрозумілим і зручним у використанні, що б дозволяло співробітникам легко виконувати свої завдання.

Наступною і важливою вимогою стала надійність, бо комплекс повинен забезпечувати високу надійність і мінімальний час простою. Це включає стійкість до збоїв та можливість швидкого відновлення після аварій. Регулярні резервні копії

та механізми відновлення даних повинні бути інтегровані у систему.

Безпека також є фундаментом при підборі програмного забезпечення для комп'ютерного комплексу. Захист від несанкціонованого доступу, шифрування даних та управління правами доступу, регулярні оновлення безпеки та тестування на вразливості повинні бути заплановані та виконані.

Масштабованість навіть в контексті програмного забезпечення є необхідною, адже комплекс повинен бути здатний до масштабування у відповідь на зростання бізнесу, включаючи збільшення обсягу даних, кількості користувачів та обсягу транзакцій.

Останнім же є продуктивність і вона є критичною, бо забезпечує швидке оброблення даних та мінімальні затримки у відповідях на запити користувачів. Оптимізація коду та баз даних для забезпечення ефективного використання ресурсів також необхідні.

2.5 Вибір і опис структурної схеми комплексу

Комп'ютерний комплекс компанії представлений у вигляді п'яти локальних підмереж, з'єднанихміж собою комутаторами і маршрутизаторами. Комплекс спеціально розроблений для підтримки роботи компанії, що займається продажем одягу з Європейського Союзу. Роутери Lytvynenko Router 1-6 забезпечують з'єднання між різними підсистемами і є основними точками маршрутизації трафіку. Lytvynenko Router IPS є ж маршрутизатором, котрий забезпечує вихід в мережу Інтернет.

Важливу роль у цьому комплексі відіграють протоколи. Serial протокол використовується для з'єднання між основними маршрутизаторами, забезпечуючи швидку передачу великої кількості даних. Gigabit Ethernet же використовується для з'єднання між маршрутизаторами та комутаторами, що дозволяє передавати великі обсяги даних без втрати якості.

Fast Ethernet є найбільш популярним у мережі, адже саме за цим протоколом відбувається з'єднання між комутаторами та хостами, а хостів як раз таки завжди

більше ніж комутаторів та маршрутизаторів.

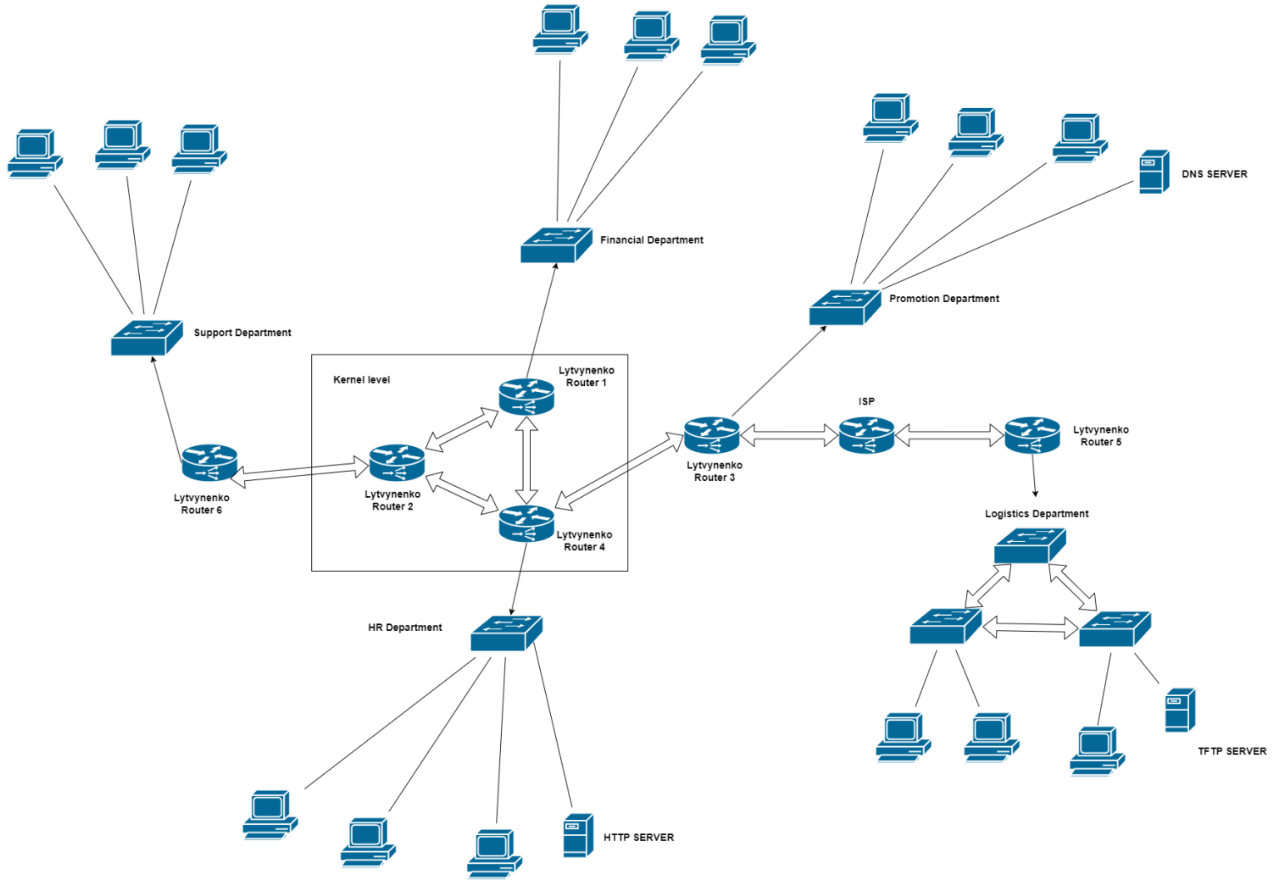


Рисунок 2.1 – Структурна схема технічних засобів комп’ютерного комплексу

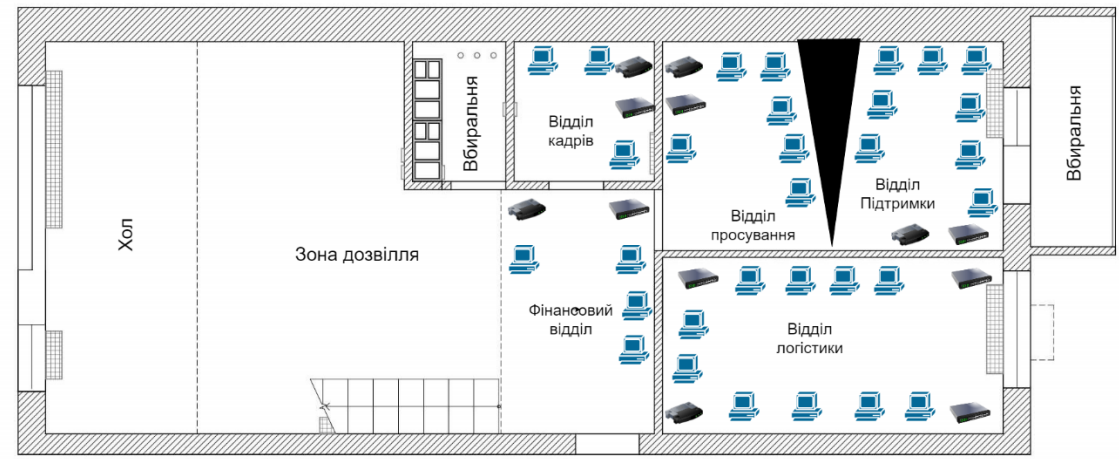


Рисунок 2.2 – Схема будівлі в котрій розташований комп’ютерний комплекс

2.6 Апаратні засоби комп’ютерного комплексу

У цьому розділі потрібно розглянути вибір маршрутизатора та комутатора. Розпочнемо з комутатора. Вибір пав на комутатори серії Cisco Catalyst 2960. Зазвичай ці комутатори використовуються як раз таки в інфраструктурних мережах середнього розміру. Ця серія пропонує пропускну здатність при пересиланні до 100-108 Гбіт/с і пропускну здатність при комутації в режимі повного дуплексу до 216 Гбіт/с. Ці комутатори мають граничний рівень 2, що забезпечує підвищену простоту використання, покращену стійкість, високу безпеку бізнес-операцій, а також безмежну мережеву взаємодію. Крім того, комутатори серії Catalyst 2960 підтримують новітню технологію стекування FlexStack з можливістю підключення до одного і десяти гігабітних портів, що може значно підвищити комфорт і продуктивність роботи підприємства. Він також має нову технологію «Power over Ethernet Plus» або PoE+ і пропонує швидкий доступ до підключення Ethernet і можливості PoE.

Важливо зауважити і такі властивості цієї серії комутатора Cisco, як здатність до резервного копіювання. У зв’язку з цим важливо знати, що резервне копіювання файлів здійснюється через USB-накопичувач. Також важливо згадати про гарантію на обладнання Cisco Catalyst 2960, адже це напряму впливає на витрати бізнесу. Обмежена довічна гарантія на обладнання включає заміну на наступний робочий день і дев’яносто днів підтримки та обслуговування.

Наступним на розгляд буде модель маршрутизатора, котру комплекс буде використовувати. Cisco 4331 – це тип маршрутизатора, вироблений компанією Cisco і він є частиною серії маршрутизаторів з інтегрованими послугами, які призначені для підприємств середнього та великого бізнесу та філій підприємств. Цей роутер пропонує високу продуктивність, безпеку і прикладні сервіси для підтримки декількох користувачів і мережевих пристроїв. Він зазвичай використовується для підключення до глобальної мережі, передачі голосу і відео та інших мережевих функцій в бізнес-середовищі. Він пропонує високопродуктивні можливості та розширені функції безпеки для забезпечення

безперебійної та безпечної передачі даних в мережі. З цікавого, апаратно, цей маршрутизатор включає в себе багатоядерний процесор, інтегровані слоти для сервісних модулів, вбудовані порти Gigabit Ethernet і підтримку віртуалізації. Також цей вибір є чудовим для підприємств, котрі прагнуть вдосконалити свою мережеву інфраструктуру. Надійна конструкція і масштабованість роблять його придатним для широкого спектру застосувань, від філій до центрів обробки даних.

Таблиця 2.1 – Кількість вузлів

№	Пристрій	Позначення пристрою на топології	Одиниці виміру	Кількість в одиницях виміру
1	Cisco Catalyst 2960 Plus 48 10/100 + 2 T/SFP LAN Lite (WS-C2960+48TC-S)	Support department, HR department, Logistics department, Promotion department, Financial department, Switch02, Switch01	Шт.	7
2	Cisco ISR4331/K9	Lytvynenko Router 6, Lytvynenko Router 5, Lytvynenko Router 4, Lytvynenko Router 3, Lytvynenko Router 2, Lytvynenko Router 1, Lytvynenko Router ISP	Шт.	7

2.7 Розрахунок інтенсивності трафіку вихідного трафіку найбільшої локальної мережі підприємства

Щоб провести правильний розрахунок інтенсивності трафіку, треба зважати, що комплекс має бути завантажений на повну потужність. Комутатор Cisco Catalyst 2960 Plus об’єднує до 48 ПК. Після цього потрібно звернути увагу на наступні

показники: найбільша кількість вузлів = 164; середній показник інтенсивності трафіку згідно варіанту: μ = 107(кадрів/с); розмір повідомлення в середньому = 850 байт; передача пакету не повинна перевищувати ≤8 мс. Після того як трафік виходить з комп'ютера користувача від перенаправляється на маршрутизатор зі швидкістю 1000 Мбіт/с. Під час розрахунків враховуємо, що послугами користуються всі 100% користувачів. Отже почати потрібно з пропускної здатності комп'ютерної мережі на рівні доступу, вона обчислюється так:

$$Pp.д=\mu*l*n*8=107*850*48*8 \approx 34,9 \text{ Мбіт/с}$$

На рівні розподілу пропускна спроможність мережі розраховується за допомогою максимальної кількості комутаторів на рівні розподілу та загальної чисельності користувачів. Обчислимо пропускну здатність:

$$Pp.p= \mu*l*N*8=107*850*164*8 \approx 119,33 \text{ Мбіт/с}$$

В цій же формулі N є кількістю вузлів в найбільшій мережі. Після обчислень зрозуміло, що результати не переважають параметри комп'ютерної мережі, з чого випливає, що навантаження на обраній ділянці мережі не буде. Загальне навантаження на комутатор не повинно перевищувати:

$$\mu_{вих} =1\ 000\ 000\ 000/(850*8) \approx 147\ 058 \text{ пакетів/с.}$$

Оскільки в середньому, кожне джерело виробляє 107 пакетів/с, то маршрутизатор обмежений кількістю приєднань, а саме:

$$N = \mu_{вих} / \mu = 147\ 058 / 107 \approx 1374 \text{ джерела}$$

Ця кількість задовольняє кількості вузлів у найбільшій локальній мережі. Далі розглянемо, коли кожен з 48 ПК посилає потік заявок з інтенсивністю у 107 кадрів/с. Знайдемо інтенсивність вихідного трафіку:

$$\lambda = N * \mu = 48*107 = 5136 \text{ пакетів/с.}$$

Розрахуємо коефіцієнт затримки:

$$\rho = \frac{\lambda}{\mu вих} = \frac{5136}{147058} \approx 0,035$$

Коефіцієнт зайнятості маршрутизатора:

$$\frac{\rho}{1-\rho} = \frac{0,035}{1-0,035} \approx 0,036$$

Середня затримка кадру, пов'язана з чергою M/M/1, дорівнює:

$$T = \frac{1}{(\mu - \lambda)} = \frac{1}{(147058 - 5136)} \approx 7{,}046 \text{ мкс}$$

Середня довжина черги:

$$L_{чер} = \frac{\rho^2}{1-\rho} = \frac{0{,}035^2}{1-0{,}035} \approx 0{,}0013$$

Середній час перебування пакета у черзі:

$$T_{оч} = \frac{Lчер}{\lambda} = \frac{0{,}0013}{5136} \approx 25 \text{ мкс}$$

Це значення задовольняє вимогам до затримки.

3 РОЗРОБКА КОРПОРАТИВНОЇ МЕРЕЖІ

Розпочати потрібно з планування мережі. Згідно завданню, отриманим від замовника, блок адрес та кількість вузлів, які буде використано у подальшій розробці комп'ютерної системи, виглядають наступним чином:

Таблиця 3.1 – Кількість вузлів

Блок адрес	LAN 1	LAN 2	LAN 3	LAN 4	LAN 5
10.25.72.0/22	36	81	164	84	58

Таблиця 3.2 – Розрахунок корпоративної мережі

Довжина	Ім'я	К-сть	IP мережі	Маска	Перша адреса	Остання адреса
Найбільша	LAN3	254/164	10.25.72.0	/24	10.25.72.1	10.25.72.254
	LAN4	126/84	10.25.73.0	/25	10.25.73.1	10.25.73.126
	LAN2	126/81	10.25.73.128	/25	10.25.73.129	10.25.73.254
	LAN5	126/58	10.25.74.0	/25	10.25.74.1	10.25.72.126
Найменша	LAN1	126/36	10.25.74.128	/25	10.25.74.129	10.25.74.254

Згідно з завданням, для зв'язку між маршрутизаторами було використано блок адрес 10.1.9.0 /24.

Таблиця 3.3 – Схема адресації мережі

Назва мережі	К-сть вузлів	Номер мережі	Маска мережі	Початкове значення діапазону можливих	Кінцеве значення діапазону можливих

				адрес вузлів у підмережі	адрес вузлів у підмережі
LAN3	254/164	10.25.72.0	/24	10.25.72.1	10.25.72.254
LAN4	126/84	10.25.73.0	/25	10.25.73.1	10.25.73.126
LAN2	126/81	10.25.73.128	/25	10.25.73.129	10.25.73.254
LAN5	126/58	10.25.74.0	/25	10.25.74.1	10.25.74.126
LAN1	126/36	10.25.74.128	/25	10.25.74.129	10.25.74.254
WAN 1	2/2	10.1.9.0	/30	10.1.9.1	10.1.9.2
WAN 2	2/2	10.1.9.4	/30	10.1.9.5	10.1.9.6
WAN 3	2/2	10.1.9.8	/30	10.1.9.9	10.1.9.10
WAN 4	2/2	10.1.9.12	/30	10.1.9.13	10.1.9.14
WAN 5	2/2	10.1.9.16	/30	10.1.9.17	10.1.9.18
WAN 6	2/2	10.1.9.20	/30	10.1.9.21	10.1.9.22
ISP 1	2/2	209.165.202.0	/30	209.165.202.1	209.165.202.2
ISP 2	2/2	64.100.13.0	/30	64.100.13.1	64.100.13.2

Серверам було привласнено IP-адреса за правилом: IP-адрес дорівнює першому можливому адресу у мережі+9+№, де № − номер варіанту студента за списком у групі.

Таблиця 3.4 – Адресація інтерфейсів серверів

Назва серверу	Назва інтерфейсу	IP–адреса	Маска	Шлюз
Server HTTP	Fa0	10.25.73.148	25	10.25.73.129
Server DNS	Fa0	10.25.74.19	25	10.25.74.1
Server TFTP	Fa0	10.25.72.19	25	10.25.72.1

Після проведеної підготовки було проведено побудову мережі, налаштовано NAT, PAT,VPN, тощо. І звісно ж для захисту було налаштовано VLAN, що є групою

хостів з загальним набором вимог, що взаємодіють так, ніби вони прикріплені до одного домену, незалежно від їх фізичного розташування. Основною причиною поділу мережі на мережі VLAN є скорочення навантажень у великій локальній мережі. Також важливою особливістю використання VLAN є те, що це допомагає економити підприємству, компанії на мережевому обладнанні, а головне це просто зручніше, адже усі налаштування можна виконати на відповідному комутаторі. Далі розглянемо типи віртуальних локальних мереж. Вони існують різні і мають свої особливості. В основному їх поділяють на:

- Типовий VLAN. Коли комутатор спочатку завантажується, всі порти комутатора додаються до VLAN за замовчуванням. Як правило, всі комутатори мають VLAN за замовчуванням як VLAN 1. VLAN1 дозволяє будь-якому мережевому пристрою, підключеному до будь-якого порту комутатора, підключатися до інших пристроїв на інших портах комутатора. Цей VLAN1, тобто VLAN за замовчуванням, не може бути перейменований або видалений.
- VLAN даних. VLAN даних також відомий як VLAN користувача. VLAN даних використовується тільки для даних, що генеруються користувачами. Цей VLAN несе тільки дані. Він не несе управління трафіком або голосом.
- Голосовий VLAN. Цей VLAN налаштований для передачі голосового трафіку. Голосові VLAN зазвичай мають вищий пріоритет передачі, ніж більшість інших типів мережевого трафіку. Це потрібно для зберігання трафіку для інших додатків.
- VLAN управління. VLAN керування використовується для налаштування комутатора для цілей керування. Він управляє системою реєстрації та моніторингу.
- Native VLAN. Цей VLAN використовується для передачі трафіку без тегів.

