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 Фінько Станіслава Андрійовича
(ПІБ)

академічної групи 123-20ск-1
(шифр)

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

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

на тему “Комп’ютерна система Інтернет-магазину IT компанії з реалізацією побудови, налаштування та безпеки корпоративної мережі”
(назва за наказом ректора)

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

Дніпро
2023

ЗАТВЕРДЖЕНО:
завідувач кафедри
інформаційних технологій
та комп'ютерної інженерії
(повна назва)
_____ Гнатушенко В.В.
(підпис) (прізвище, ініціали)

« » 2023 року

ЗАВДАННЯ
на кваліфікаційну роботу
ступеня бакалавр

студента Фінько С.А. **академічної групи** 123-20ск-1
(прізвище та ініціали) (шифр)

спеціальності 123 «Комп'ютерна інженерія»

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

на тему "Комп'ютерна система Інтернет-магазину IT компанії з реалізацією побудови, налаштування та безпеки корпоративної мережі"
затверджену наказом ректора НТУ «Дніпровська політехніка» від 16.05.2023 № 350-с

Розділ	Зміст	Термін виконання
Стан питання та постановка завдання	На основі матеріалів виробничих практик, інших науково-технічних джерел конкретизується предмет та мету роботи та виконується постановка завдання	22.05.2023
Розробка апаратної частини	На основі аналізу підприємства формулюються технічні вимоги до комп'ютерної системи та розробляється апаратна частина системи	02.06.2023
Розробка корпоративної мережі	Виконується розрахунок налаштувань корпоративної мережі та перевірка роботи системи, розробляються методи та налаштування обладнання для захисту інформації в системі	10.06.2023
Розробка компонента системи	Розробка компонента системи	25.06.2023

Завдання видано _____ доц. Соколова Н.О.
(підпис керівника) (прізвище, ініціали)

Дата видачі 16.05.2023

Дата подання до екзаменаційної комісії 01.07.2023

Прийнято до виконання _____ Фінько С.А.
(підпис студента) (прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 103 с., 29 рис., 5 табл., 1 дод., 39 джерел.

СИСТЕМА, БЕКЕНД, HTTP, ІНТЕРНЕТ-МАГАЗИН, IP, КОРПОРАТИВНА МЕРЕЖА

Об’єкт розробки − комп’ютерна система Інтернет-магазину з реалізацією побудови, налаштування та безпеки корпоративної мережі.

Мета роботи − створення та налаштування комп’ютерної системи для обслуговування і забезпечення функціювання Інтернет-магазину для IT компанії.

У першій частині кваліфікаційної роботи було розглянуто структуру підприємства, завдання до кваліфікаційної роботи, шляхи його вирішення та було обрано один з них. Також було приведено опис обраних технологій для вирішення поставленої задачі.

У другій частині кваліфікаційної роботи було розроблено вимоги до розробляємого рішення. Було обрано апаратну частину для комп’ютерної мережі, розроблено структурну схему обладнання мережі підприємства та топологічну схему розміщення апаратної частини підприємства.

У третій частині було зроблено розрахунки IP адресації для підрозділів підприємства, та проведено налаштування моделі системи корпоративної мережі із застосуванням програми Cisco Packet Tracer.

В четвертій частині роботи було розроблено бекенд частину Інтернет-магазину з документацією до неї. Також була додана система авторизації користувачів з реалізованим інтерфейсом, та видачею відповідних прав доступу до кінцевих точок. Було налаштовано цикл постійної інтеграції додаваємого коду на віддалений репозиторій GitHub, з автоматичним проходженням тестів та перевірки на відповідність стандартам написання коду.

Розроблений комплекс інженерно-програмних рішень може бути використаний компанією замовником в якості фундаменту для переходу з фізичної до електронної комерції.

ЗМІСТ

Стор.

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАК, ОДИНИЦЬ І ТЕРМІНІВ

API	– прикладний програмний інтерфейс (англ. Application Programming Interface);
CI	– практика постійної інтеграції коду (англ. Continuous Integration);
DNS	– система доменних імен (англ. Domain Name System);
DHCP	– протокол динамічного налаштування вузла (англ. Dynamic Host Configuration Protocol);
HTTP	– протокол передачі гіпертексту (англ. Hypertext Transfer Protocol);
TCP/IP	– набір протоколів мережі Інтернет (англ. Transmission Control Protocol/Internet Protocol);
LAN	– локальна мережа (англ. Local Area Network);
REST	– підхід до архітектури мережевих протоколів (англ. Representational State Transfer);
SQL	– мова структурованих запитів (англ. Structured Query Language);
VLAN	– віртуальна локальна мережа (англ. Virtual Local Area Network);

ВСТУП

В сучасних умовах існування, для будь-якої торгової компанії, онлайн торгівля це важлива частина бізнесу, без якої масштабування дається вкрай важко. Деякі компанії реалізують товар на вже існуючих платформах, деякі ж обирають інший шлях, створюючи свій власний Інтернет-магазин.

Будь-який Інтернет-магазин, як складний застосунок, буде складатися з бекенд та фронтенд частини. Продумане та якісне налаштування структури проекту та бекенд частини, буде слугувати надійним фундаментом для будь-якого веб-додатоку, та дозволить в подальшому масштабувати та модифікувати проект, з мінімальними ризиками на створення значних помилок, та втраті у складності підтримки наявного коду та інфраструктури. Оскільки від якості роботи з даними, і налаштування доступу до них, напряму залежить репутація та доходи підприємства, то бекенд-частині Інтернет-магазину має приділятися першочергова та основна увага в розробці веб-додатку.

Для якісної роботи всього циклу обслуговування клієнтів Інтернет-магазину, та самого веб-додатку, неможливо обійтися без комп’ютерної мережі, використовуючи яку співробітники компанії зможуть виконувати свою роботу.

Метою кваліфікаційної роботи є розробка комп’ютерної мережі з бекенд частиною Інтернет-магазину.

1 СТАН ПИТАННЯ І ПОСТАНОВКА ЗАВДАННЯ

1.1 Стисла характеристика сфери та умов застосування комп'ютерної системи для Інтернет-магазину

Під час війни та пандемії Covid-19 Україна зіткнулася зі значними викликами, які серйозно вплинули на традиційну галузь роздрібної торгівлі. Обмеження на роботу більшості підприємств, складні умови пересування та обмеження на відвідування магазинів призвели до значного падіння продажів у магазинах та торгових центрах. За таких обставин розширення бізнесу в онлайн-середовище є важливим і необхідним рішенням.

Електронна комерція стала дуже популярною формою торгівлі, яка надає бізнесу нові можливості та переваги. Інтернет-магазин дозволяє збільшити аудиторію клієнтів за рахунок територіальної незалежності. Таким чином, власник Інтернет-магазину може залучати клієнтів з будь-якої точки країни. Крім того, зменшення витрат на оренду приміщень для фізичних точок продажу, оплату комунальних послуг та заробітну плату працівників знижує собівартість товарів і послуг, що, в свою чергу, стимулює попит на них.

Напередодні війни другий рік поспіль спостерігалася важлива тенденція: двократне переважання темпів зростання онлайн над оффлайн. 22% респондентів зазначили, що почали купувати частіше онлайн, і лише 9% купують частіше оффлайн. І ця тенденція лише посилювалася. Багато в чому завдяки наслідкам пандемії, яка привчила багатьох людей купувати онлайн [1-4].

Оскільки на українському ринку вже є кілька відомих Інтернет-магазинів, які мають значний досвід, можна розглянути можливість співпраці з ними, але часто відомі компанії вважають за краще створювати власний Інтернет-магазин для впізнаваності бренду і мати більшу гнучкість в налаштуванні власної системи.