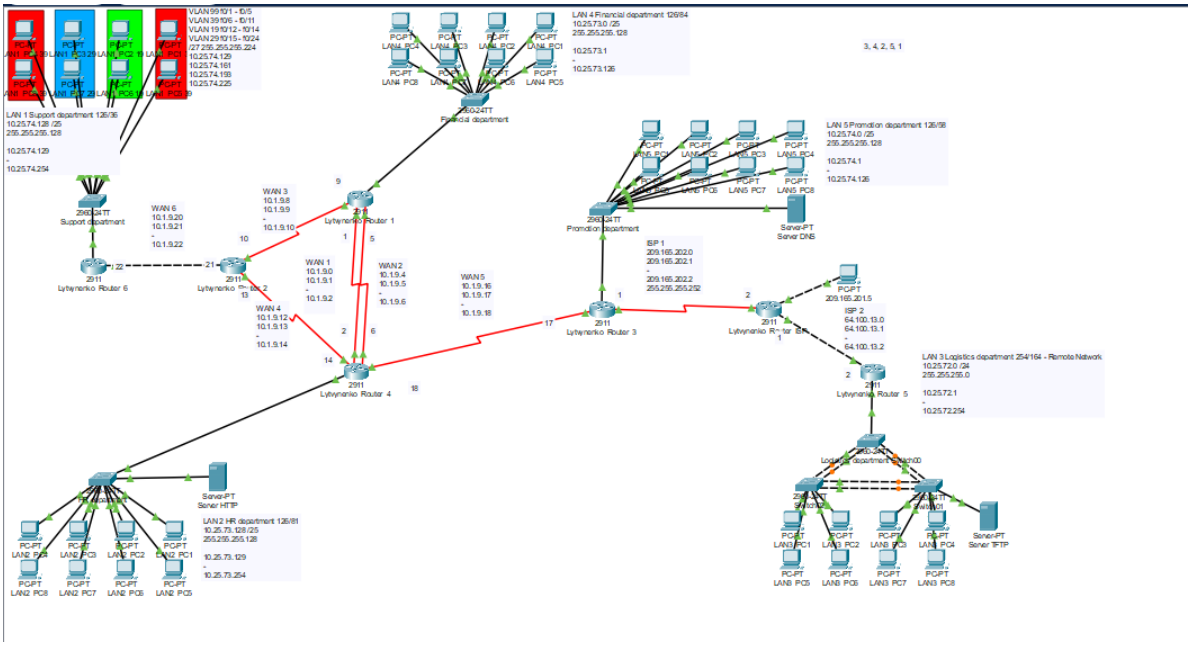


Рисунок 3.1 – Побудована та налаштована топологія

```
Lytvynenko_Switch_LAN_1>show vlan brief

VLAN Name                              Status    Ports
---- --------------------------------- --------- -------------------------------
1    default                           active    Fa0/1, Fa0/2, Fa0/3, Fa0/4
                                                 Fa0/5, Gig0/2
19   Accounting                        active    Fa0/12, Fa0/13, Fa0/14
29   Resources_Department              active    Fa0/15, Fa0/16, Fa0/17, Fa0/18
                                                 Fa0/19, Fa0/20, Fa0/21, Fa0/22
                                                 Fa0/23, Fa0/24
39   Guest                             active    Fa0/6, Fa0/7, Fa0/8, Fa0/9
                                                 Fa0/10, Fa0/11
99   Management                        active
100  Native                            active
1002 fddi-default                      active
1003 token-ring-default                active
1004 fddinet-default                   active
1005 trnet-default                     active
Lytvynenko_Switch_LAN_1>
```

Рисунок 3.2 – Налаштування VLAN на одному з комутаторів мережі

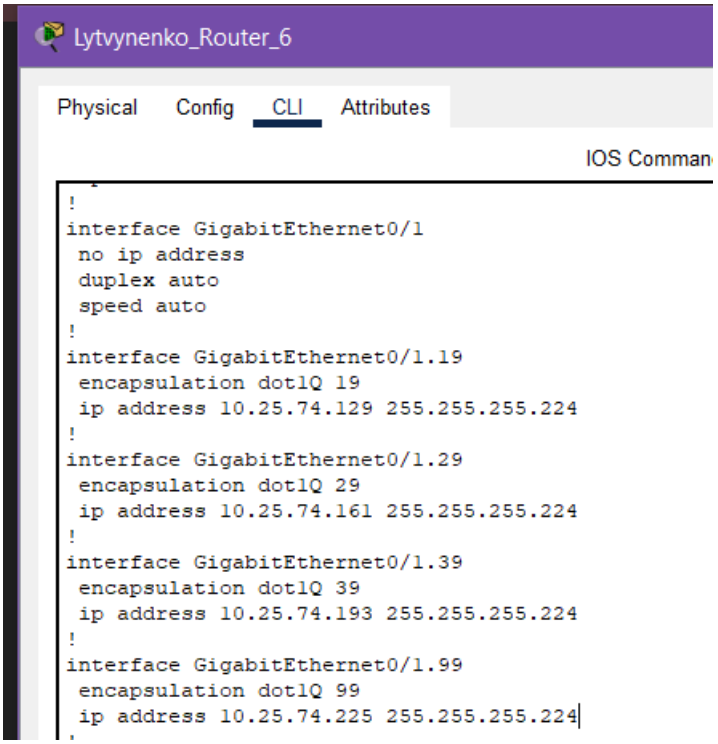


Рисунок 3.3 – Інкапсуляція підінтерфейсів GigabitEthernet за допомогою протоколу 802.1Q

Звісно ж перед цим було важливо провести і базові налаштування роутерів, адже це підвищує безпеку мережі.

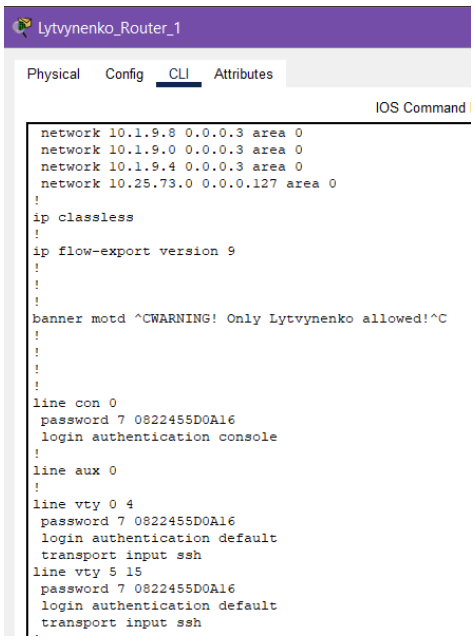


Рисунок 3.4 – Приклад базових налаштувань маршрутизатора

```
Press RETURN to get started!

WARNING! Only Lytvynenko allowed!

User Access Verification
```

Рисунок 3.5 – Налаштований банер

Не забути потрібно і за DHCP, адже кожному пристрою, підключеному до Інтернету, має бути присвоєна унікальна IP-адреса, а робити це власноруч незручно і є шанс присвоїти неправильне значення. DHCP допомагає мережевим адміністраторам контролювати та призначати IP-адреси централізовано. Він може автоматично призначати нову IP-адресу комп'ютеру, коли його переміщують в інше місце. DHCP автоматизує процес розподілу IP-адрес, що скорочує час, необхідний для конфігурації та розгортання пристроїв, а також зменшує ймовірність помилок конфігурації. Крім того, DHCP-сервер може керувати конфігураціями декількох сегментів мережі. Коли конфігурація сегмента мережі змінюється, адміністратору потрібно лише оновити відповідну конфігурацію на DHCP-сервері.

```
ip dhcp excluded-address 10.25.73.0 10.25.73.10
!
ip dhcp pool poollan4
 network 10.25.73.0 255.255.255.128
 default-router 10.25.73.1
 dns-server 10.25.74.19
!
```

Рисунок 3.6 – Приклад налаштувань пулу адрес для dhcp

Не забути потрібно і про налаштування протоколу OSPF, але для цього потрібно розуміти навіщо він потрібен. Перш за все потрібно дати відповідь на те, що робить мережа? Мережа дозволяє комп'ютерам спілкуватися один з одним, пересилаючи пакети даних від маршрутизатора до маршрутизатора, поки пакети не досягнуть кінцевого пункту призначення. І комп'ютер з однієї мережі має знати як дістатися до іншої мережі, бо інакше він відкине пакети. Як правило, всі маршрутизатори в мережі повинні знати, як дістатися до кожної IP-підмережі. Для цього маршрутизатори встановлюють IP-маршрути у свої таблиці маршрутизації.

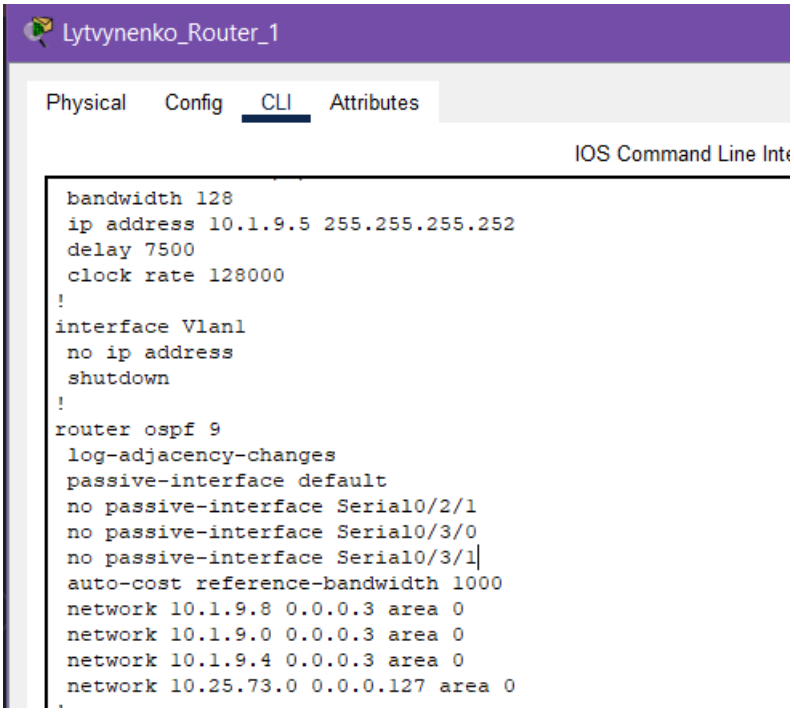


Рисунок 3.7 – Приклад налаштування маршрутів на одному з роутерів

Звісно ж важливим кроком є і налаштування VPN, що захищає своїх користувачів, шифруючи їхні дані та маскуючи їхні IP-адреси. Це приховує їхню активність в Інтернеті, особистість і місцезнаходження, забезпечуючи більшу конфіденційність і автономію.

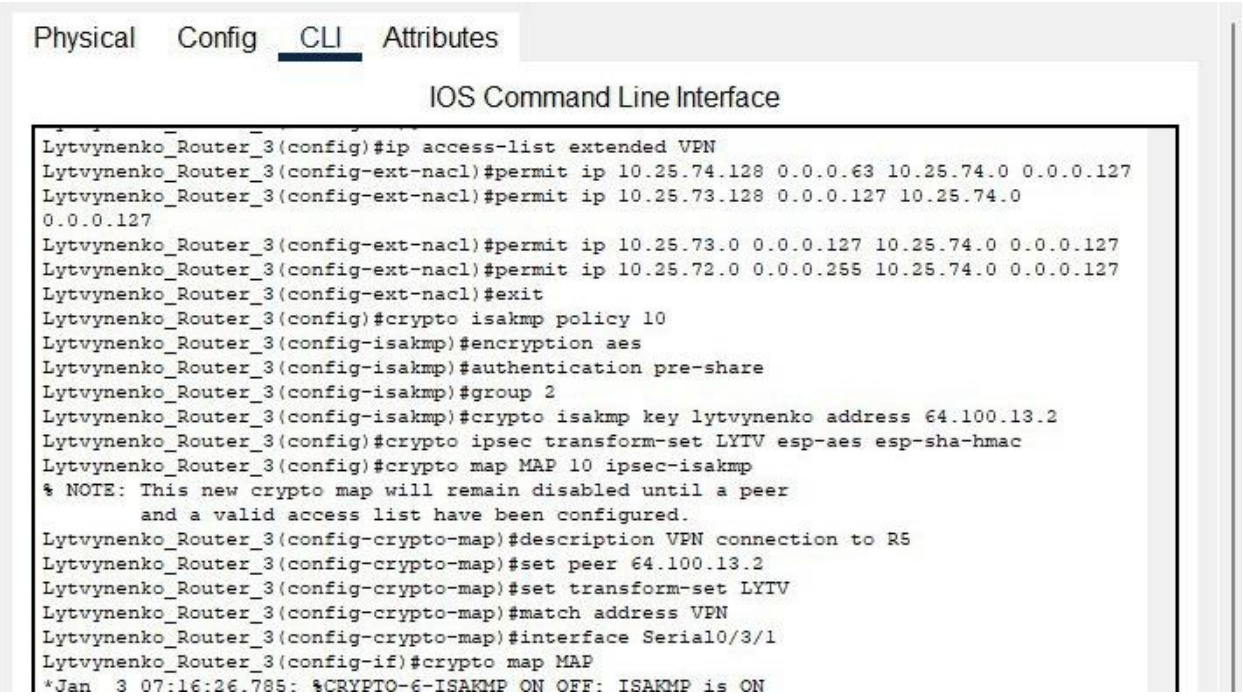


Рисунок 3.8 – Конфігурація маршрутизатора з використанням політики Ipsec для захищених з’єднань VPN

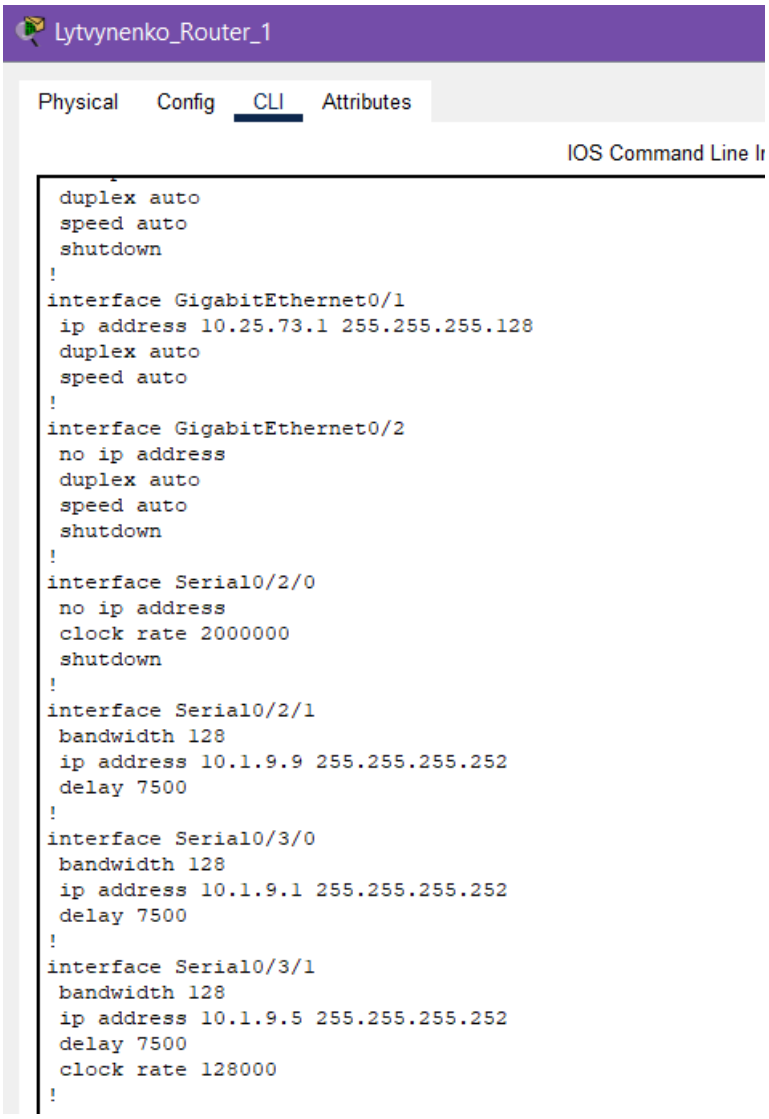


Рисунок 3.9 – Налаштування Serial інтерфейсів на одному з роутерів

Останнім кроком стало налаштування HTML файлу для http сервера, а саме верстка титулу дипломної роботи.

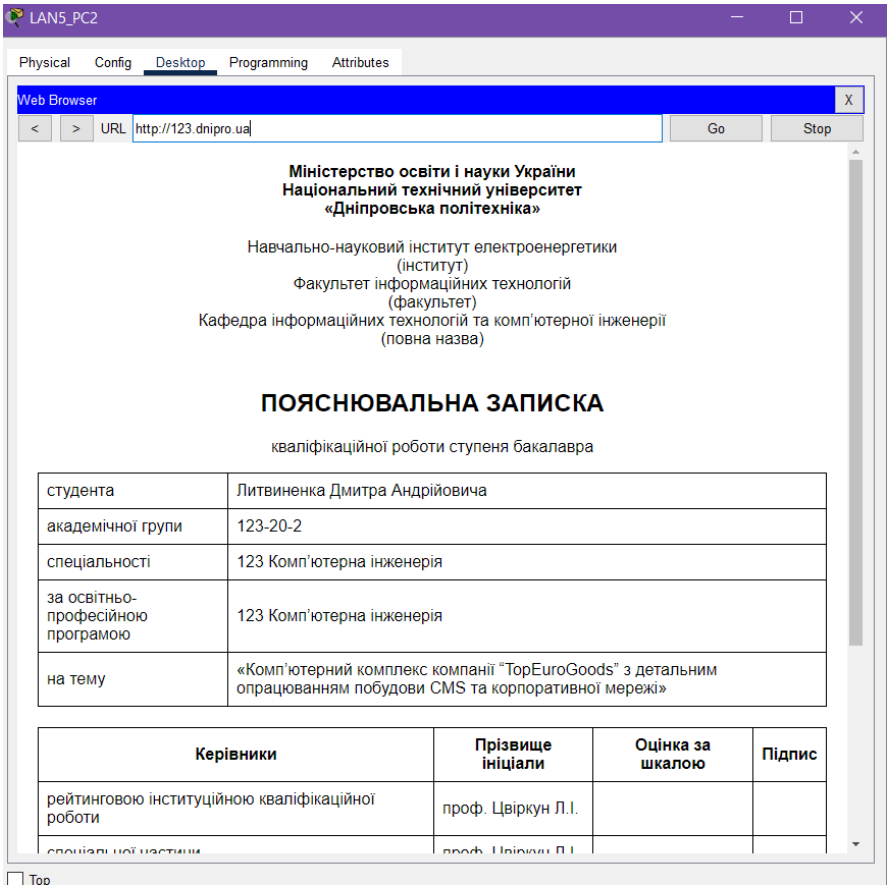


Рисунок 3.10 – Початок зміненого html файлу на http сервері

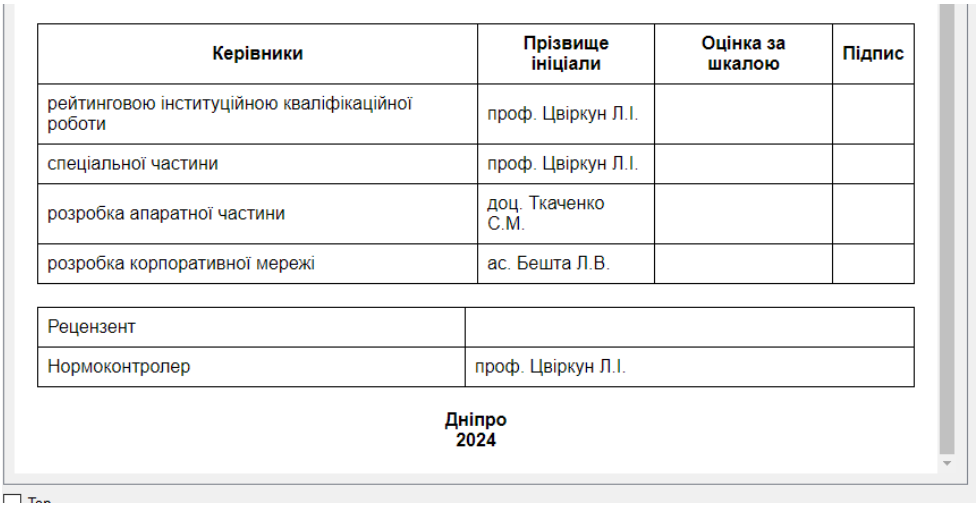


Рисунок 3.11 – Кінець зміненого html файлу на http сервері

4 РОЗРОБКА КОМПОНЕНТА СИСТЕМИ

4.1 Обґрунтування технічних характеристик

4.1.1 Вибір frontend фреймворку для створення CMS та веб-сайту

Створення веб-сайту вимагає значного обсягу технічних знань і навичок розробки. Щоб спростити веб-розробку, frontend-розробники часто використовують фреймворки.

Фреймворки відіграють важливу роль у веб-розробці, оскільки вони допомагають структурувати та спростити процес створення веб-сайту. Frontend - розробники використовують ці фреймворки для швидкого створення графічного інтерфейсу користувача веб-сайту і створення односторінкових веб-додатків (вони ж Single-page application), котрі працюють без перезавантажень, як звичайні додатки.

Веб-сайт, розроблений за допомогою фреймворку, як правило, більш ефективний і безпечний, оскільки його код є уніфікованим і послідовним. Крім того, фреймворки часто постачаються з багатьма компонентами, які можна використовувати повторно, що економить час і ресурси на розробку.

Перш ніж заглибитися в деталі найкращих фреймворків для frontend, треба спочатку визначити, що вони собою являють. Щоб зрозуміти, що таке frontend-фреймворки, спочатку потрібно знати, що таке інтерфейс. Frontend веб-сайту або додатку – це клієнтська частина, з котрою взаємодіють користувачі. Сюди входять дизайн, верстка, призначений для користувача інтерфейс і будь-які інтерактивні елементи, такі як форми і кнопки. Фреймворки інтерфейсу – це набори заздалегідь написаного коду, які надають розробникам масштабовану і підтримувану структури для більш ефективного створення користувацьких інтерфейсів. Вони містять компоненти HTML, CSS та JavaScript, які розробники можуть повторно використовувати в інших проєктах, допомагаючи підтримувати кодову базу узгодженою та організованою. Ці фреймворки для веб-розробки надають структуру процесу розробки і полегшують написання послідовного і добре організованого коду. Деякі з завдань, з якими допомагають впоратися фреймворки інтерфейсу,

включають в себе:

- Компонентна архітектура: frontend-фреймворки використовують компонентну архітектуру, де інтерфейс розбивається на багаторазові компоненти. Такий модульний підхід спрощує розробку та підтримку.
- Віртуальний DOM: багато фреймворків використовують віртуальний DOM (Document Object Model) для ефективного оновлення інтерфейсу без повторного рендерингу всієї сторінки. Це покращує продуктивність і забезпечує більш плавний користувацький досвід.
- Управління станами: frontend-фреймворки пропонують надійні рішення для управління станами, що дозволяє розробникам більш ефективно керувати даними та станом інтерфейсу.
- Маршрутизація: фреймворки надають можливості маршрутизації, що дозволяє створювати односторінкові додатки з декількома поданнями або сторінками, які динамічно оновлюються на основі взаємодії з користувачем.
- Стилізація: більшість фреймворків мають вбудовану підтримку стилів, включаючи препроцесори CSS, такі як SASS або LESS, і пропонують рішення для управління стилями в масштабі.
- Інструментарій та інтеграція з DevOps: frontend-фреймворки часто включають інструменти для розробки, налагодження та тестування, а також інтеграцію з популярними практиками DevOps, такими як безперервна інтеграція та розгортання (CI/CD).