Однак запуск власного Інтернет-магазину часто вимагає великих інвестицій на початковому етапі. Крім того, необхідно мати достатній рівень

кваліфікації в IT-технологіях, знання маркетингу та економіки, щоб успішно вести бізнес в онлайн-середовищі. Тому завжди хорошим рішенням буде звернутися до аутсорсингових компаній, які допоможуть вирішити поставлені завдання максимально ефективно.

У світлі викликів, спричинених війною та пандемією, перехід до онлайн-комерції може стати важливим кроком у розвитку бізнесу. Такий перехід забезпечить безпеку та зручність покупок для клієнтів, а також дозволить збільшити продажі та зберегти бізнес у складний період.

1.2 Характеристика і структура об’єкта впровадження

Аутсорс компанія Yalantis займається наданням послуг зі створення на замовлення інформаційних систем для вирішення бізнес питань різних компаній-замовників. Одним з найбільш популярних напрямків впровадження IT рішень є роздрібнена торгівля [5].

Компанія замовник, ТОВ «АВР Solutions», є суб’єктом у сфері роздрібненої торгівлі в місті Дніпро, та розглядає можливості щодо реорганізації свого бізнесу, шляхом переходу до онлайн торгівлі. Для ефективного впровадження нових рішень та оптимізації витрат було впроваджено нову структуру підприємства, та були орендовані нові офісні приміщення. Таким чином для розміщення різних підрозділів підприємства було вирішено орендувати офісні приміщення на 3-му поверсі одному з бізнес-центрів в центрі міста Дніпро.

Бізнес-центр з орендованими приміщеннями компанії розміщений в багатоповерховій офісній будівлі, за адресою м.Дніпро, вул. Шевченка, 53.

Гео-розміщення бізнес-центру, в якого були орендовані офісні приміщення компанії Інтернет-магазину представлено на рисунку 1.1.

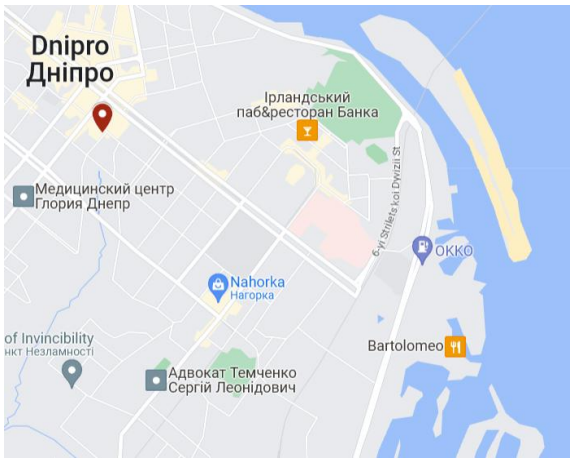

Рисунок 1.1 – Гео-розміщення бізнес-центру в якому було арендовані приміщення для потреб компанії

У бізнес-центрі, за адресою вул. Шевченка, 53, було арендовано 8 офісних приміщень на третьому поверсі, а також 2 кімнати під серверні приміщення, та одну кімнату для головного системного адміністратора. Схема третього поверху будинку була наведена на рисунку 1.2.

Рисунок 1.2 – Схема третього поверху за адресою вул. Шевченка, 53

До організаційної структури підприємства відносяться наступні відділи: відділ логістики, відділ продажу, бухгалтерія та IT.

Так, відділ логістики відповідає за управління процесами, пов'язаними з виконанням замовлень, управлінням запасами, пакуванням і відправкою замовлень, а також доставкою та поверненням товарів, проданих компанією. Виконання замовлень передбачає зберігання продукції на складі або в розподільчому центрі, відбір товарів для кожного замовлення та передачу їх перевізнику для доставки. центрі, відбір товарів для кожного замовлення та передачу їх перевізнику для доставки.