Першим звісно ж треба розглянути React, адже React є найпопулярнішим на ринку. Він розроблений Facebook, також важливо зауважити, що це бібліотека JavaScript для створення користувацьких інтерфейсів, тобто він не фреймворк. Ця бібліотека широко відома завдяки своїй компонентній архітектурі, яка дозволяє розробникам створювати багаторазові компоненти інтерфейсу, що чудово вписується у принципи SOLID. Віртуальний DOM React ефективно оновлює інтерфейс, що призводить до покращення продуктивності. Ключовими особливостями є:

- Компонентна архітектура: React заохочує модульний підхід до розробки

інтерфейсу, де кожен компонент керує власним станом та логікою інтерфейсу.

- Віртуальний DOM: віртуальний DOM у React мінімізує накладні витрати на маніпуляції з DOM, що призводить до швидшого рендерингу та покращення продуктивності. Також він надає інструменти для контролю та оптимізації рендерингу.
- Декларативний синтаксис: React використовує декларативний синтаксис, що полегшує розуміння та підтримку коду.
- Багата екосистема: React має широку екосистему бібліотек та інструментів (наприклад, Redux, React Router DOM), які розширюють його можливості для управління станами, маршрутизації тощо. Також абсолютна кастомізація є надзвичайною перевагою для досвідчених розробників.
- Ком'юніті: широка спільнота React дозволяє з легкістю новачкам вирішувати купу проблем і швидко знайомитись з екосистемою бібліотеки.

Але також React має й недоліки:

- Надзвичайна гнучкість: іноді велика кількість вибору може лише ускладнити архітектурні рішення.
- Швидкі зміни: часті оновлення: У порівнянні з іншими популярними frontend-фреймворками, React швидко оновлюється, і за ним може бути важко встигнути.
- Проблеми з оптимізацією: для великих додатків може знадобитися додаткове налаштування продуктивності.
- Проблеми з CSS: CSS може стати більш заплутаним, ніж використання традиційних таблиць стилів, тому має бути чітко побудована архітектура.

Наступним на черзі на розгляд стане Vue. Це прогресивний JavaScript-фреймворк для створення користувацьких інтерфейсів. Він відомий своєю простотою, гнучкістю та легкістю інтеграції з існуючими проєктами. Vue дозволяє розробникам поступово впроваджувати його функції, що робить його придатним для проєктів будь-якого розміру. Основними перевагами є:

- Реактивне прив'язування даних: Vue забезпечує реактивне зв'язування даних, що дозволяє компонентам інтерфейсу автоматично оновлюватися при зміні даних.
- Розробка на основі компонентів: як і React, Vue підтримує архітектуру на основі компонентів, що дозволяє розробникам створювати багаторазові та комбіновані елементи інтерфейсу.
- Vue CLI: Vue CLI пропонує інтерфейс командного рядка для створення та управління проєктами Vue.js, що дозволяє легко налаштовувати середовища розробки та створювати готові до виробництва додатки.
- Однофайлові компоненти: Vue.js підтримує однофайлові компоненти, де HTML, CSS та JavaScript інкапсульовані в одному файлі, що покращує організацію коду та його читабельність.
- Легкість у вивченні: простий синтаксис і документація Vue роблять його доступним для більшості розробників, особливо у порівнянні з Angular;

Vue.js як і React має свої проблеми, а саме:

- Частка ринку на Заході: менш поширений на ринку праці в Європі.
- Впровадження на підприємствах: не так широко використовується на великих підприємствах.
- Екосистема плагінів: хоча вона зростає, вона може бути не такою широкою, як у більш популярних фреймворків.
- Надмірна гнучкість: гнучкість фреймворку іноді може призвести до неузгодженості кодової бази, якщо ним не керувати належним чином.
- Прихована спільнота: левова частка спільноти також знаходиться у Східній Азії і не розмовляє англійською.

Наступним та останнім великим гравцем у “вічному змаганні” frontend-фрейморків є Angular, котрий розроблений Google. Це комплексний TypeScript-фреймворк для створення веб-додатків. Він пропонує повний набір інструментів і функцій для розробки надійних, масштабованих і підтримуваних додатків. Angular відомий своїми можливостями двостороннього зв'язування даних та ін'єкції

залежностей. Основні можливості:

- Двостороннє зв'язування даних: Angular полегшує двостороннє зв'язування даних, дозволяючи змінам в інтерфейсі автоматично оновлювати базову модель даних, і навпаки.
- Ін'єкція залежностей: система ін'єкції залежностей Angular сприяє створенню модульного та тестованого коду, керуючи залежностями між компонентами та сервісами.
- Angular CLI: Angular CLI надає інтерфейс командного рядка для створення компонентів, сервісів, модулів тощо, що спрощує процес розробки.
- Інтеграція з RxJS: Angular інтегрується з RxJS (Reactive Extensions for JavaScript), що дозволяє розробникам легко створювати реактивні та асинхронні додатки.

Недоліками Angular є:

- Важкість у вивченні: Вимагає знання TypeScript та складних концепцій Angular. Це може здатися особливо складним для розробників, які не знайомі з ООП.
- Багато-умовність і складність: деякі розробники також вважають Angular більш багато-умовним і складним у порівнянні з альтернативами.
- Фреймворк з власною думкою: прописує конкретні способи виконання завдань, які можуть підходити не всім розробникам.
- Початковий час завантаження: Більші розміри пакунків можуть призвести до більшого часу початкового завантаження додатків, що може відлякувати користувачів і знижувати відвідуваність сайту і як наслідок шкодити бізнесу.
- Розділена спільнота: різниця між різними версіями Angular може бути більш значною, ніж між Angular та іншими інтерфейсними веб-фреймворками.

Наступним на розгляд стане Svelte, котрий має революційний підхід до побудови користувацьких інтерфейсів, відрізняється від традиційних frontend-фреймворків JS тим, що переносить більшу частину роботи на час компіляції. Замість використання віртуальної DOM, Svelte пише високоефективний

імперативний код, який безпосередньо оновлює DOM при зміні стану додатку. Цей інноваційний підхід призводить до швидшого початкового завантаження, більш плавного оновлення та спрощення роботи розробника. Реактивність Svelte вбудована в саму мову, що дозволяє розробникам створювати складні функції з меншою кількістю рядків коду. Компактний розмір та ефективність Svelte роблять його привабливим вибором для розробників, які шукають продуктивність без накладних витрат на традиційні фреймворки. Хоча кілька років тому Svelte все ще була експериментальним рішенням, сьогодні, завдяки зростаючій спільноті та екосистемі, Svelte набирає обертів і стає доволі вагомим інструментом. Тож, розглянемо переваги цього фреймворку:

- Відсутність віртуального DOM: пряма маніпуляція з DOM для оновлень, що призводить до чудової продуктивності.
- Менша кількість коду: зазвичай вимагає менше рядків коду для тієї ж функціональності порівняно з попередніми фреймворками.
- Магія часу компіляції: перекладає більшу частину роботи на компіляцію.
- Реактивний за задумом: вбудована реактивність без необхідності використання додаткових бібліотек або шаблонного коду.
- Чудово підходить для SEO: Створює високооптимізований ванільний JavaScript, покращуючи час завантаження та має SEO з “коробки”, що є важливим для бізнесу.
- Ентузіастична спільнота: незважаючи на те, що Svelte нещодавно з'явився, його спільнота дуже активна.
- Зручна інтеграція: Можна легко інтегрувати в існуючі проєкти або використовувати разом з іншими бібліотеками.

Мінусами ж Svelte є:

- Менші екосистеми: менш зріла екосистема з меншою кількістю бібліотек та інструментів у порівнянні з іншими інтерфейсними JavaScript-фреймворками.
- Менше корпоративне прийняття: ще не набув широкого розповсюдження на

підприємствах.

- Навчальні ресурси: менша кількість навчальних ресурсів та підручників у порівнянні з більш відомими фреймворками.
- Відсутність інтеграції з різними IDE;
- Відсутність серйозної підтримки станом на 2024 рік.

Останнім ж на розгляд має бути Next.js, котрий набрав шалену популярність, адже базується на найпопулярнішій бібліотеці для рендерингу контенту, на React. Цей фреймворк JavaScript, дозволяє розробникам створювати зручні та швидкі статичні веб-сайти та статичні веб-додатки за допомогою вище згаданої бібліотеки. Цей фреймворк веб-розробки з відкритим вихідним кодом, що допомагає залучати широку спільноту до його вдосконалення. Він дозволяє розробникам створювати рендеринг на стороні сервера, що зараз є надзвичайно важливим і трендовим, адже замість вже популярного підходу SPA, де клієнт отримував пустий HTML-документ та за допомогою AJAX-запитів він наповнювався контентом, прийшов підхід з серверним рендерингом де розмітка генерується на сервері і вже готова віддається клієнту, що чудово вплинуло на швидкість завантаження, на метрики від Google, а головне на SEO-оптимізацію сайтів.

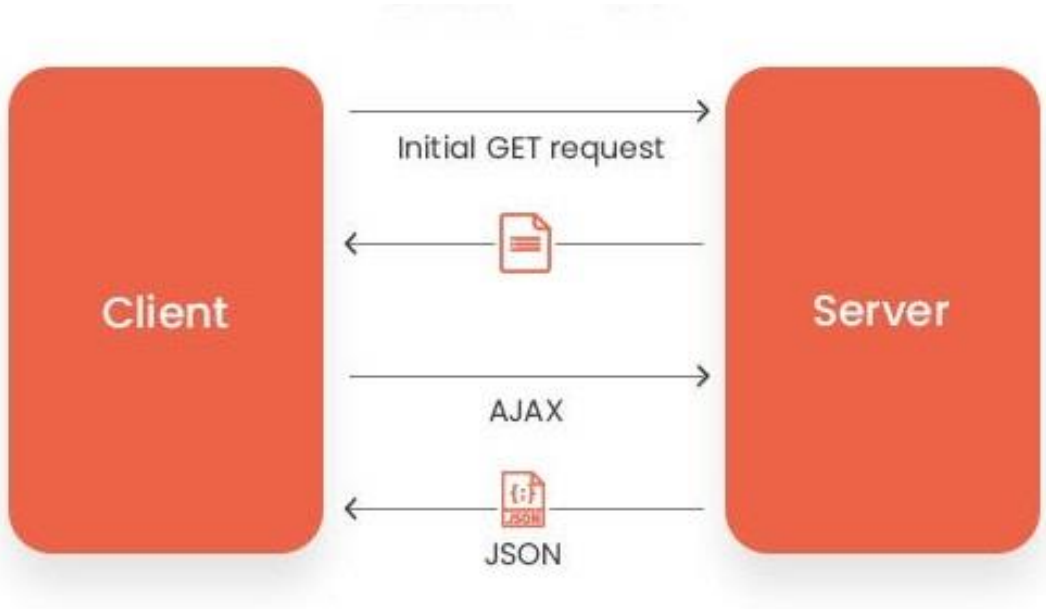


Рисунок 4.1 – Спосіб роботи односторінкових веб-додатків

Також, Next.js базується на React babel та webpack, що надає готове рішення

для серверного рендерингу (SSR) React-компонентів.

Оскільки для розробки найбільш привабливим варіантом є React або фреймворк який базується на React, бо він забезпечує високу продуктивність завдяки своїй віртуальній DOM, яка оптимізує процес оновлення сторінки, дозволяючи здійснювати мінімальні зміни без повного рендирингу; модульній структурі, що дозволяє розробникам створювати компоненти, які можна легко повторно використовувати і тестувати, що сприяє підвищенню ефективності і якості коду; великій та активній спільноті розробників розмаїттю бібліотек і інструментів, що значно спрощує процес розробки і вирішення проблем. Тож вибір пав на Next.js, котрий значно розширює можливості React додатковими особливостями, такими як рендеринг на стороні сервера; статична генерація сайтів, маршрутизація на основі файлів; директорія API, що дозволяє писати backend поруч з frontend; оптимізація картинок; новий бандлер, розроблений на мові програмування Rust; чудова підтримка SEO, тощо. Також важливо зауважити, що на вибір вплинув аналіз статистики з аналітичних ресурсів.

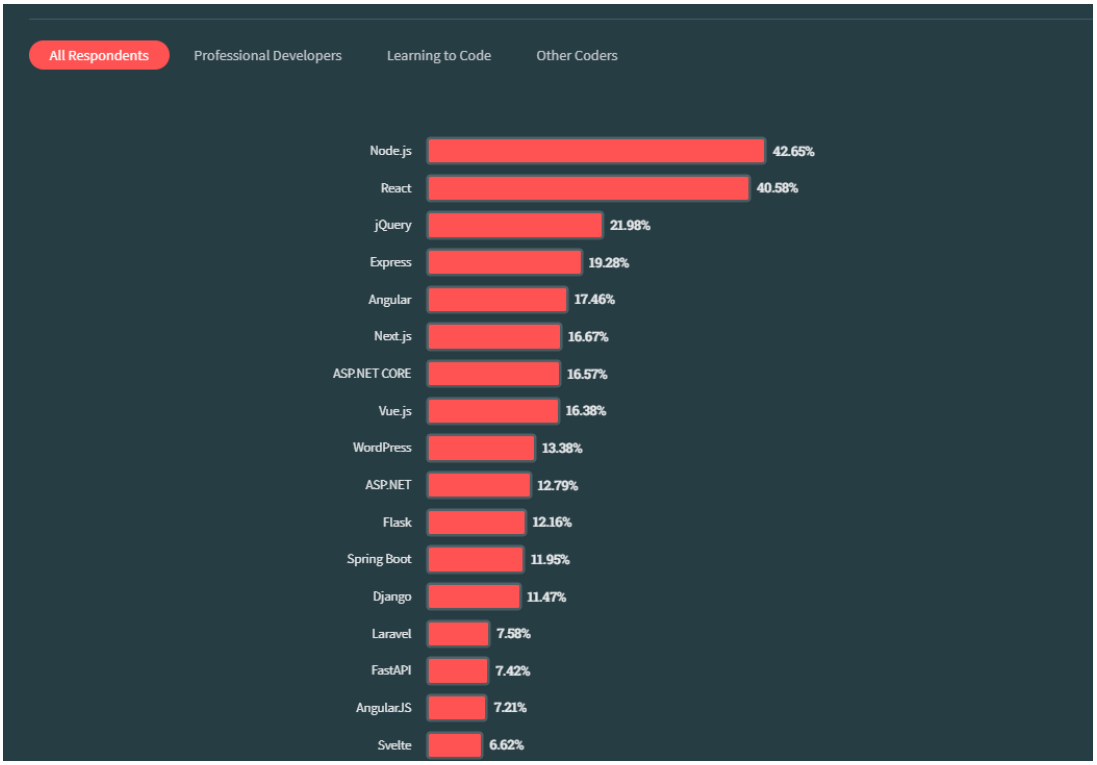


Рисунок 4.2 – Статистика популярності веб-фреймворків (backend та frontend) з ресурсу stackoverflow [19]

4.1.2 Вибір інструменту для написання Backend частини

Не менш важливим у розробці такого продукту є і backend частина, а оскільки для розробки frontend частини було обрано JavaScript фреймворк, то гарним вибором для консистентності буде також TypeScript/JavaScript backend фреймворк.

Почати варто з того, що стандартної версії JavaScript недостатньо для створення ефективного backend'у. Його акцент на запуск у браузері клієнта означає, що JavaScript не може самостійно виконувати важливу серверну роль [8]. Щоб виконувати цю роль, потрібне середовище виконання, яке дозволить JavaScript працювати незалежно від браузера користувача в чисто серверній якості. І саме для цього був розроблений Node.js, котрий використовують як веб-сервер абсолютна більшість, адже він розроблений на тому ж движку (V8), що і найпопулярніший браузер у світі – Google Chrome [16][10].

Хоча Node.js є найвідомішим прикладом таких фреймворків, у нього є конкуренти. Такі альтернативи, як Deno та Next.js, надають можливості, які взагалі не залежать від Node.js. Багато фреймворків, таких як Express.js, доповнюють можливості Node.js. Інші інструменти, такі як Gatsby та Meteor, побудовані на основі Node.js, розширюючи його функціональність та надаючи нові можливості. Для отримання більш детальної інформації їх треба розглянути пристальніше, але перед цим треба ще порівняти переваги та недоліки у використанні JavaScript backend'у.

Переваги:

- Мінімізує розрив між frontend і backend.
- Спрощує процес розробки та сприяє створенню більш послідовних додатків. Легко ділитися та повторно використовувати код.
- Node.js чудово працює для багатьох додатків у порівнянні з такими альтернативами, як PHP або Go.
- Проста у вивченні мова програмування та знайомий більшості розробників синтаксис.
- Підтримується великою спільнотою розробників.

Недоліки:

- Потенційні проблеми з продуктивністю для ресурсоємних додатків через однопотоковість.
- Необхідно уникати вкладених зворотних викликів, які ускладнюють читання та розробку коду.
- Можливі труднощі при роботі з реляційними базами даних.

Як результат, JavaScript з його фреймворками є потужним вибором для створення сервернох частини від веб-сайтів до мобільних додатків. Величезна популярність JavaScript є сильним стимулом для розробників використовувати його як на frontend, так і на backend. Багато розробників знайомі з JavaScript, а можливість швидко розгортати код на стороні сервера та клієнта дозволяє їм створювати потужні додатки. Розробникам більше не потрібно поєднувати різні технології для виконання базових завдань, таких як синхронізація інтерфейсу з backend. Однак JavaScript є однопотоковим, скрипти обробляються лінійно. Отже, JavaScript не може повною мірою використовувати багатопотоковість процесора та інші можливості підвищення продуктивності.

Наступним кроком треба розглянути фреймворки JS/TS для подальшого вибору стеку розробки.

- Deno: Deno, створений спільно з творцем Node.js, – це менеджер виконання та пакування, заснований на JavaScript V8 Engine. Фреймворк написаний на Rust і дозволяє розробникам виконувати JavaScript-код на стороні сервера, а використання мови Rust підвищує його швидкість роботи. Розроблений для покращення деяких оригінальних дизайнерських рішень, які були в Node.js. Deno розглядають як нову ітерацію Node.js – хоча і набагато менш популярну.
- Express.js: Express.js – це фреймворк для створення інтерфейсів передачі представницького стану (REST) за допомогою JavaScript. Однак він також здатний надавати функціональність на стороні сервера. Express.js часто використовується разом з Node.js для створення серверного фреймворку, який дозволяє Node.js працювати як серверна внутрішня мова сценаріїв.

- Next.js: Розроблений для тісної співпраці з React, Next.js продовжує те, на чому зупинився React. Next.js дозволяє React-додаткам використовувати переваги серверного рендерингу. Хоча Next.js може створювати потужний backend, він часто виконує скоріше допоміжну роль, надаючи React-додаткам простий backend. Синтаксис Next дуже схожий на Express, що робить його дуже простим для вивчення. Регулярні оновлення ж додають лише більше можливостей цьому фреймворку.
- Meteor: Meteor розширює можливості Node.js. Він надає доступ до Meteor MongoDB, яка дозволяє швидко синхронізувати дані між додатками Meteor та сховищем MongoDB.

Доречним у контексті розробки frontend частини на Next.js було б роздивитись найбільш популярні та стабільні варіанти, а саме Next та Express, що є чудовими рішеннями в екосистемі Node.js. Хоча вони служать схожим цілям, існують конкретні сценарії, коли один фреймворк може бути більш придатним, ніж інший. Що стосується веб-додатків та API, Next.js виділяється своїми вбудованими можливостями рендерингу на стороні сервера та статичної генерації сайтів. Це робить його чудовим вибором для веб-сайтів з великим вмістом або додатків, які потребують динамічної вибірки даних під час збірки, що чудово вписується у контекст розробки рішень для бізнесу. З іншого боку, Express добре підходить для створення RESTful API завдяки гнучкій системі маршрутизації та підтримці проміжного програмного забезпечення. Якщо ціллю є фреймворк, який відмінно справляється з рендерингом на стороні клієнта, то Next.js – це абсолютно правильний вибір. Він буде легко інтегруватись з майбутнім frontend на Next і забезпечуватиме автоматичне розділення коду, що призведе до кращої продуктивності завдяки завантаженню лише необхідних компонентів при кожному завантаженні сторінки. На відміну від нього, Express в першу чергу фокусується на функціональності на стороні сервера без будь-якої підтримки CSR з “коробки”. Обидва фреймворки мають власний набір функцій, які відповідають різним сценаріям використання. Next.js пропонує SEO-дружню статичну генерацію сайтів та розширені можливості маршрутизації за допомогою файлових конвенцій.

Express може похвалитися гнучкістю завдяки системі проміжного програмного забезпечення, яка дозволяє розробникам легко розширювати функціональність. Розгортання додатку Next.js в периферійній мережі з використанням мережі доставки контенту (CDN) для розподілу статичних ресурсів і контенту додатку між декількома серверами, розташованими в різних географічних точках. Потрібно мати на увазі, що вибір backend-технологій буде залежати від таких факторів, як складність проєкту, потреби в масштабуванні, структура даних і досвід команди розробників.

Отже, проаналізувавши всі ці фактори було прийнято рішення щодо використання Next у якості фреймворку для розробки backend частини системи менеджменту контенту.

4.1.3 Вибір бази даних та відповідного сервісу для її розміщення у хмарі

Дані – це рушійна сила будь-якого бізнесу. Вони допомагають краще розуміти клієнтів, відстежувати запаси та приймати обґрунтовані рішення. Але без належного управління базами даних дані можуть швидко стати дезорганізованими та непридатними для використання. Управління базами даних, та вибір потрібної бази – це безумовно один з найважливіших процесів, адже потрібно зробити так, щоб було легко знаходити, використовувати та аналізувати дані. Ефективне управління базами даних має важливе значення для бізнесу будь-якого розміру, від малого бізнесу для котрого і розроблюється відповідне рішення до великих підприємств, оскільки воно може призвести до покращення роботи, підвищення якості обслуговування клієнтів, покращення процесу прийняття рішень та підвищення безпеки даних. Існує багато різних систем керування базами даних, кожна з яких має свої сильні та слабкі сторони. Щоб визначити конкретні потреби бізнесу, треба взяти до уваги такі фактори, як обсяг і тип даних, які потрібно зберігати, рівень безпеки, котрий потрібен і звісно ж бюджет. Після чіткого розуміння потреб – можна оцінити різні системи управління базами даних на основі їхніх функцій і можливостей, щоб знайти ту, яка найкраще відповідає вимогам.

Перш за все потрібно розглянути переваги ефективного менеджменту бази:

- Впорядковані операції: централізовані дані можуть допомогти бізнесу впорядкувати свої операції, усуваючи необхідність ручного введення даних і дублювання зусиль. Це може призвести до підвищення продуктивності, зниження витрат і підвищення ефективності.
- Покращення обслуговування клієнтів: точні та актуальні дані про клієнтів можуть допомогти бізнесу покращити обслуговування клієнтів, надаючи персоналізовані послуги та проводячи цільові маркетингові кампанії. Це може призвести до підвищення лояльності та задоволеності клієнтів.
- Сприяння прийняттю обґрунтованих рішень: точні та надійні дані можуть допомогти бізнесу приймати обґрунтовані рішення, визначаючи тенденції, закономірності та інсайти. Це може призвести до прийняття стратегічних рішень, підвищення прибутковості, зниження витрат і поліпшення продуктивності.
- Підвищення безпеки даних: системи управління базами даних забезпечують шифрування даних, контроль доступу користувачів та резервне копіювання даних, що допомагає захистити їх від несанкціонованого доступу, крадіжки та втрати. Це гарантує, що компанії зберігають конфіденційність і цілісність своїх даних, захищаючи як свою репутацію, так і свої прибутки.

Наступним кроком для обрання бази стане визначення потреб. Реляційні бази даних є хорошим вибором для малих підприємств, адже через хорошу структурованість та невелику кількість даних, відповідь від бази даних буде моментальною. Також потрібно врахувати і поняття вертикального та горизонтального розширення. Як виявилось зараз дешевше купляти (орендувати) додаткові комп’ютери ніж покращувати оснащеність одного, а тому великі компанії все більше переходять на нереляційні бази даних, бо велика кількість join’ів при використанні реляційних знижує продуктивність ресурсів, але при використанні нереляційних баз з’являється проблема архітектури; розробник має дуже чітко знати як правильно спроєктувати і розподілити документну базу даних між серверами. Реляційні бази даних зберігають дані в таблицях, які складаються з

рядків і стовпців. Це дозволяє легко робити запити та аналізувати дані. Деякі популярні реляційні бази даних включають MySQL, PostgreSQL і Microsoft SQL Server. Бази даних NoSQL зберігають дані не в таблицях, а в різних структурах даних, таких як документи, графіки або пари ключ-значення. Це робить їх добре придатними для зберігання даних, які погано вписуються в традиційну реляційну базу даних. Деякі популярні бази даних NoSQL включають MongoDB, Cassandra та Hadoop.

Оскільки компанія невелика то є сенс розглянути реляційну базу даних, а саме вибір пав на дві найбільш популярні: PostgreSQL та MySQL. У кінці-кінців було обрано MySQL через наступні переваги:

- Популярність і простота використання: оскільки це одна з найпопулярніших систем баз даних у світі, не бракує адміністраторів баз даних, які мають досвід роботи з MySQL. Крім того, існує велика кількість документації в друкованому вигляді та у всесвітній мережі Інтернеті про те, як встановлювати та керувати базою даних MySQL.
- Безпека: MySQL постачається зі скриптом, який допоможе підвищити безпеку бази даних, встановивши рівень захисту паролем, визначивши пароль для користувача root, видаливши анонімні облікові записи та видаливши тестові бази даних, які за замовчуванням доступні для всіх користувачів. Крім того, на відміну від SQLite, MySQL підтримує управління користувачами і дозволяє надавати привілеї доступу для кожного користувача окремо.
- Швидкість: вирішивши не реалізовувати певні функції SQL, розробники MySQL змогли зробити пріоритетом швидкість. Хоча останні бенчмарк-тести показують, що інші СУБД, такі як PostgreSQL, можуть відповідати або принаймні наближатися до MySQL за швидкістю, MySQL все ще має репутацію надзвичайно швидкого рішення для роботи з базами даних.
- Реплікація: MySQL підтримує кілька різних типів реплікації, що є практикою обміну інформацією між двома або більше хостами для підвищення надійності, доступності та відмовостійкості. Це корисно для налаштування

резервного копіювання бази даних або горизонтального масштабування бази даних.

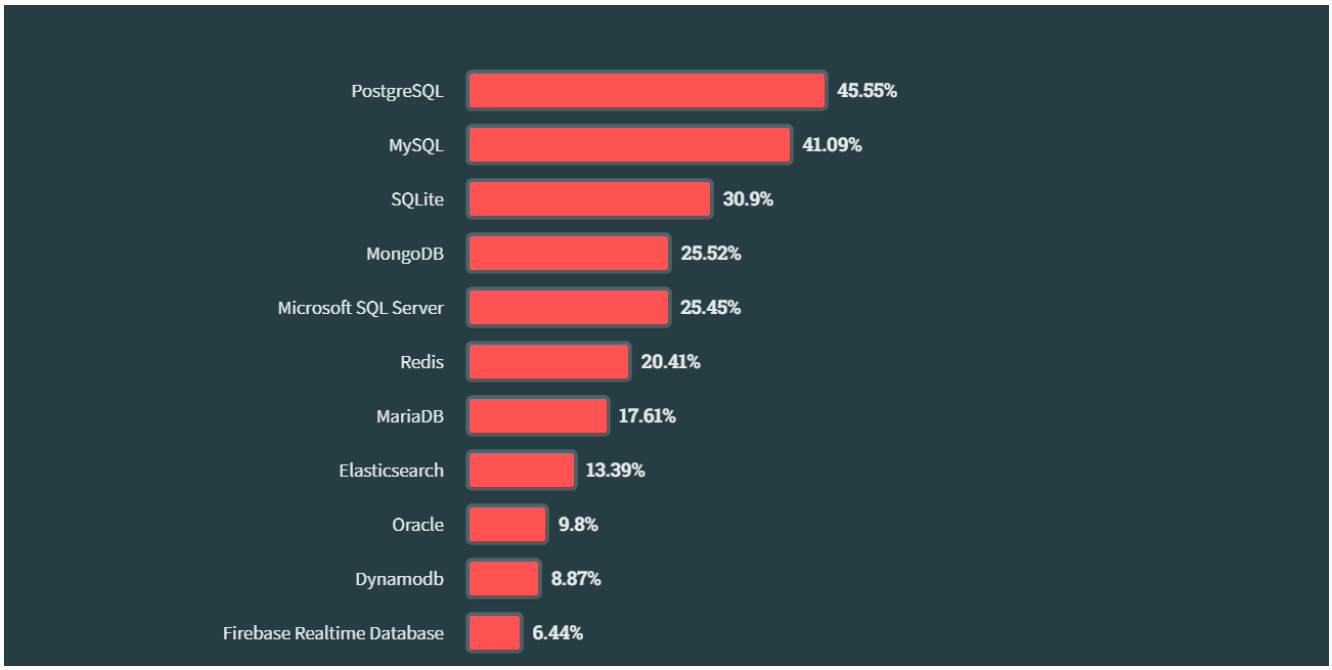


Рисунок 4.3 – Статистика популярності баз даних з ресурсу stackoverflow [19]

4.2 Опис розробленої програми

4.2.1 Ініціалізації проєкта. Підхід до створення стилей

Першим кроком стане ініціалізація проєкта і тут одразу ж постає два питання, а саме, яку мову програмування використовувати TypeScript чи JavaScript. Звісно ж вибором буде TypeScript, котрий останнім часом “увірвався” у динамічну сферу веб-розробки, де інновації та ефективність мають першорядне значення. JavaScript вже давно є основою веб-розробки, але TypeScript з його розширеними функціями та можливостями став переконливою альтернативою, тому треба розкрити нюанси цього вибору.

TypeScript, розроблений Microsoft і є статично типізованою мовою, яка компілюється у звичайний JavaScript [9]. Впроваджуючи статичну типізацію, TypeScript покращує досвід розробки, виявляючи помилки на ранніх стадіях процесу розробки та надаючи покращену підтримку інструментальних засобів. Основними причинами обрання цієї мови для подальшої розробки є:

- Статична типізація: TypeScript дозволяє розробникам визначати типи для змінних, параметрів функцій та значень, що повертаються, забезпечуючи кращу якість коду та зручність його супроводу [9].
- Об'єктно-орієнтоване програмування: TypeScript підтримує традиційні концепції об'єктно-орієнтованого програмування, такі як класи, інтерфейси, успадкування та інкапсуляція, полегшуючи розробку масштабованих і підтримуваних кодових баз [9].
- Покращена підтримка інструментальних засобів: завдяки надійній підтримці інтегрованих середовищ розробки, таких як Visual Studio Code, Webstorm та потужним функціям, таким як навігація по коду, інтелектуальне завершення коду та інструменти рефакторингу, TypeScript спрощує процес розробки[9].
- Підтримка функцій ES6/ES7: TypeScript забезпечує підтримку новітніх функцій ECMAScript, що дозволяє веб-розробникам використовувати сучасний синтаксис і можливості JavaScript, зберігаючи при цьому зворотну сумісність [9].
- Додаткові параметри та значення за замовчуванням: TypeScript дозволяє розробникам визначати необов'язкові параметри і задавати значення за замовчуванням для параметрів функцій, підвищуючи гнучкість і читабельність [9].
- Дженеріки: TypeScript вводить загальні типи, що дозволяє розробникам писати багаторазовий і безпечний код, створюючи функції і структури даних, які працюють з різними типами даних [9].
- Декоратори: TypeScript підтримує декоратори – функцію, натхненну шаблоном проєктування “Декоратор”, яка дозволяє розробникам додавати метадані до класів, методів і властивостей, забезпечуючи такі потужні можливості, як ін'єкція залежностей і аспектно-орієнтоване програмування [9].

Тож логічним вибором став саме TypeScript. Наступним на черзі є використання Tailwind. Але перед тим як прийняти таке важливе рішення для проєкту як використання CSS-фреймворку треба розглянути якими вони бувають, і чому саме

Tailwind набирає нечувану популярність.

Перш за все потрібно розібратись у перевагах CSS-фреймворків. Вони надають:

- Скорочення часу розробки: CSS-фреймворки постачаються з готовими компонентами і стилями, що позбавляє від необхідності писати все з нуля. Це прискорює процес розробки і дозволяє розробникам зосередитися на налаштуванні та доопрацюванні конкретних аспектах проєктів, а не на створенні з нуля.
- Послідовний стиль і дизайн: фреймворки CSS допомагають забезпечити цілісний та узгоджений вигляд різних компонентів і сторінок. Вони гарантують, що всі стилі, елементи інтерфейсу, кнопки та типографіка підтримують єдину мову дизайну, позбавляючи розробників зайвого часу на стилізацію та забезпечуючи кращий користувацький досвід.
- Покращують співпрацю та зручність обслуговування: CSS-фреймворки зазвичай пропонують добре задокументовані бібліотеки та встановлені конвенції, що полегшує розробникам співпрацю та підтримку кодових баз. Завдяки спільній кодовій базі та великій документації розробники можуть легко розуміти та працювати з кодом один одного.

Другим питанням на розгляд будуть типи CSS-фреймворків. Зараз існує 3 типи:

- Компонентно-орієнтовані фреймворки. Компонентні фреймворки пропонують набір готових компонентів інтерфейсу, які розробники можуть підключати до своїх додатків для швидкого створення інтерфейсів. Мета полягає в тому, щоб забезпечити модульну і багаторазову систему дизайну, яка допоможе створювати послідовні і візуально привабливі веб-додатки без необхідності щоразу починати з нуля.
- Фреймворки, орієнтовані на утиліти. Ідея фреймворків, орієнтованих на утиліти, полягає в тому, що CSS не повинен бути описовим і не повинен сильно покладатися на розмітку. Наприклад для розробки клас footer, який позначає футер веб-сайту має базуватися на функціональності. Щоб не обмежувати дизайн додатку лише тим, що надається фреймворком,

утилітарні фреймворки пропонують стилі та класи CSS, які виконують лише одну функцію, або невеликий набір функцій, як будівельні блоки для розширення та налаштування дизайну додатку поза межами компонентно-орієнтованого фреймворку.

– CSS-in-JS. З появою бібліотек JavaScript, таких як React, були створені фреймворки CSS-in-JS, які дозволяють розробникам маніпулювати стилями безпосередньо в JavaScript шляхом включення CSS в розмітку JavaScript. CSS-in-JS використовує динамічну природу JavaScript, щоб забезпечити спосіб написання інтерактивних стилів CSS, які є продуктивними і базуються на даних користувача та взаємодії.

Існує більше типів фреймворків CSS, але ці три категорії охоплюють найбільш помітні групи. Не існує тонкої межі, яка б розділяла ці категорії. Більшість CSS-фреймворків можуть перетинатися в декількох категоріях – наприклад, фреймворк на основі компонентів може надавати утиліти, а фреймворк на основі утиліт може надавати компоненти.

Отже, вибір пав на Tailwind, котрий орієнтований на утиліти і дозволяє швидко створювати розмітку для проєкту, адже стилі прописуються прямо у className замість стандартного підходу з імпортом файлу стилів та створенням класів. Це чудово підходить для самостійної та швидкої розробки. Для розширення функціональності Tailwind було використано ShadcnUI, котрий використовує компонетний підхід використовуючи вже знайомий Tailwind. Така комбінація сильно підвищує продуктивність і створює загальний консистентний стиль для веб-додатку. Tailwind посідає друге місце серед найбільш використовуваних фреймворків в опитуванні State of CSS 2023 з показником близько 76% [17].

```
return (
  <div className="flex-col">
    <div className="flex-1 space-y-4 p-8 pt-6">
      <SizesClient data={formattedSizes} />
    </div>
  </div>
);
```

Рисунок 4.4 – Приклад використання стилів Tailwind

Рисунок 4.5 – Приклад додавання безголового компоненту за допомогою пакетного менеджеру

4.2.2 Опис ініціалізації проєкта. Обґрунтування вибору бандлера, менеджеру пакетів

При ініціалізації проєкта Next на вибір є 2 бандлери: Webpack та TruboPack. Але перед тим як створити проєкт треба дослідити який з них краще, навіщо вони потрібні та переваги використання бандлеру.

Бандлер – це інструмент, що використовується у веб-розробці для перетворення вихідних файлів програми на статичні ресурси, що використовуються браузерами. Основне призначення бандлерів полягає в організації модулів і залежностей у інтерфейсних проєктах, оптимізації продуктивності веб-сайтів і зменшенні розміру файлів. Зазвичай це відбувається шляхом мінімізації кількості HTTP-запитів, а в деяких випадках шляхом збільшення кількості запитів (більшої кількості чанків), щоб розвантажити основний бандл. Основними функціями бандлерів є:

- Транспіляція: бандлер перетворює вихідний код, наприклад, з TypeScript на JS або з SCSS на CSS.
- Конкатенація модулів: аналізує модулі та їх залежності, а потім об'єднує їх в один або декілька вихідних файлів, дозволяючи розробникам писати модульний код більш організовано та ефективно.
- Мініфікація: під час цього процесу бандлер часто застосовує мінімізацію, яка передбачає видалення непотрібних пробілів, коментарів та інших елементів,

які не впливають на функціональність коду. Це зменшує розмір результуючих файлів, що призводить до швидшого завантаження програми та зменшення споживання даних.

- Управління ресурсами: бандлери також допомагають керувати іншими ресурсами, такими як таблиці стилів, зображення і шрифти, об'єднуючи їх в один або кілька файлів, оптимізуючи їх завантаження при рендерингу сторінки.
- Управління залежностями: бандлери відстежують залежності між модулями, дозволяючи автоматично включати лише ті частини коду, які дійсно використовуються в проєкті.

Вибір пав на Webpack одразу лише з однієї причини – це стабільність. TurboPack, хоч і написаний мовою програмування Rust та позиціонує себе як більш швидке та продуктивне рішення, але все ж знаходиться досі в альфа моді. Також Webpack через велику популярність має велику кількість ресурсів для ознайомлення з тим, як його оптимізувати. Тобто коротко описуючи переваги Webpack це:

- Широкі можливості конфігурації: Webpack пропонує численні варіанти конфігурації, що дозволяє налаштувати процес комплектації та пакування ресурсів відповідно до потреб конкретного проєкту. Це особливо корисно у великих і складних проєктах.
- Управління різноманітними ресурсами: інструмент підтримує різні типи ресурсів, такі як JavaScript, CSS, зображення тощо.
- Підтримка спільноти: Більш ніж десятирічна історія розробки призвела до створення великої спільноти користувачів і великої документації, що полегшує використання пакувальника.

Не менш важливим є вибір і вибір менеджеру пакету. Вибір тут доволі не великий, але у цьому випадку обирати потрібно npm, адже бібліотека компонентів shadcnUI, котру було обрано для розробки інтерфейсу ще не в повному обсязі підтримує yarn, більш захищене рішення на відміну від npm, а використання двох менеджерів пакетів є поганою практикою у світі розробки, адже вони створюють різні файли для встановлення залежностей, що може призвести до проблем з

бандлінгом і хаосу з версіями пакетів.

4.2.3 Опис логічної структури. Використані сервіси

Сервіси є неймовірно важливим інструментом, адже вони захищають дані користувачів, а також знижують навантаження на базу даних і звісно ж створюють комфортні умови розробки для розробників. Першим, що потрібно для магазину звісно ж сервіс для зберігання картинок, адже люди мають бачити той товар, що купують.

Доступ до фотографій з будь-якого місця, синхронізація фотографій між пристроями, автоматичне резервне копіювання і використання хмарних інструментів для редагування та керування фотографіями. Це все дуже заощаджує місце на локальних пристроях зберігання, підвищує продуктивність і зменшує ризик втрати даних через збій обладнання або крадіжку. Для цього було обрано сервіс Cloudinary, котрий надає хоч і обмежений, але безкоштовний доступ.

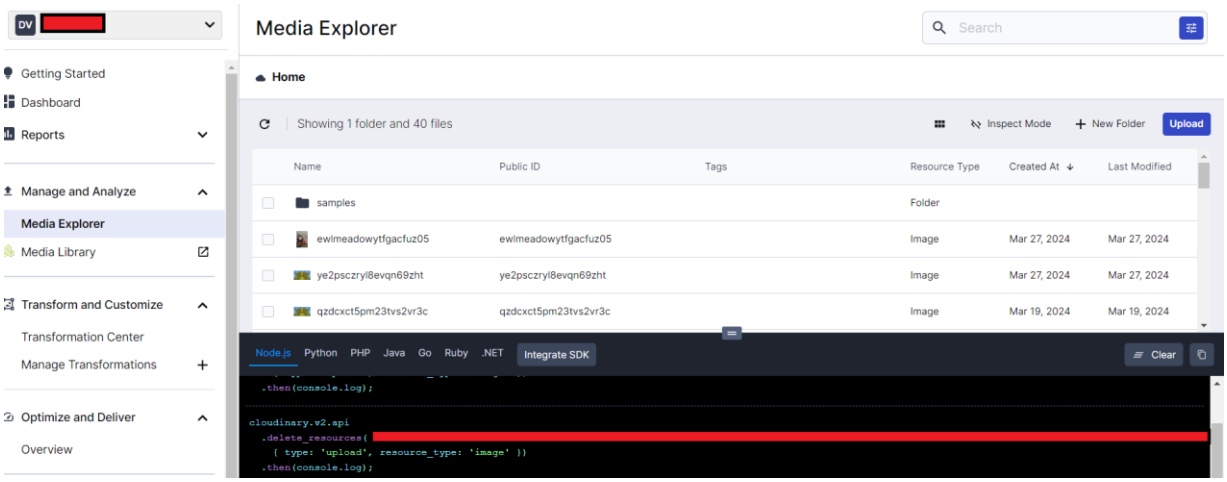


Рисунок 4.6 – Адмін-панель сервісу Cloudinary

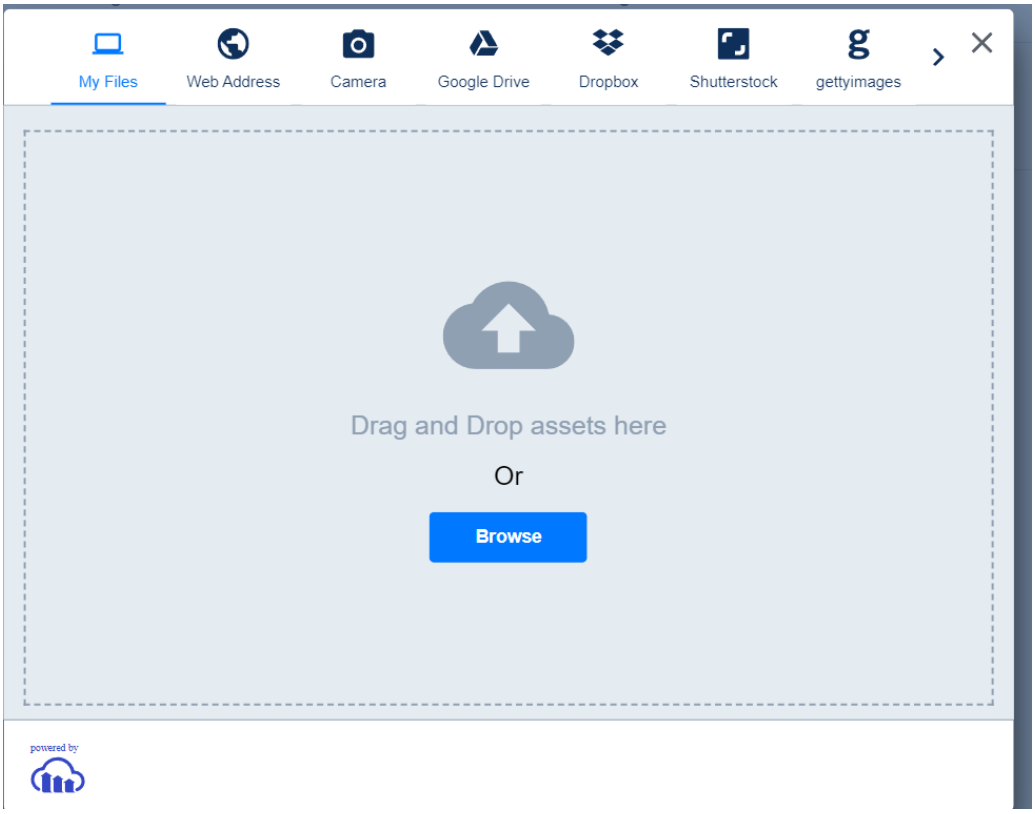


Рисунок 4.7 – Використання компоненту завантаження зображення Cloudinary

Наступним і не менш важливим є сервіс для контролю користувачів і для розробки логіну, в тому числі і через Google. Для цієї задачі було обрано Clerk, адже це інструмент, котрий рекомендують розробники Next, тобто компанія Vercel. Так, є чудові Cognito від Amazon, але також потрібно враховувати цінову політику компанії, тому оптимальним вибором став Clerk.