Управління запасами передбачає відстеження рівня запасів і поповнення їх за потреби. Пакування та відправка замовлень передбачає вибір відповідних пакувальних матеріалів, захист крихких предметів та вибір найкращого перевізника для кожного замовлення. Доставка та повернення передбачає забезпечення доставки замовлень клієнтам вчасно та в належному стані, а також вирішення будь-яких питань чи скарг, які можуть виникнути у клієнтів.

Відділ логістики відіграє вирішальну роль у забезпеченні задоволеності клієнтів, зниженні витрат і збільшенні прибутку компанії. Він також повинен адаптуватися до мінливих вимог та очікувань клієнтів, таких як швидка доставка, безкоштовна доставка, легке повернення тощо.

Відділ продажів відповідає за отримання доходу від продажу продуктів або послуг, які пропонує компанія клієнтам. До складу відділу збуту входять менеджер з продажу та спеціалісти з продажу.

Менеджер з продажу відповідає за нагляд за командою з продажу, встановлення цілей і цілей продажів, розробку стратегій і планів продажів, моніторинг ефективності продажів і результатів, а також надання зворотного зв’язку та інструктаж спеціалістів з продажу. Менеджер з продажу також підтримує зв’язок з іншими відділами, такими як логістика та зі спеціалістами по маркетингу, щоб забезпечити плавний і ефективний процес продажу. Фахівці з продажу – це ті, хто безпосередньо взаємодіють з клієнтами і переконують їх купити продукти чи послуги, які пропонує компанія. Фахівці з

продажу повинні мати чудові навички спілкування та ведення переговорів, а також знання продукту та навички обслуговування клієнтів. Фахівці з продажу також повинні стежити за клієнтами після продажу, розглядати будь-які скарги чи проблеми, шукати рекомендацій або повторювати покупки.

Відділ продажів відіграє вирішальну роль у забезпеченні задоволеності клієнтів, підвищенні впізнаванності бренду та збільшенні частки ринку компанії. Він також має адаптуватися до мінливих уподобань і очікувань клієнтів, таких як пропонування знижок, безкоштовної доставки, легкого повернення тощо. Відділ продажів може використовувати різні інструменти або платформи для оптимізації своїх операцій, наприклад, онлайн-магазини, соціальні мережі, маркетинг електронною поштою тощо.

Бухгалтерія компанії відповідає за облік, звітність та аналіз фінансових операцій і результатів діяльності компанії. До складу бухгалтерії входять головний бухгалтер і фінансист.

Головний бухгалтер відповідає за нагляд за системою бухгалтерського обліку, забезпечення дотримання стандартів і правил бухгалтерського обліку, підготовку та перевірку фінансових звітів і звітів, управління податковими зобов'язаннями та перевірками, а також нагляд за персоналом бухгалтерії. Головний бухгалтер також підтримує зв’язок з іншими відділами, такими як продажі, логістика та зі спеціалістами по маркетингу, для надання фінансової інформації та вказівок.

Фінансист – це той, хто керує грошовими потоками та бюджетом компанії, а також фінансовою та інвестиційною діяльністю. Фінансист повинен мати відмінні аналітичні та прогнозні навички, а також знання фінансових ринків та інструментів. Фінансист також повинен стежити за фінансовими ризиками та можливостями для компанії, а також консультувати керівництво щодо найкращих фінансових стратегій і рішень.

Бухгалтерський відділ відіграє вирішальну роль у забезпеченні фінансової точності, прозорості та підзвітності компанії. Він також містить цінну інформацію та рекомендації щодо підвищення прибутковості та

ефективності компанії. Бухгалтерський відділ може використовувати різні інструменти або платформи для оптимізації своїх операцій, наприклад, бухгалтерське програмне забезпечення онлайн, хмарні рішення, інтеграцію з платформами електронної комерції тощо.