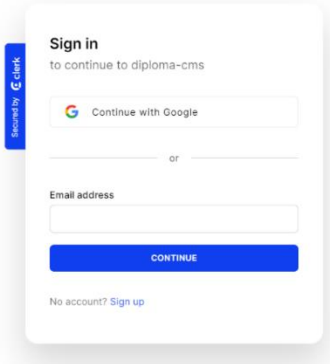


Рисунок 4.8 – Використання компоненту логіну від Clerk

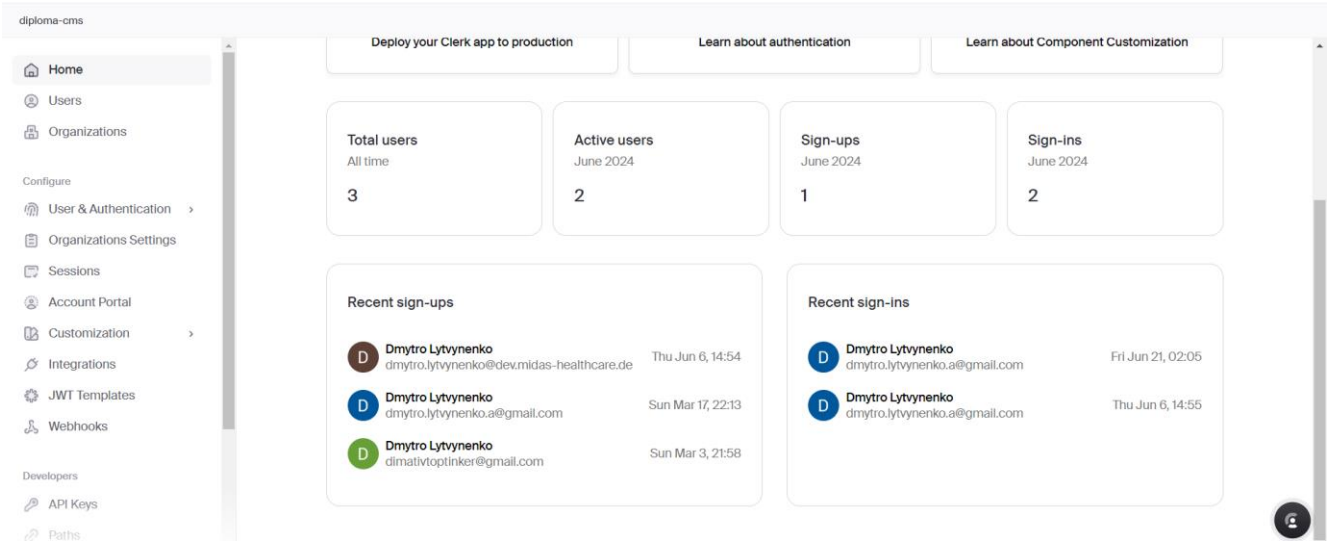


Рисунок 4.9 – Адмін-панель сервісу Clerk

І звісно ж потрібно попіклуватись і про базу даних. Зважаючи на ресурси компанії виділені на сервіси, було обрано оптимальний варіант, а саме Aiven, адже альтернативи у вигляді продуктів від Amazon та PlanetScale було визнано надто коштовною.

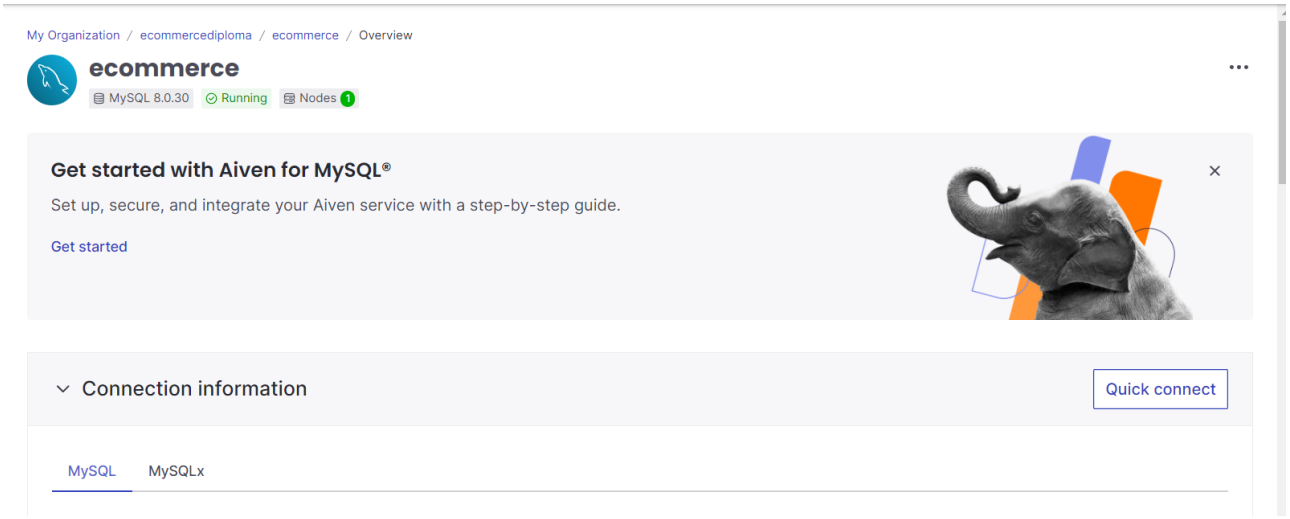


Рисунок 4.10 – Адмін-панель сервісу Aiven

Одним з останніх є сервіс оплати, котрий надасть зручні інструменти для прийняття платежів, грамотного менеджменту та звісно ж захистить користувачів від зловмисників. Для цього було обрано один з найпопулярніших сервісів оплати, а саме Stripe. Перевагами Stripe є:

– Створити акаунт Stripe можна за лічені секунди на відміну від інших

платіжних систем. Потрібно лише вказати своє повне ім'я, адресу електронної пошти та пароль, щоб почати роботу.

– Не потрібен торговий рахунок зі Stripe Більшість платіжних процесорів вимагають відкрити торговий рахунок у банку, щоб приймати онлайн-платежі. Stripe, використовує власний торговий рахунок від імені користувача. Це економить багато часу, клопоту та звільняє від паперової роботи та періоду очікування.
– Регулярні платежі за допомогою Stripe. Якщо бізнес пропонує продукти або послуги на основі підписки то іноді буває важко виставляти рахунки покупцям щомісяця, але цю проблему цей платіжний провайдер також вирішив.
– Stripe інтегрується з різним стеком і продуктами. З розвитком бізнесу може знадобитися інтеграція платіжного процесора з іншими платформами та інструментами. Наприклад з’явиться інструмент для управління фінансами, або якийсь інший бухгалтерський інструмент і його інтеграція пройде без неприємностей.

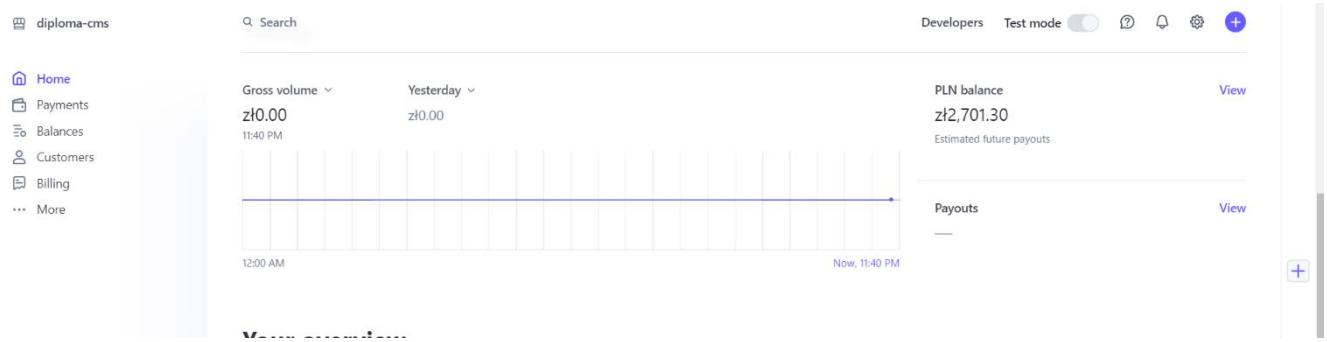


Рисунок 4.11 – Адмін-панель сервісу Stripe з відомостями про продажі

Останнім став сервіс для розміщення веб-сайту у мережі Інтернет. Для цього було використано Vercel, адже це продукт компанії Vercel, котрий чудово працює з Next. Додатковими перевагами є:

– Швидке розгортання. Vercel вирізняється швидким розгортанням сайтів, що підтримується низкою готових шаблонів, які сприяють швидкому налаштуванню без складнощів з розгортанням серверів. Ця можливість ще

більше розширюється завдяки підтримці універсального формату, який дозволяє легко розгортати різні фреймворки JavaScript. В результаті можна розгорнути застосунок на Astro так само швидко, як і на Next.js.

– Інтеграція та співпраця. Vercel легко інтегрується з широким спектром інструментів, баз даних, журналів та систем сповіщень (наприклад – Slack), що покращує співпрацю в команді. Інтеграція з популярними системами контролю версій, такими як GitHub, GitLab та Bitbucket, спрощує управління проєктами та їх налаштування.
– Глобальна CDN і HTTPS. Vercel надає глобальну мережу доставки контенту та автоматичний протокол HTTPS, що забезпечує швидку та безпечну доставку веб-контенту.
– Безкоштовний план та економічна ефективність. Vercel пропонує щедрий безкоштовний тарифний план Hobby, що робить його привабливим варіантом для невеликих проєктів та особистого використання.

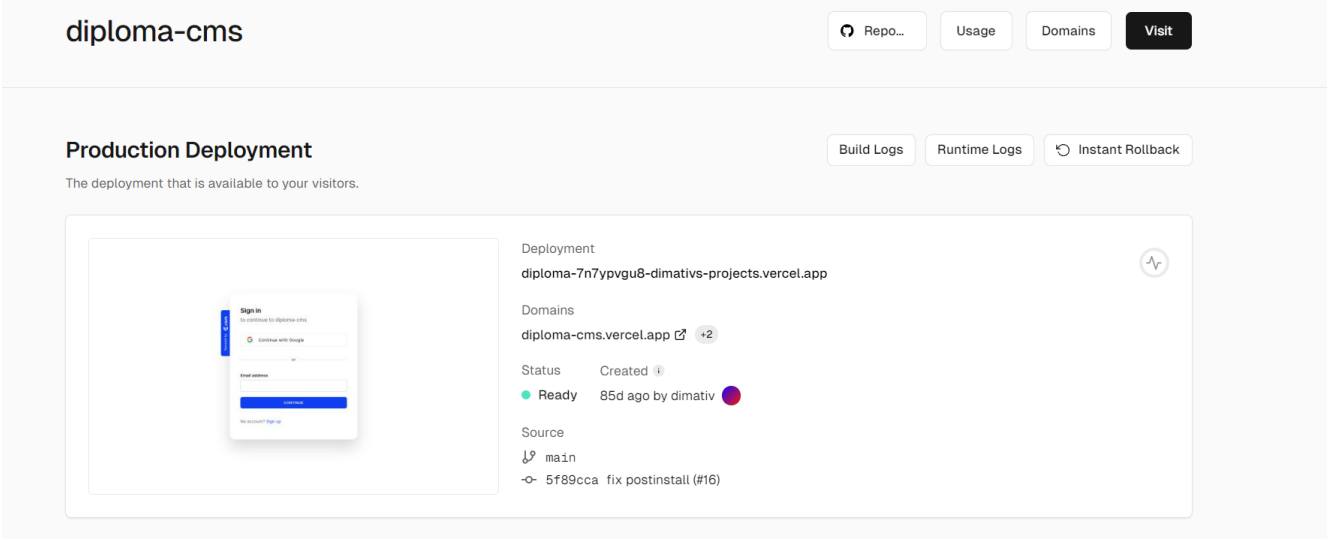


Рисунок 4.12 – Панель сервісу Vercel з задеплоєним проєктом системи менеджменту контенту

4.2.4 Опис логічної структури. Архітектура проєкту

Звісно ж важливим є описати як загальну архітектуру так і архітектуру

проєкта на Next. Отже почати потрібно саме з проєкту на Next.

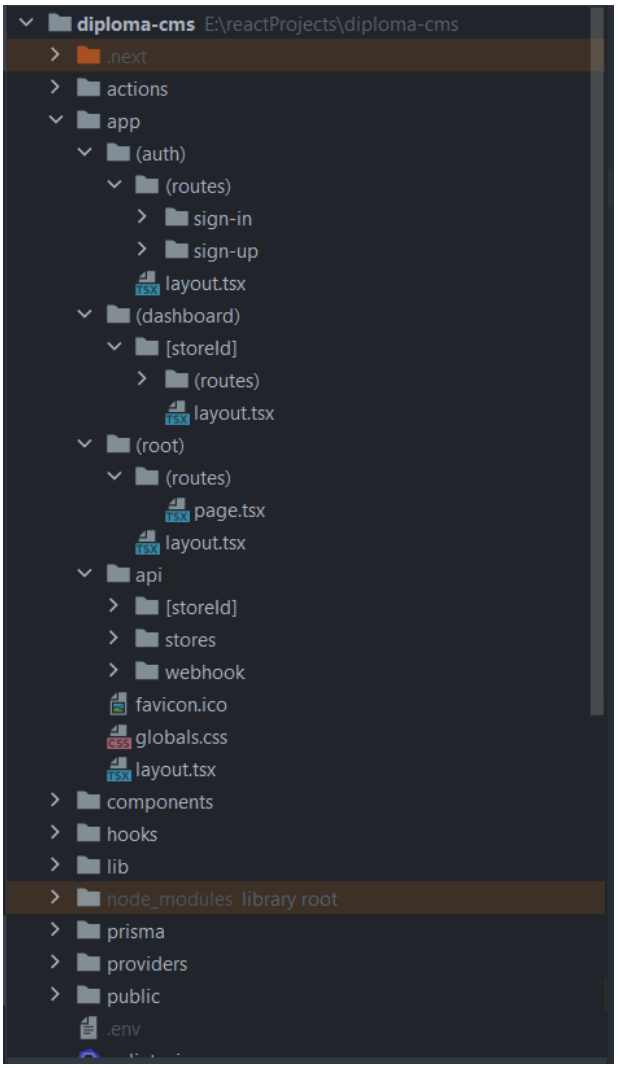


Рисунок 4.13– Побудована архітектура проєкту на Next

Тут можна побачити перша за все директорію actions, котра відповідає за отримання результатів запитів на серверних компонентах, адже action'и не використовують React hooks. Далі йде папка app, котра містить в собі всі роути програми, котрі поділені на організаційні папки, а саме: auth, root, dashboard. Наступною є папка api, котра містить в собі backend розробленого проєкту з його роутами. Наступною організаційною директорією є components, де зберігаються усі компоненти, котрі можуть бути перевикористані в проєкті, що як раз є слідуванням одному з принципів SOLID, а саме принципу під літерою S – Single Responsibility principle, тобто принцип єдиної відповідальності [6]. Варто зауважити, що ця папка поділяється ще на декілька під-директорій, а саме: ui, modals. Де modals зберігає у

собі усі розроблені модальні вікна, а ui – імпортовані компоненти від бібліотеки shadcnUI.

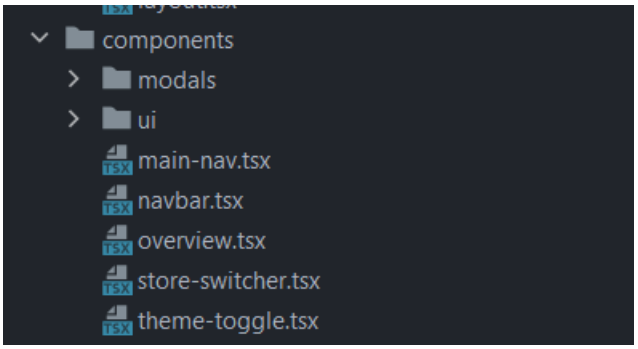


Рисунок 4.14 – Вміст папки components

Далі папка hooks, що є аналогом директорії actions, але вона містить в собі React hooks, через це функції, так звані хуки, потрібно використовувати лише в клієнтських компонентах.

Останніми, але не менш важливими є наступні папки: lib, містить в собі корисні функції котрі дуже часто перевикористовуються; prisma – це ORM для зручного керування базою даних, там містяться відповідні конфігураційні файли; providers – якісь загальні функції, котрі мають стан, що має розповсюджуватись на увесь проєкт.

Останньою дією є підбивання підсумків та створення відповідної архітектурної документації [7]. Для цього було розроблено відповідну діаграму (див. рис. 2.28).

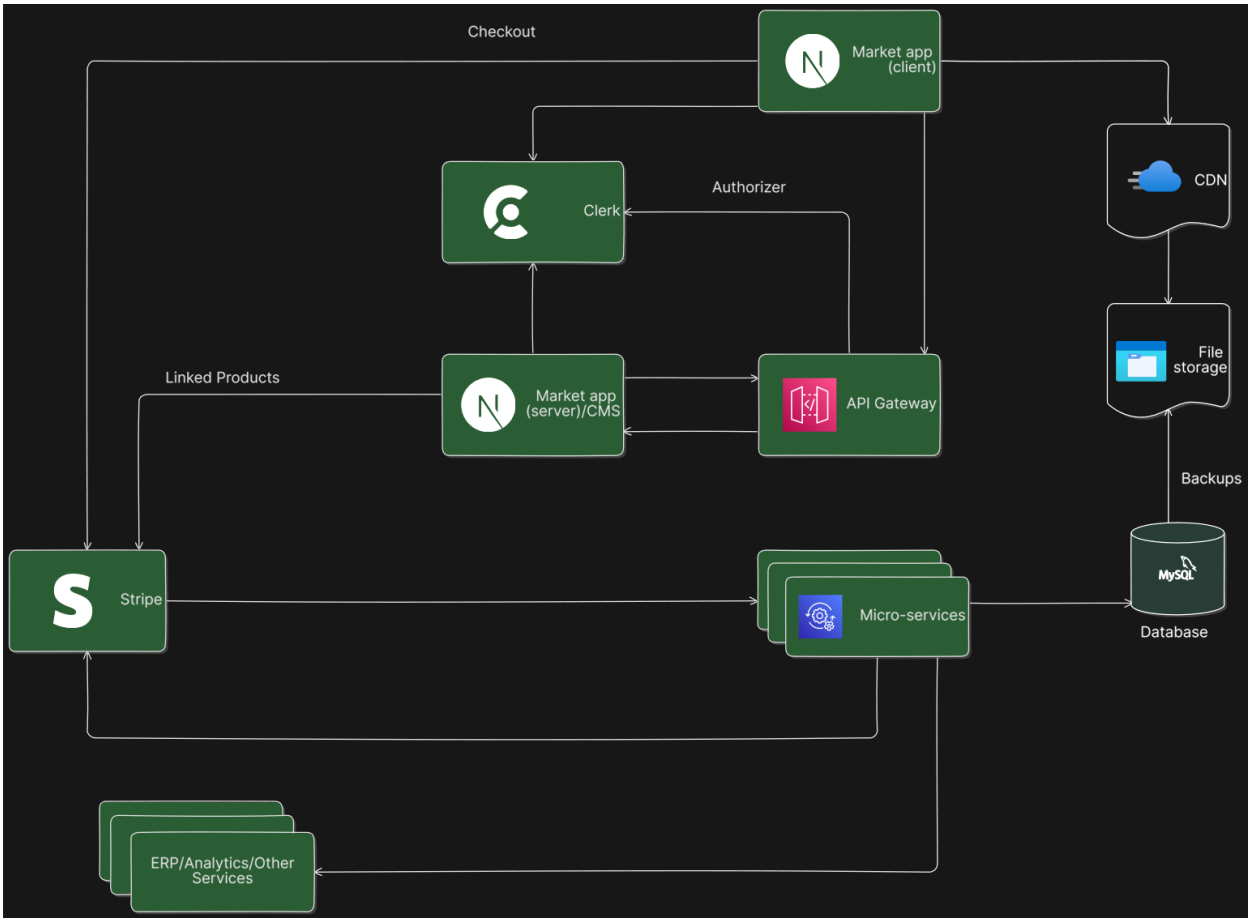


Рисунок 4.15 – Загальна архітектура проєкта

4.2.5 Опис роботи створеного продукту

Перш за все, щоб скористатись розробленим додатком потрібно створити для цього новий магазин, після чого користувач отримає унікальний API ключ для побудови свого магазину.

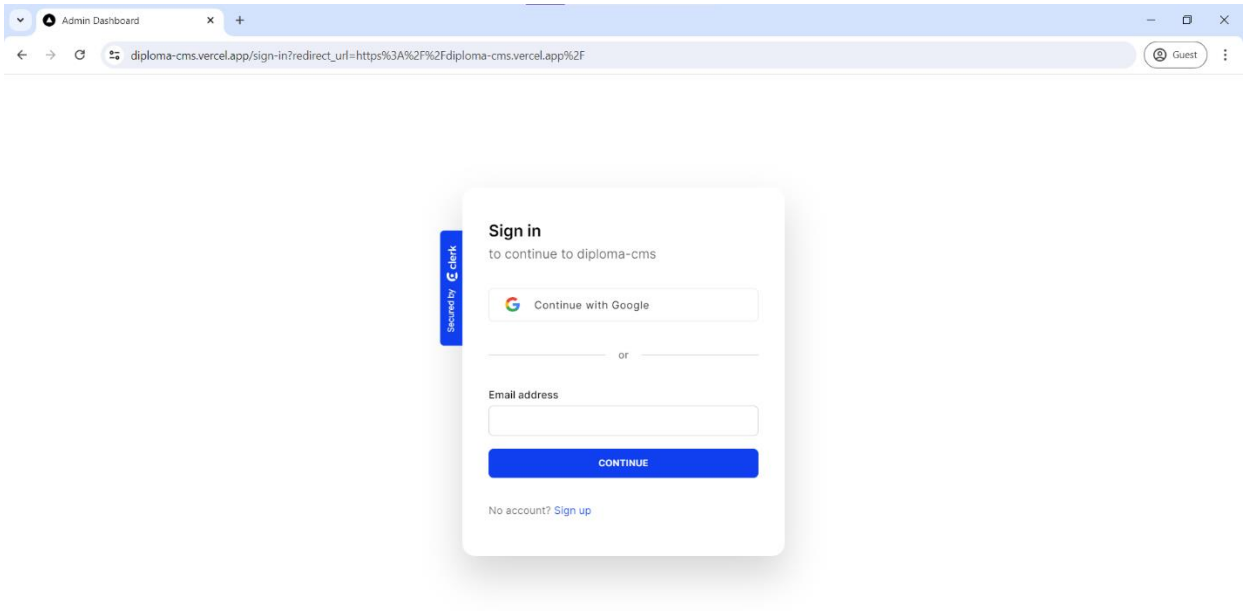


Рисунок 4.16 – Реєстрація при першому відвідуванні сайту CMS

Далі користувач бачить модальне вікно, котре неможливо закрити, доки не буде надана назва новоствореному магазину. Також у майбутньому індивід може створити ще один магазин з бажаною назвою, що є чудовим рішенням менеджменту для середніх підприємств, котрі мають дочірні компанії.

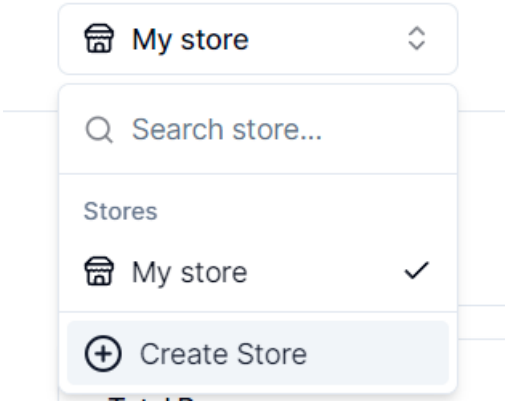


Рисунок 4.17 – Створення нового магазину

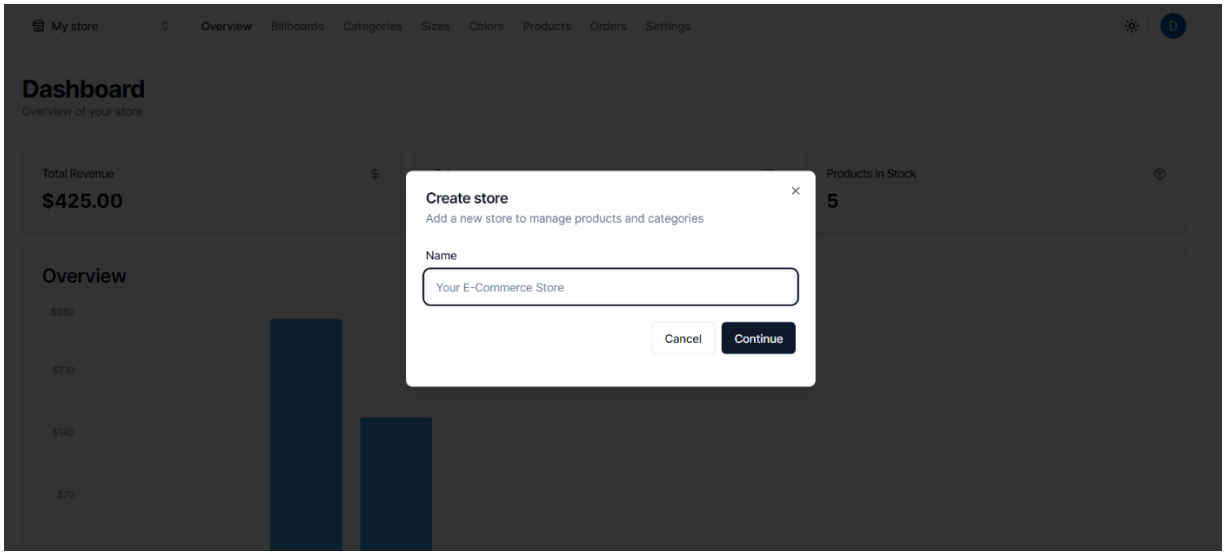


Рисунок 4.18 – Модальне вікно надання ім’я магазину

Після успішної реєстрації магазину, персону відповідальну за наповнення контентом “зустрічає” сторінка зі статистикою основних показників, а саме: скільки товарів залишилось, на скільки зросли продажі, зароблені кошти протягом усього часу, та статистика по місяцях.

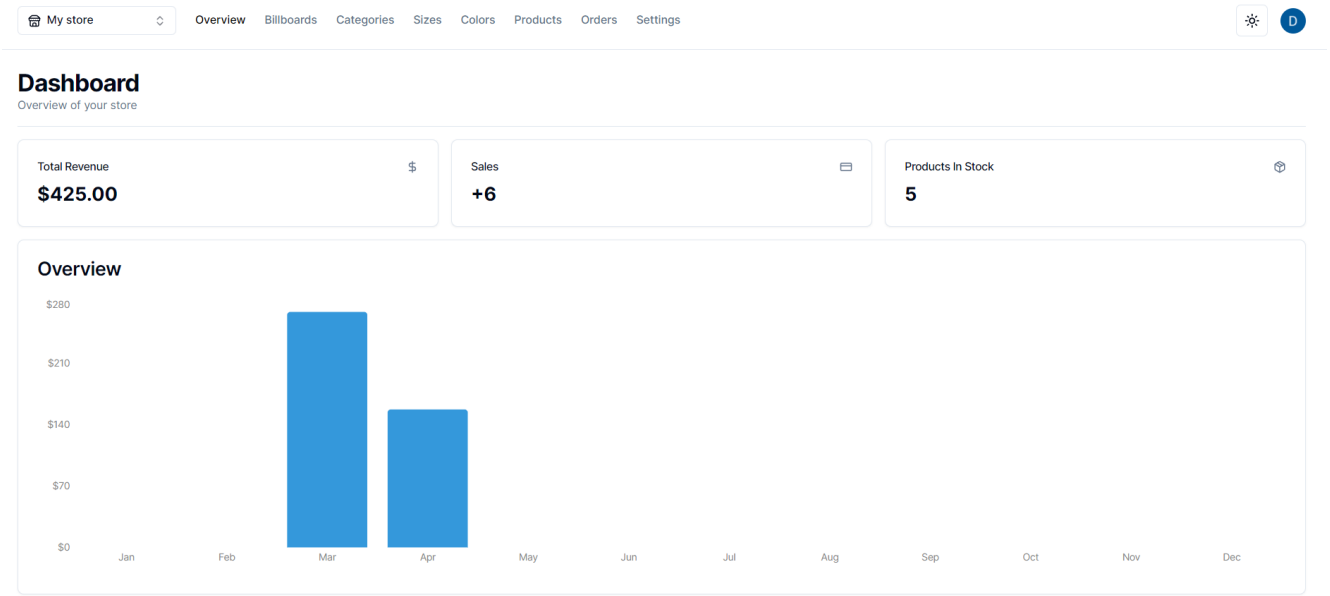


Рисунок 4.19 – Початкова сторінка CMS

Також можна побачити у правому верхньому куті перемикач теми, за допомогою котрого, кожен може вибрати більш привабливу тему, що є чудовим підвищенням комфорту користування системою адміністрації контенту. Всього

зараз доступні 3 теми: світла, темна, та тема девайсу з котрим людина користується.

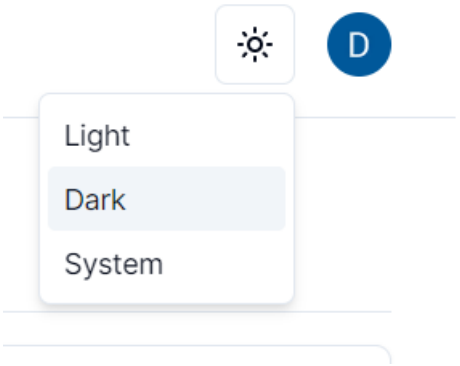


Рисунок 4.20 – Вибір теми

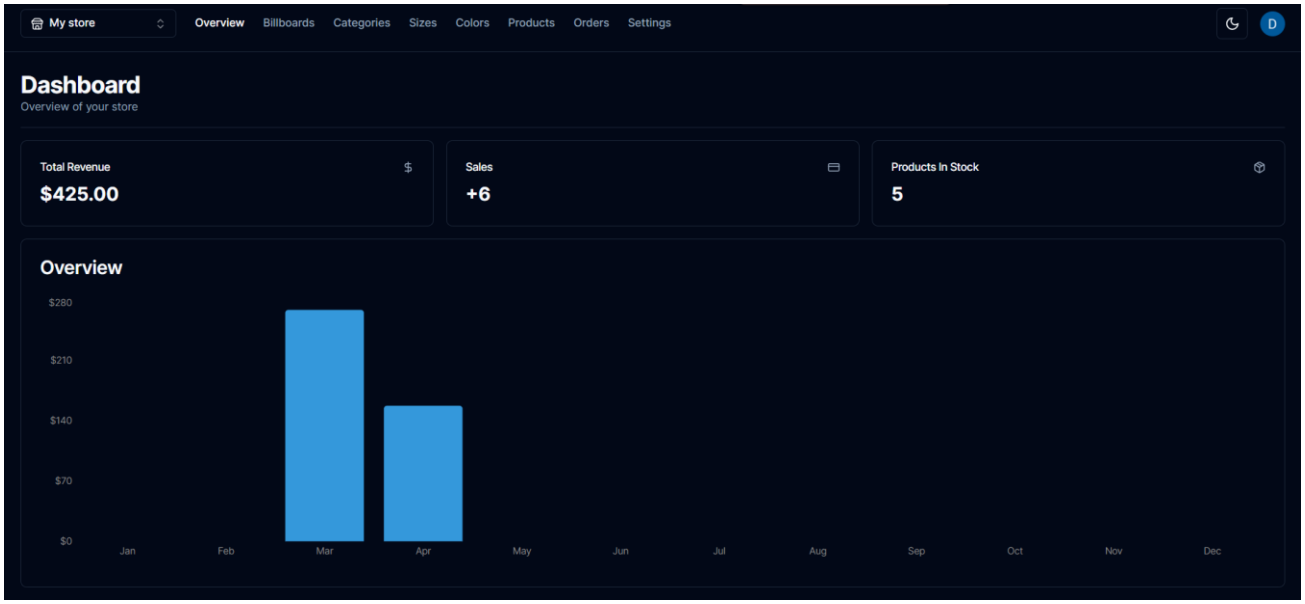


Рисунок 4.21 – Приклад застосованої темної теми

Звісно ж можна і розлогінитися клікнувши у правий верхній кут на іконку.

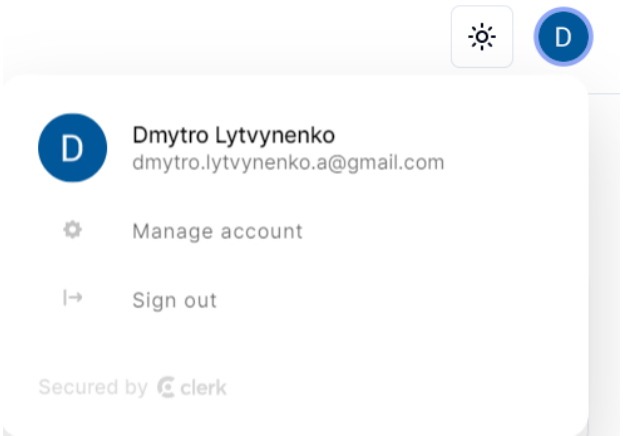


Рисунок 4.22 – Вікно користувача

Перейшовши у вкладку Settings, користувач може отримати унікальний API ключ для створення свого власного сайту за допомогою безголової CMS, змінити назву магазину, або навіть видалити його.

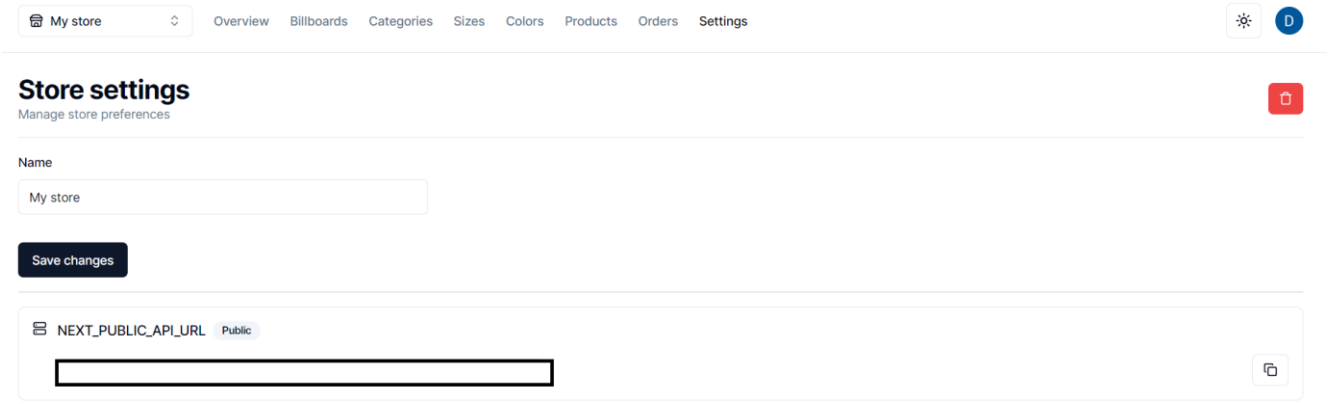
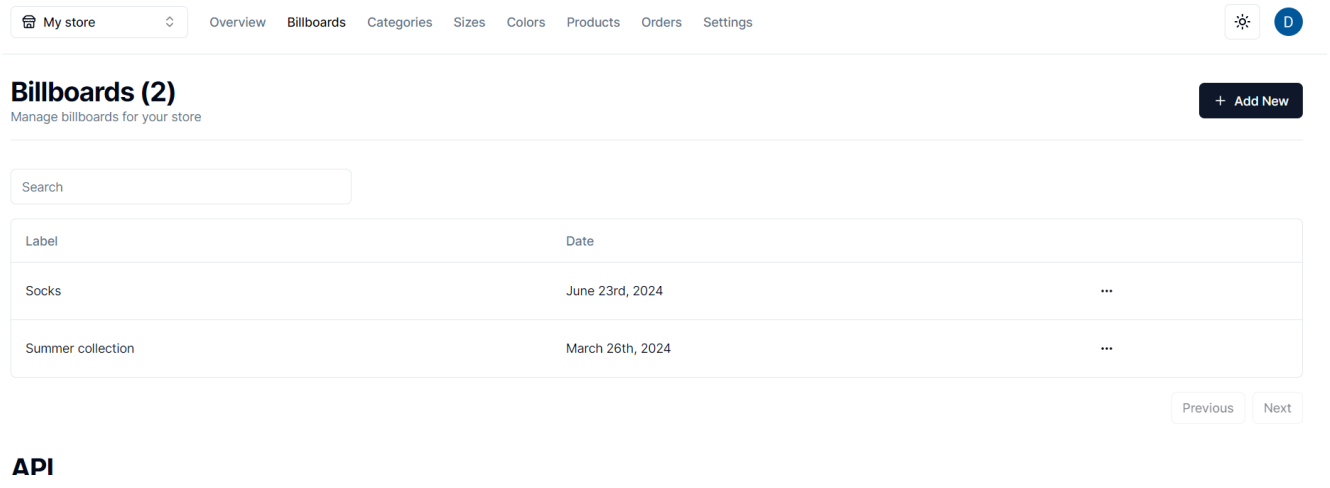


Рисунок 4.23 – Сторінка для видалення магазину та отримання ключа

Наступним і не менш важливим для кожного сайту електронної комерції є білборди, перейшовши на вкладку Billboards та натиснувши кнопку “Add New” можна створити білборд, завантажити зображення, надати назву. Також на сторінці доступний пошук білбордів. Натиснувши на іконку “…” можна отримати швидку можливість до редагування, копіювання, або видалення білбордів.



API

Рисунок 4.24 – Сторінка для створення, видалення та редагування білбордів

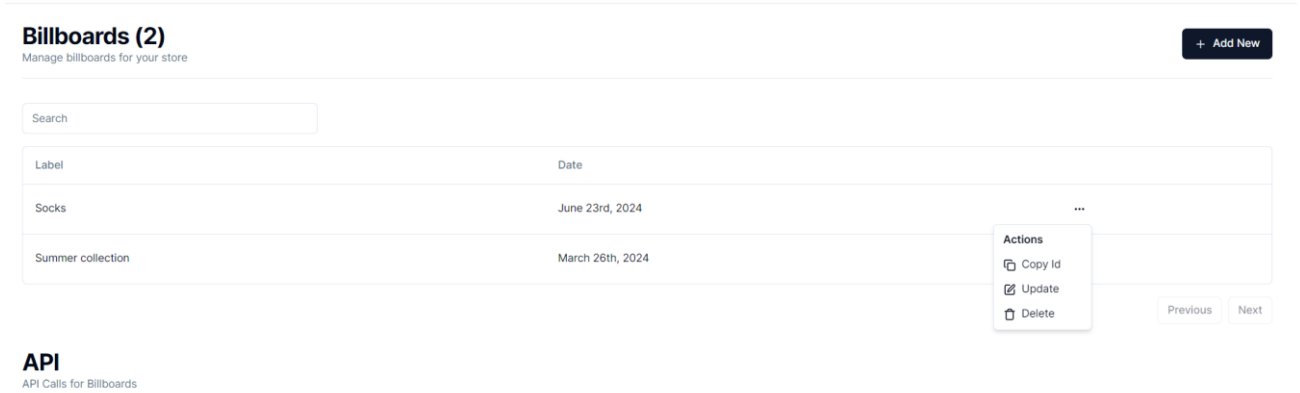


Рисунок 4.25 – Приклад роботи швидких дій для білбордів

Якщо ж опуститись по сторінці нижче, можна отримати відповідні endpoint’и для виконання маніпуляцій з білбордами. Аналогічні речі розроблені для кожної складової продукта.

Рисунок 4.26 – Приклад endpoint’ів для білбордів

Наступні сторінки відповідають за відповідну характеристику товару, котру можна створити, найбільш цікавим з цих сторінок є створення кольорів, адже користувач має змогу одразу побачити колір поруч з кодом цього кольору, що не створить ситуацій, коли колір буде підписаний наприклад як червоний, а насправді він буде жовтий.