Відділ розробки веб-сайту: Цей відділ відповідає за проектування, розробку, тестування та підтримку веб-сайту Інтернет-магазину та його функцій. Він складається з власника продукту, керівника команди та технічного відділу, який включає DevOps спеціаліст, backend, frontend розробників та тестувальників. Власник продукту визначає бачення та дорожню карту для веб-сайту, керівник команди керує процесом розробки та координує роботу з іншими відділами, а технічний відділ реалізує функціональність веб-сайту та забезпечує його якість та продуктивність.

Інженер DevOps – це той, хто застосовує принципи та практики DevOps на різних етапах життєвого циклу розробки програмного забезпечення. Роль та обов'язки DevOps інженера на різних етапах розробки та розширення функціоналу Інтернет-магазину можуть включати:

– проектування, створення, тестування та підтримка процесу безперервної інтеграції та безперервної доставки (CI/CD): Сюди входить налаштування та конфігурація інструментів і платформ, які дозволяють автоматизувати інтеграцію коду, тестування, розгортання та зворотній зв'язок. Інженер DevOps також гарантує, що процес CI/CD є надійним, безпечним і масштабованим;

– вибір найкращих інструментів і технологій, необхідних команді для задоволення бізнес-потре. Це передбачає дослідження, оцінку та рекомендацію найбільш підходящих інструментів і технологій для різних аспектів життєвого циклу розробки програмного забезпечення, таких як контроль версій, управління конфігурацією, хмарні сервіси, контейнеризація, моніторинг і ведення журналів;

– автоматизація різних фаз конвеєра DevOps: Це передбачає написання скриптів або коду, які автоматизують такі завдання, як забезпечення,

конфігурація, розгортання, тестування та усунення несправностей. Інженер DevOps також оптимізує процес автоматизації, щоб зменшити кількість помилок, підвищити ефективність і якість.

Управління IT-інфраструктурою: Це передбачає забезпечення, конфігурацію, обслуговування та оновлення IT-інфраструктури, яка підтримує програмні системи, такі як сервери, мережі, бази даних та сховища. Інженер DevOps також гарантує, що IT-інфраструктура є доступною, продуктивною та безпечною.

Надання послуг за викликом: Це передбачає готовність надати технічну підтримку та допомогу іншим командам або користувачам, коли виникають проблеми з програмними системами або інфраструктурою. Інженер DevOps також усуває несправності та вирішує проблеми швидко та ефективно.

Відділ просування сайту: Цей відділ відповідає за створення та управління контентом і маркетинговими кампаніями для веб-сайту Інтернет-магазину. Він складається з контент-менеджерів і маркетологів. Контент-менеджери створюють і оновлюють контент сайту, наприклад, описи товарів, зображення, відео та блоги. Фахівці з маркетингу планують і реалізують маркетингові стратегії в Інтернеті, такі як SEO, соціальні мережі, маркетинг електронною поштою та платна реклама.

Відділ обслуговування IT-інфраструктури: Цей відділ відповідає за нагляд і забезпечення безперебійної роботи IT-інфраструктури Інтернет-магазину, зокрема серверів, мереж, систем безпеки. Він також надає технічну підтримку і допомогу іншим відділам коли це необхідно. Відділ обслуговування IT-інфраструктури здійснює моніторинг IT-систем, виконує регулярне резервне копіювання та оновлення, усуває несправності, закуповує та встановлює нове обладнання та програмне забезпечення, а також забезпечує відповідність стандартам якості та законодавчим вимогам.

На схемі організаційної структури підприємства, яка наведена на рисунку 1.3, приведено кожний відділ, та та його відповідний склад у структурі підприємства.