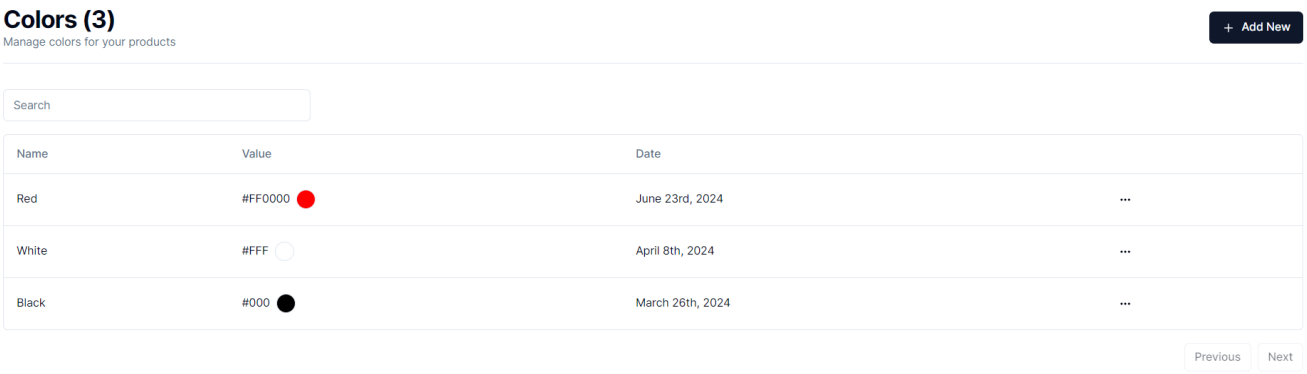


Рисунок 4.27 – Сторінка кольорів

Наступною є сторінка з менеджментом продуктів, де можна створити продукт, додати йому картинки, ім’я, ціну, обрати категорію вже з існуючих, надати розмір та колір. Останні 2 прапорці це чи активний продукт чи він архівований. Важливо зазначити, що чекбокс архівації продукта має вищий пріоритет перед чекбоксом наявності продукту, це зроблено задля непередбачуваних ситуацій. По друге, важливим є те, що неможливо видалити категорію продуктів, поки не будуть видалені усі продукти з цієї категорії; те саме з категоріями і білбордами, неможливо видалити білборд без видалення усіх категорій, які йому притаманні.

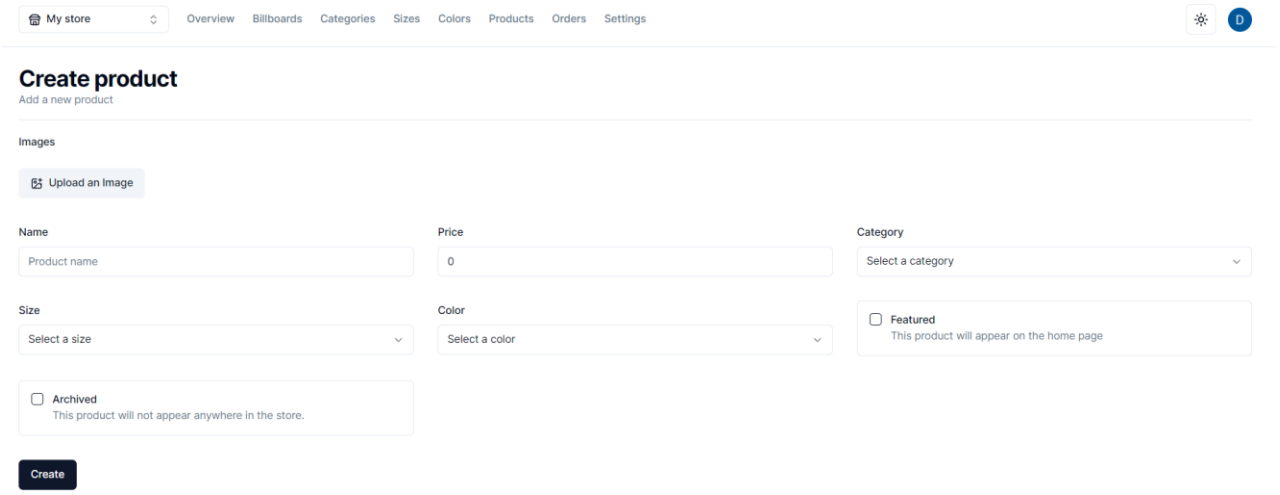


Рисунок 4.28 – Приклад сторінки для створення продукта

Останнім шляхом маршрутизації є відомість про замовлення. Тут зручно відображається куплений продукт, телефон замовника, його адреса з поштовим

кодом та сума, котру має покупець оплатити. Також видно і статус оплати.

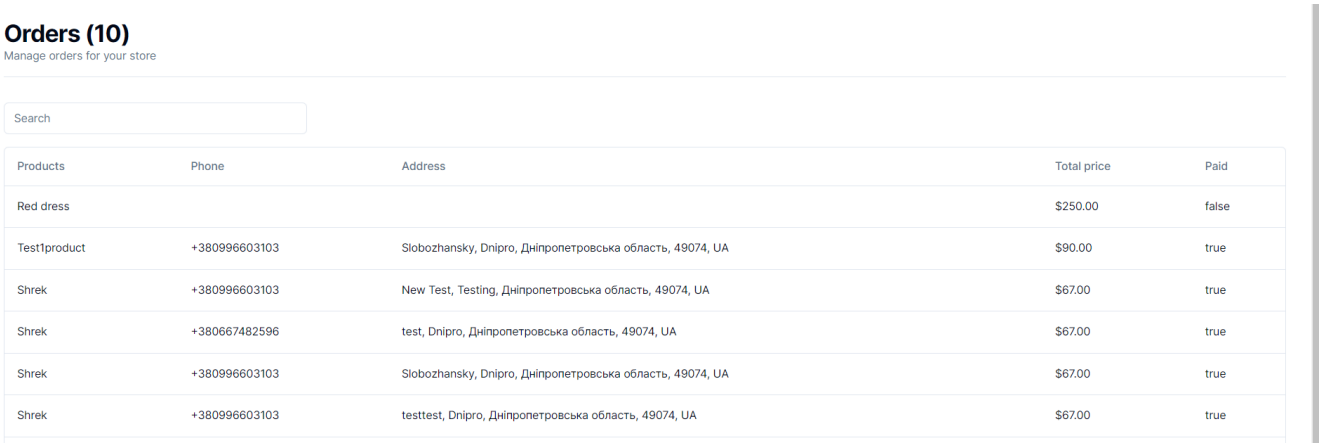


Рисунок 4.29 – Сторінка замовлень

Після огляду створеного продукту, можна перейти до перегляду веб-сайту створеного за допомогою цієї CMS. Одразу ж покупців зустрічає банер з літньою колекцією магазину.

Рисунок 4.30 – Банер та категорії розробленого сайту

Після цього можна побачити усі доступні продукти з цінами та кнопками швидкого доступу, котрі з’являються при наведені курсора миші.

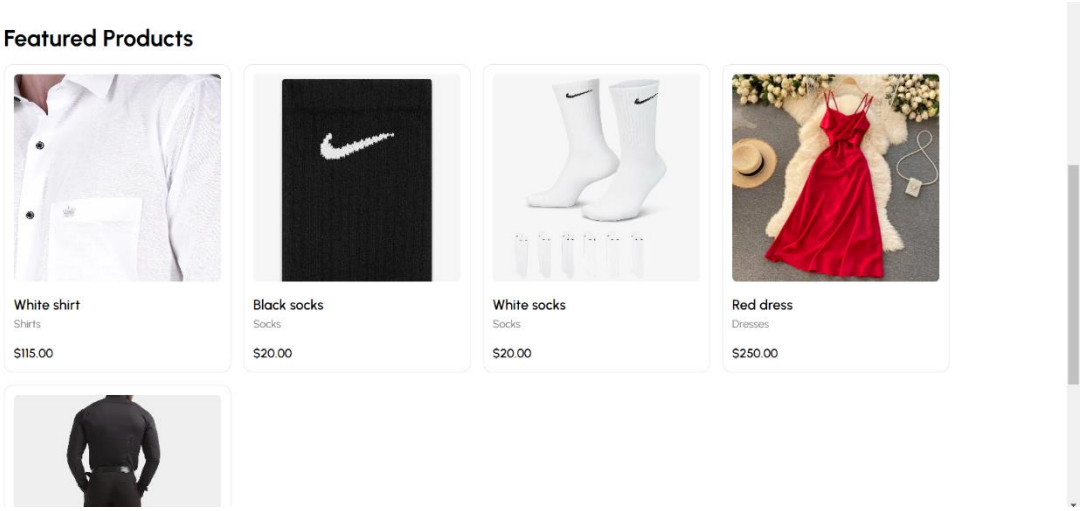


Рисунок 4.31 – Додані продукти через CMS на веб-сайті

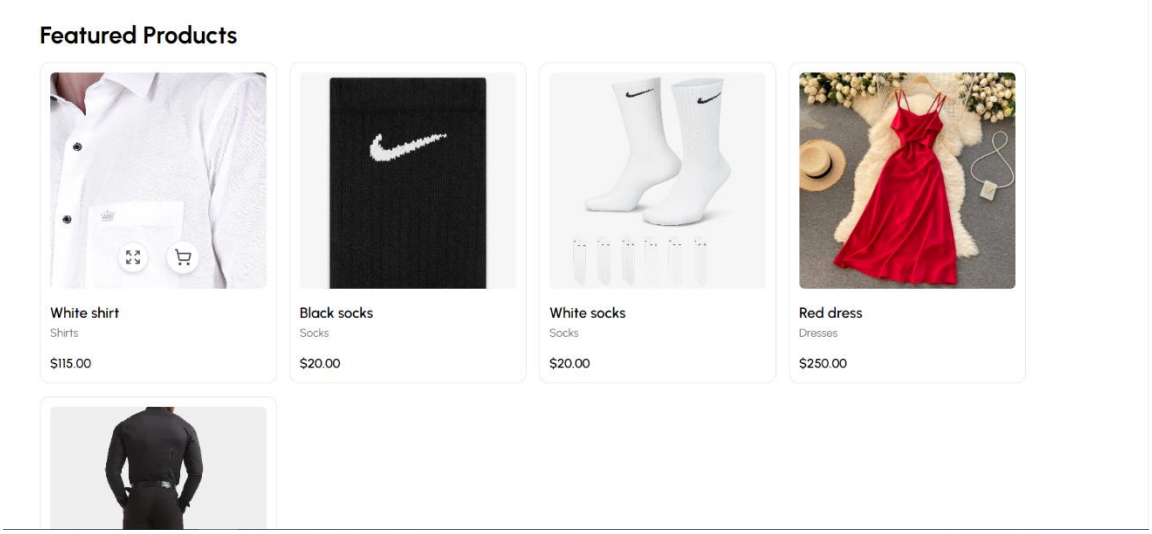


Рисунок 4.32 – Кнопки швидкого додавання до корзини та відкриття модального вікна продукту

Якщо натиснути на ліву кнопку, котра відповідає за відкриття поп-апа продукту, то можна побачити всі фото відповідного товару, ціну, характеристики та кнопку додати до корзини.

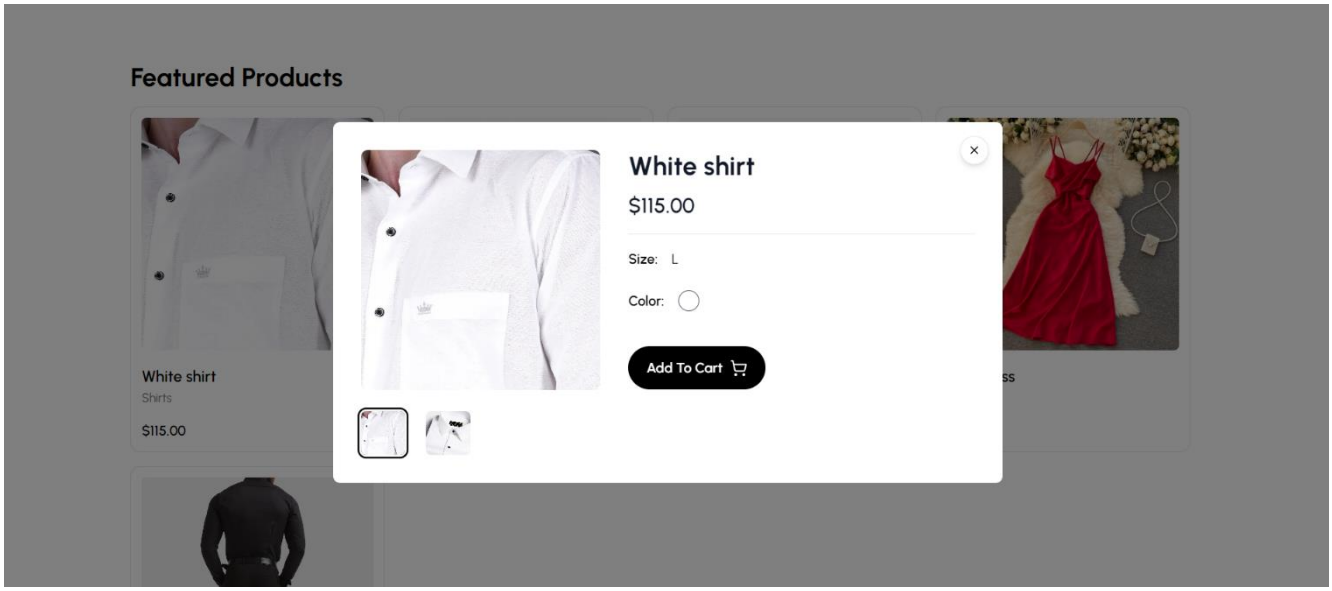


Рисунок 4.33 – Модальне вікно продукту

Перейшовши на будь-яку з категорій продуктів, що є доступною зверху можна побачити відповідно другий доданий банер, котрий використовується для цього товару, зону з фільтрами та усі доступні варіанти для придбання.

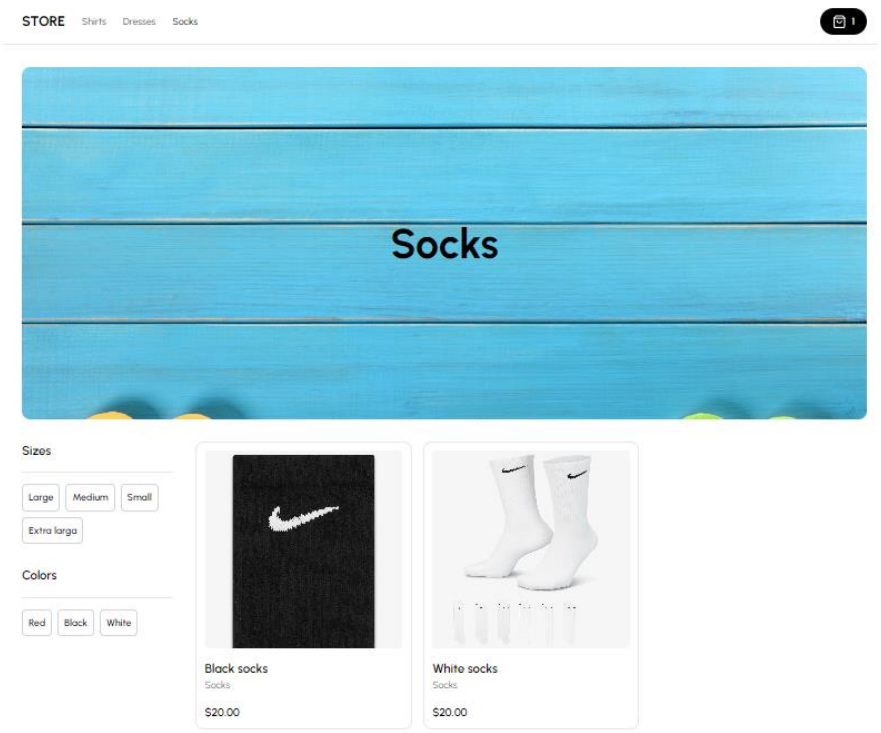


Рисунок 4.34 – Фільтрація продуктів за розміром та кольором

Після успішного додавання до корзини можна перейти до відповідної сторінки, де продовжити придбання обраного товару.

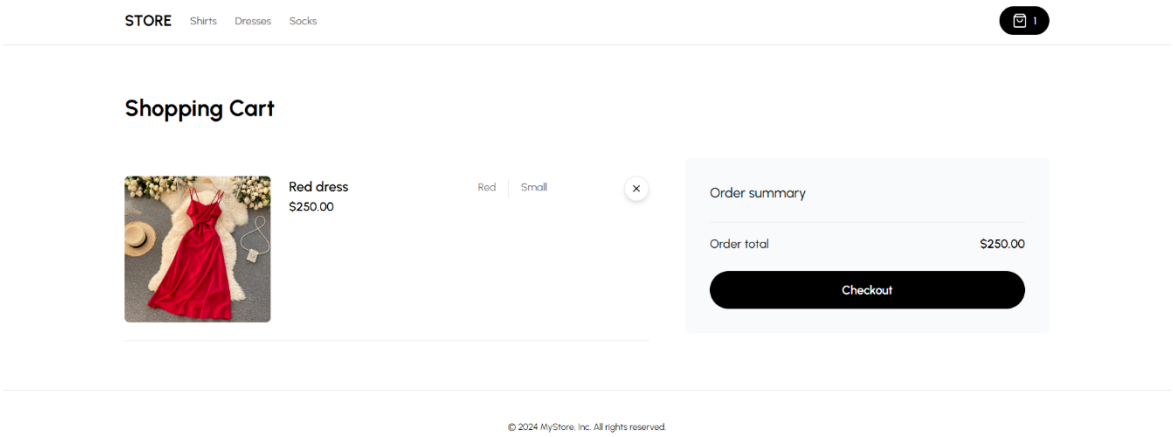


Рисунок 4.35 – Розроблена корзина з доданим продуктом

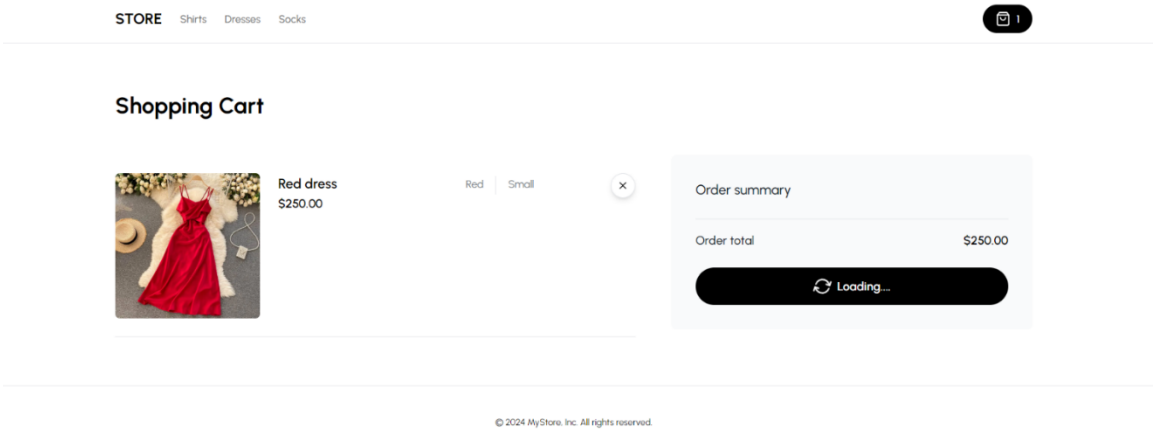


Рисунок 4.36 – Процес покупки товару

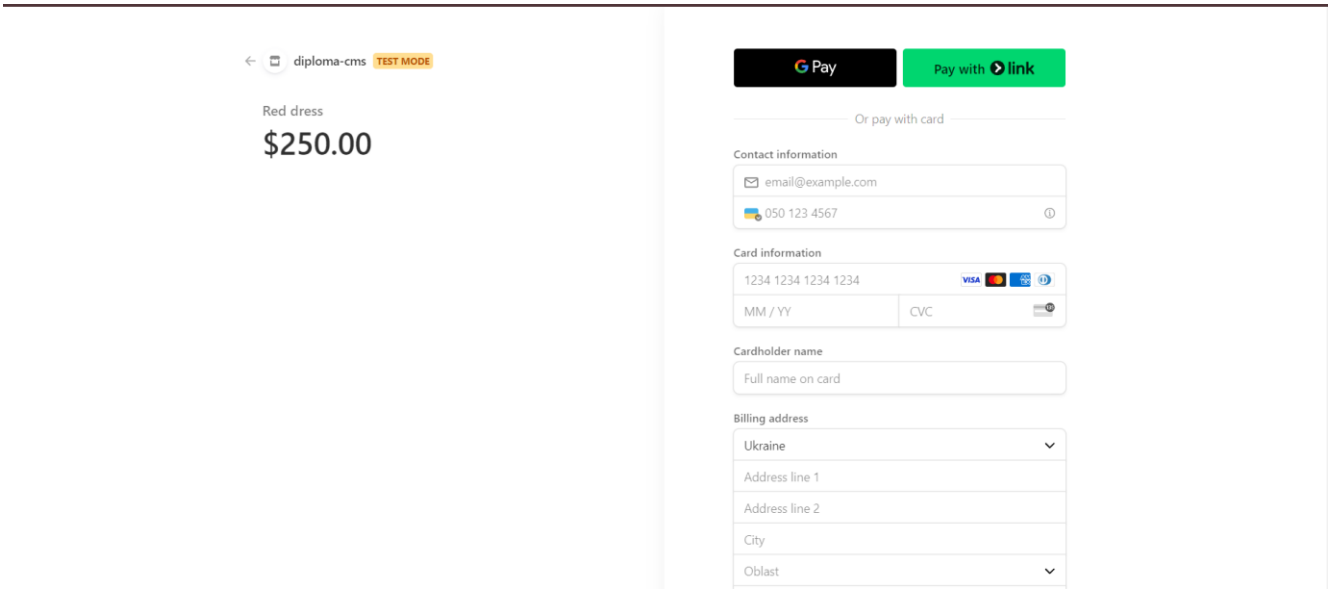


Рисунок 4.37 – Процес оплати товару через Stripe

Останнім же кроком стала перевірка метрик за допомогою PageSpeed, що впливає на видачу сайту у пошукових системах, показує його недоліки, тощо.

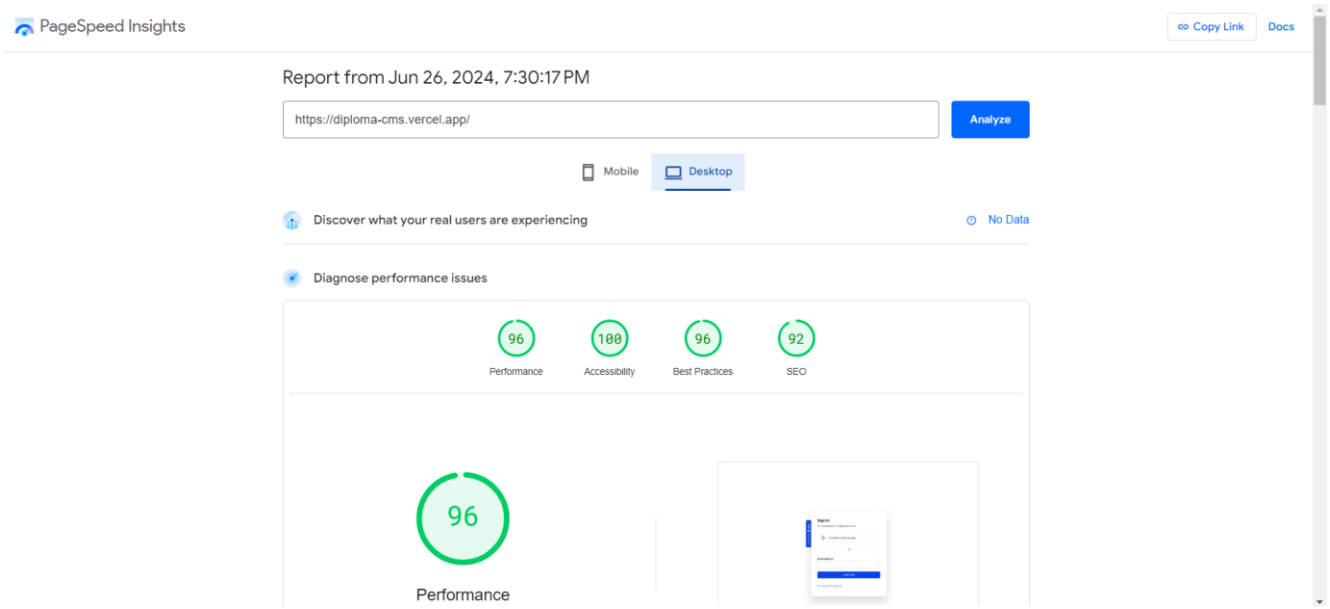


Рисунок 4.38 – Оцінка метрик сайту CMS на десктопних пристроях

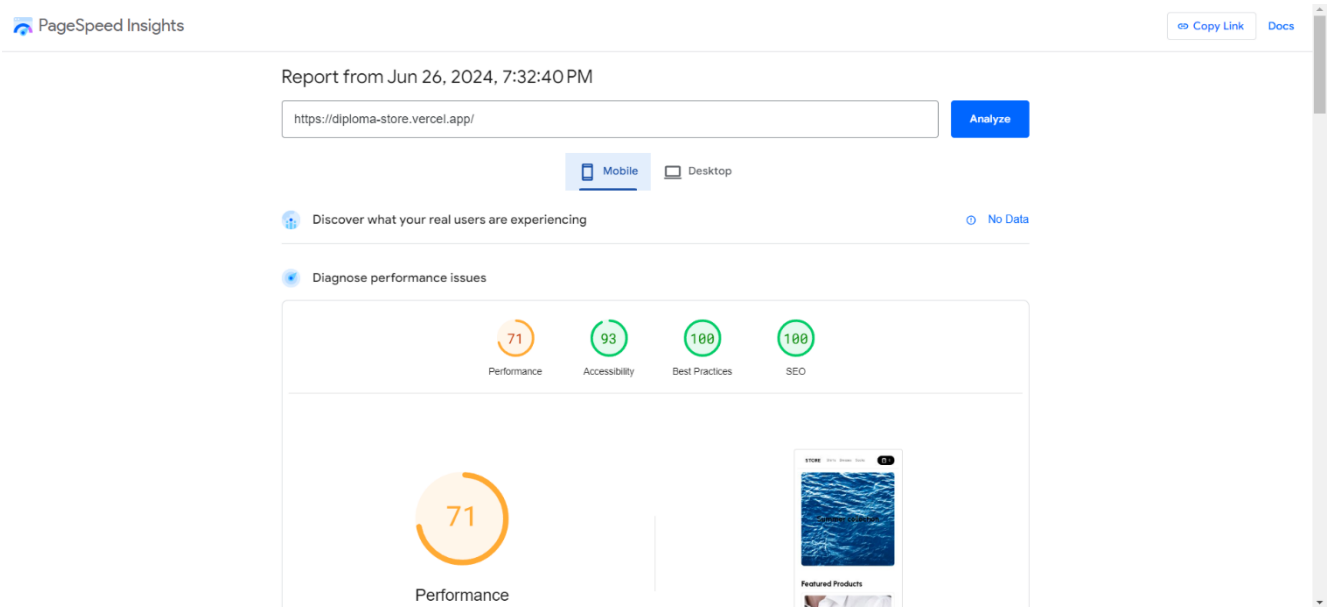


Рисунок 4.39 – Оцінка метрик розробленого веб-сайту за допомогою адміністрації контенту на мобільних пристроях

Стає зрозуміло, що обидва продукти чудово оптимізовані і готові для комерційного використання бізнесом у своїх цілях.

ВИСНОВКИ

Отже, в результаті цієї роботи було детально розглянуто різноманітні системи керування контентом, створено та протестовано унікальний продукт для компанії “TopEuroGoods” з фокусом на галузь електронної комерції.

Розробивши відповідну систему стало зрозуміло, що зараз є широкий вибір постачальників CMS: чисті безголові CMS, безголові CMS наступного покоління з frontend’ами на кшталт Next.js, конструктори сайтів на кшталт Wix та інші різновиди. Щоб зрозуміти та обрати справді таку систему адміністрації вмісту, яка підходить під відповідні бізнес задачі потрібно витратити не одну годину часу, тому цей процес є дуже ретельним.

Також стало зрозуміло, що система управління контентом – це потужний інструмент, який може зробити управління та оновлення веб-сайту набагато простішим та ефективнішим. CMS дозволяє користувачам створювати, редагувати та публікувати контент, організовувати та керувати контентом за допомогою категорій та тегів, керувати користувачами та дозволами, інтегруватися з іншими інструментами та платформами, а також надавати аналітику та звіти про відвідуваність та залученість веб-сайту. При створені системи керування контентом було враховувано потреби малого та середнього бізнесу у галузі електронної комерції під час повномасштабної війни в Україні та післяковідної кризи.

Було представлено схему архітектури створеного проєкту, його опис, та опис створення веб-додатку на основі попередньо розробленої системи керування вмістом. Також було наведено перелік функцій, що вміє розроблена система адміністрування вмісту. На етапі проєктування було проаналізовано предметну область, описано задачі та обґрунтовано вибір технологій для створення, що є невід’ємним кроком, адже саме використання сучасних веб-фреймворків, по-перше, дуже пришвидшило розробку, по-друге, зробило веб-застосунок конкурентноспроможним з такими гігантами як WordPress, Joomla!, Drupal,

Shopify, Contentful. Також було проведено повне мануальне тестування веб-додатку, та виконано його оптимізацію, результати котрої були представлені за допомогою сервісу PageSpeed та Lighthouse.

Результати дослідження можуть бути використані для покращення роботи комп'ютерного комплексу компанії “TopEuroGoods” та створять ефективну та безпечну IT-інфраструктуру, що сприятиме подальшому розвитку бізнесу та підвищенню конкурентоспроможності на ринку. Виконані роботи забезпечили досягнення поставлених цілей та підтвердили доцільність обраних технічних рішень. Також вони проєктувались з можливістю розширення, та можуть бути використані модернізації існуючої інфраструктури, коли бізнес буде зростати.

ПЕРЕЛІК ПОСИЛАНЬ

1. Forbes home – [Електронний ресурс] – Режим доступу до ресурсу: https://www.forbes.com/home-improvement/internet/internet-statistics/

2. World Bank – [Електронний ресурс] – Режим доступу до ресурсу: https://www.worldbank.org/en/publication/wdr2022/brief/chapter-1-introduction-the-economic-impacts-of-the-covid-19-crisis#:

3. United Nations Development Programme – [Електронний ресурс] – Режим доступу до ресурсу: https://www.undp.org/sites/g/files/zskgke326/files/2024-02/UNDP-UA-assessment-war-impact-enterprises-ukraine-summary.pdf

4. Блог Дріса Буйтаерту – [Електронний ресурс] – Режим доступу до ресурсу: https://dri.es/from-content-management-to-digital-experience-management

5. Web technology surveys – [Електронний ресурс] – Режим доступу до ресурсу: https://w3techs.com/

6. Чистий код: створення і рефакторинг за допомогою Agile / Мартін Роберт, пер. з англ. І. Бондар-Терещенко. – Харків : Вид-во “Ранок” : Фабула, 2024.

7. Robert C. Martin (2018). Clean Architecture: A Craftsman's Guide to Software Structure and Design.

8. Martin Kleppmann (2017). JavaScript. The Definitive Guide. Master the World's Most-Used Programming Language 7th Edition.

9. Adam Freeman (2021). Essential TypeScript 4: From Beginner to Pro.

10. Mario Casciaro, Luciano Mammino (2020). Node.js Design Patterns - Third Edition By Mario Casciaro , Luciano Mammino.

11. Computing history – [Електронний ресурс] – Режим доступу до ресурсу: https://www.computinghistory.org.uk/det/6986/The-first-commercial-modem-is-manufactured/

12. WFHResearch – [Електронний ресурс] – Режим доступу до ресурсу: https://wfhresearch.com/

13. TopDesignFirms – [Електронний ресурс] – Режим доступу до ресурсу: https://clutch.co/

14. Siteefy – [Електронний ресурс] – Режим доступу до ресурсу: https://siteefy.com/

15. IBISWorld – [Електронний ресурс] – Режим доступу до ресурсу: https://www.ibisworld.com/

16. Statista – [Електронний ресурс] – Режим доступу до ресурсу: https://www.statista.com/chart/30734/browser-market-share-by-region/

17. StateofCSS – [Електронний ресурс] – Режим доступу до ресурсу: https://2023.stateofcss.com/en-US/css-frameworks/

18. Law Insider – [Електронний ресурс] – Режим доступу до ресурсу: https://www.lawinsider.com/dictionary/patent-purity-cleanliness

19. Stack Overflow Survey – [Електронний ресурс] – https://survey.stackoverflow.co/2023/

20. OVHcloud – [Електронний ресурс] – https://www.ovhcloud.com/en/web-hosting/uc-cms-comparison/

Додаток А

Частини коду розробленої CMS та веб-сайту для компанії “TopEuroGoods”

Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
“ДНІПРОВСЬКА ПОЛІТЕХНІКА”

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ВЕБ-ЗАСТОСУНКІВ ДЛЯ КОМПАНІЇ “TopEuroGoods”

Текст програми

804.02070743.24010-01 12 01

Листів 13

2024

АНОТАЦІЯ

Ця анотація містить в собі деякі частини програмного коду веб-застосунку для компанії “TopEuroGoods”, а саме: розроблені моделі бази даних для Prisma ORM, інструменту, що дозволяє легко працювати з базами даних; під’єднання вебхука від Stripe з отриманням адреси покупця, країни, міста/населеного пункту, поштового коду, котрий буде “слухати” подію покупки, та робити переадресацію при бажанні клієнта придбати товар; приклад створення компоненту модального вікна, з описаними props’сами, що приймає компонент, та з вирішенням проблеми гідрації за допомогою використання стану “isMounted”, що виникає під час рендерингу клієнтського компоненту у серверному; код головного файлу шаблону проєкта з усіма провайдерами, що потрібні для отримання окремих станів деяких компонентів, що глобально впливають на веб-сайт; приклад створення одного з action’ів, що створює запит до бази даних через Prisma та підраховує суму усіх замовлень; код компоненту корзини з забороною кешування, адже для корзини використовується localStorage, що вже зберігає усі продукти додані до корзини.

Програма призначена для надавання бізнесу можливості швидкого розгортання в умовах війни, пандемії, кризи, тощо. Надаються необхідні інструменти управління контентом для B2C платформи, забезпечуючи тим самим їхню конкурентоспроможність та здатність розвиватися в умовах постійних змін.

Текст програми написана мовами TypeScript та JavaScript, відлагоджена із застосуванням середовища Webstorm.

Зміст

1 Розроблені моделі бази даних для Prisma ORM

```
generator client {
  provider = "prisma-client-js"
}

datasource db {
  provider     = "mysql"
  url          = env("DATABASE_URL")
  relationMode = "prisma"
}

model Store {
  id         String      @id @default(uuid())
  name       String
  userId     String
  createdAt  DateTime    @default(now())
  sizes      Size[]      @relation("StoreToSize")
  updatedAt  DateTime    @updatedAt
  billboards Billboard[] @relation("StoreToBillboard")
  categories Category[]  @relation("StoreToCategory")
  colors     Color[]     @relation("StoreToColor")
  products   Product[]   @relation("StoreToProduct")
  orders     Order[]     @relation("StoreToOrder")
}

model Billboard {
  id         String     @id @default(uuid())
  storeId    String
  store      Store      @relation("StoreToBillboard", fields: [storeId], references: [id])
  label      String
  imageUrl   String
  createdAt  DateTime   @default(now())
  updatedAt  DateTime   @updatedAt
  categories Category[]

  @@index([storeId])
}

model Category {
  id          String    @id @default(uuid())
  storeId     String
  store       Store     @relation("StoreToCategory", fields: [storeId], references: [id])
  billboardId String
  billboard   Billboard @relation(fields: [billboardId], references: [id])
  name        String
  createdAt   DateTime  @default(now())
  updatedAt   DateTime  @updatedAt
  products    Product[] @relation("CategoryToProduct")

  @@index([storeId])
```

```
  @@index([billboardId])
}

model Size {
  id        String    @id @default(uuid())
  storeId   String
  store     Store     @relation("StoreToSize", fields: [storeId], references: [id])
  name      String
  value     String
  createdAt DateTime  @default(now())
  updatedAt DateTime  @updatedAt
  products  Product[]

  @@index([storeId])
}

model Color {
  id        String    @id @default(uuid())
  storeId   String
  store     Store     @relation("StoreToColor", fields: [storeId], references: [id])
  name      String
  value     String
  createdAt DateTime  @default(now())
  updatedAt DateTime  @updatedAt
  products  Product[]

  @@index([storeId])
}

model Product {
  id         String      @id @default(uuid())
  storeId    String
  store      Store       @relation("StoreToProduct", fields: [storeId], references: [id])
  categoryId String
  category   Category    @relation("CategoryToProduct", fields: [categoryId], references: [id])
  name       String
  price      Decimal
  isFeatured Boolean     @default(false)
  isArchived Boolean     @default(false)
  sizeId     String
  size       Size        @relation(fields: [sizeId], references: [id])
  colorId    String
  color      Color       @relation(fields: [colorId], references: [id])
  createdAt  DateTime    @default(now())
  updatedAt  DateTime    @updatedAt
  images     Image[]
  orderItems OrderItem[]

  @@index([storeId])
  @@index([categoryId])
  @@index([sizeId])
```

```
  @@index([colorId])
}

model Image {
  id        String   @id @default(uuid())
  productId String
  product   Product  @relation(fields: [productId], references: [id], onDelete: Cascade)
  url       String
  createdAt DateTime @default(now())
  updatedAt DateTime @updatedAt

  @@index([productId])
}

model Order {
  id         String      @id @default(uuid())
  storeId    String
  store      Store       @relation("StoreToOrder", fields: [storeId], references: [id])
  orderItems OrderItem[]
  isPaid     Boolean     @default(false)
  phone      String      @default("")
  address    String      @default("")
  createdAt  DateTime    @default(now())
  updatedAt  DateTime    @updatedAt

  @@index([storeId])
}

model OrderItem {
  id        String  @id @default(uuid())
  orderId   String
  order     Order   @relation(fields: [orderId], references: [id])
  productId String
  product   Product @relation(fields: [productId], references: [id])

  @@index([orderId])
  @@index([productId])
}
```

2 Під’єднання вебхука від Stripe

```
import Stripe from 'stripe';
import { headers } from 'next/headers';
import { NextResponse } from 'next/server';

import { stripe } from '@/lib/stripe';
import prismadb from '@/lib/prismadb';

export async function POST(req: Request) {
  const body = await req.text();
  const signature = headers().get('Stripe-Signature') as string;

  let event: Stripe.Event;

  try {
    event = stripe.webhooks.constructEvent(body, signature, process.env.STRIPE_WEBHOOK_SECRET!);
  } catch (error: any) {
    return new NextResponse(`Webhook Error: ${error.message}`, { status: 400 });
  }

  const session = event.data.object as Stripe.Checkout.Session;
  const address = session?.customer_details?.address;

  const addressComponents = [
    address?.line1,
    address?.line2,
    address?.city,
    address?.state,
    address?.postal_code,
    address?.country,
  ];

  const addressString = addressComponents.filter((c) => c !== null).join(', ');

  if (event.type === 'checkout.session.completed') {
    const order = await prismadb.order.update({
      where: {
        id: session?.metadata?.orderId,
      },
      data: {
        isPaid: true,
        address: addressString,
        phone: session?.customer_details?.phone || '',
      },
      include: {
        orderItems: true,
      },
    });
```

```
    const productIds = order.orderItems.map((orderItem) => orderItem.productId);

    await prismadb.product.updateMany({
      where: {
        id: {
          in: [...productIds],
        },
      },
      data: {
        isArchived: true,
      },
    });
  }

  return new NextResponse(null, { status: 200 });
}
```

3 Приклад створення компоненту модального вікна

```
'use client';

import { FC, useEffect, useState } from 'react';

import { Modal } from '@/components/ui/modal';
import { Button } from '@/components/ui/button';

interface AlertModalProps {
  isOpen: boolean;
  onClose: () => void;
  onConfirm: () => void;
  loading: boolean;
}

export const AlertModal: FC<AlertModalProps> = ({ isOpen, onClose, onConfirm, loading }) => {
  const [isMounted, setIsMounted] = useState(false);

  useEffect(() => {
    setIsMounted(true);
  }, []);

  if (!isMounted) {
    return null;
  }

  return (
    <Modal
      title="Are you sure?"
      description="This action cannot be undone."
      isOpen={isOpen}
      onClose={onClose}
    >
      <div className="pt-6 space-x-2 flex items-center justify-end w-full">
        <Button disabled={loading} variant="outline" onClick={onClose}>
          Cancel
        </Button>
        <Button disabled={loading} variant="destructive" onClick={onConfirm}>
          Continue
        </Button>
      </div>
    </Modal>
  );
};
```

4 Код головного файлу шаблону проєкта з усіма провайдерами

```
import type { Metadata } from 'next';
import { Inter } from 'next/font/google';
import './globals.css';
import { ReactNode } from 'react';
import { ClerkProvider } from '@clerk/nextjs';
import { ModalProvider } from '@/providers/modal-provider';
import { ToasterProvider } from '@/providers/toast-provider';
import { ThemeProvider } from '@/providers/theme-provider';

const inter = Inter({ subsets: ['latin'] });

export const metadata: Metadata = {
  title: 'Admin Dashboard',
  description: 'Create your own Admin Dashboard',
};

export default function RootLayout({
  children,
}: Readonly<{
  children: ReactNode;
}>) {
  return (
    <ClerkProvider>
      <html lang="en">
        <body className={inter.className}>
          <ThemeProvider attribute="class" defaultTheme="system" enableSystem>
            <ToasterProvider />
            <ModalProvider />
            {children}
          </ThemeProvider>
        </body>
      </html>
    </ClerkProvider>
  );
}
```

5 Приклад створення одного з action’ів, що створює запит до бази даних через Prisma

```
import prismadb from '@/lib/prismadb';

export const getTotalRevenue = async (storeId: string) => {
  const paidOrders = await prismadb.order.findMany({
    where: {
      storeId,
      isPaid: true,
    },
    include: {
      orderItems: {
        include: {
          product: true,
        },
      },
    },
  });

  const totalRevenue = paidOrders.reduce((total, order) => {
    const orderTotal = order.orderItems.reduce((orderSum, item) => {
      return orderSum + item.product.price.toNumber();
    }, 0);
    return total + orderTotal;
  }, 0);

  return totalRevenue;
};
```

6 Код компоненту корзини з забороною кешування

```
'use client';

import { useEffect, useState } from 'react';

import { Container } from '@/components/ui/container';
import useCart from '@/hooks/use-cart';

import { Summary } from './_components/summary';
import { CartItem } from './_components/cart-item';

export const revalidate = 0;

const CartPage = () => {
  const [isMounted, setIsMounted] = useState(false);
  const cart = useCart();

  useEffect(() => {
    setIsMounted(true);
  }, []);

  if (!isMounted) {
    return null;
  }

  return (
    <div className="bg-white">
      <Container>
        <div className="px-4 py-16 sm:px-6 lg:px-8">
          <h1 className="text-3xl font-bold text-black">Shopping Cart</h1>
          <div className="mt-12 lg:grid lg:grid-cols-12 lg:items-start gap-x-12">
            <div className="lg:col-span-7">
              {cart.items.length === 0 && (
                <p className="text-neutral-500">No items added to cart.</p>
              )}
              <ul>
                {cart.items.map((item) => (
                  <CartItem key={item.id} data={item} />
                ))}
              </ul>
            </div>
            <Summary />
          </div>
        </div>
      </Container>
    </div>
  );
};
```

```
export default CartPage;
```