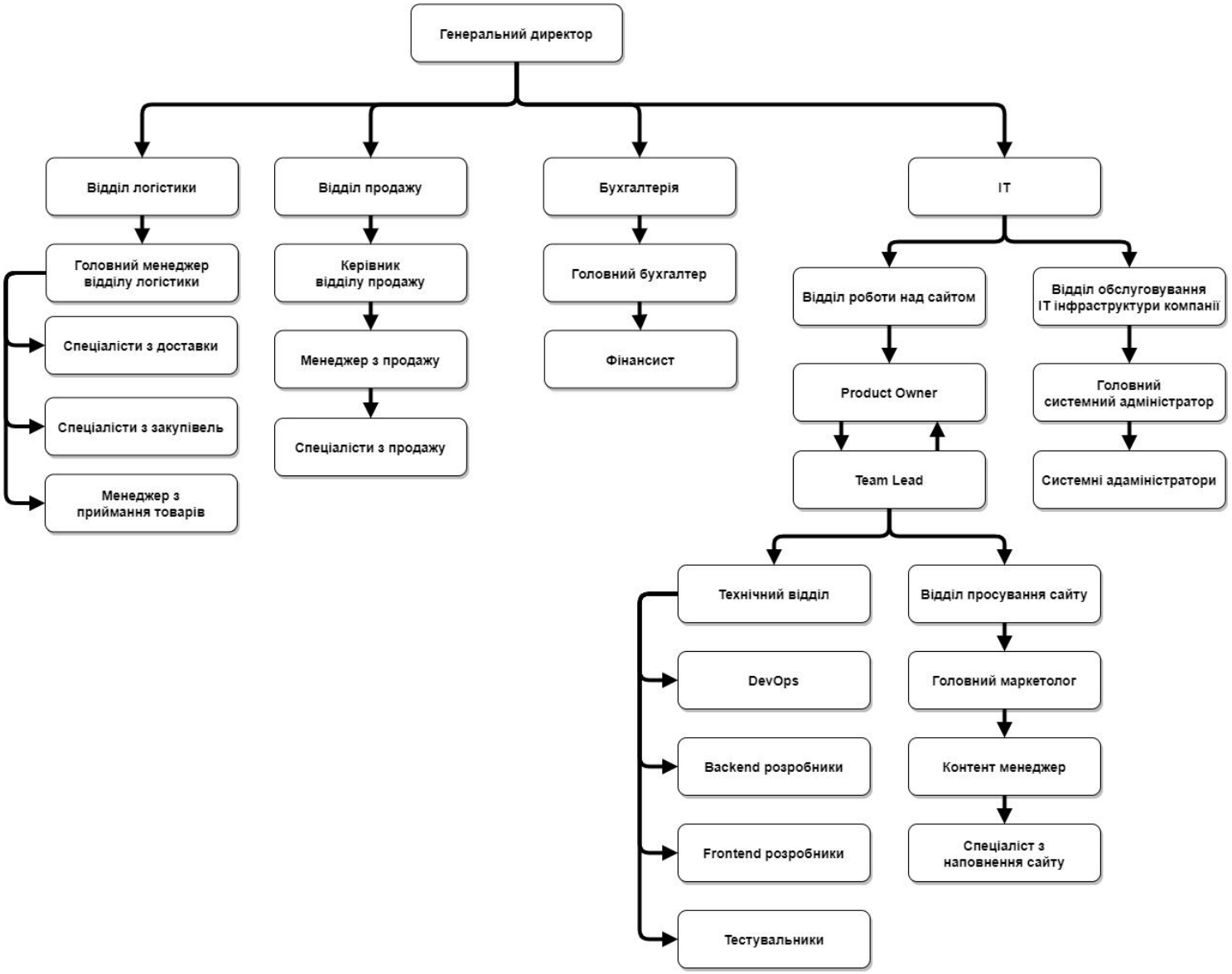

Рисунок 1.3 – Організаційна структуру підприємства

В орендованих офісних приміщеннях 3-го поверху, за адресою вулиця Шевченка, 53, передбачено розміщення генерального директора, всього відділу бухгалтерії, у складі головного бухгалтера та фінансиста; також у на цьому ж поверсі заплановано розміщення відділу продажу, у складі керівника відділу продажу, 18 спеціалістів з продажу. Також передбачено розміщення головного системного адміністратора, який буде відповідальним за інфраструктуру компанії, делегуючи задачі іншим системним адміністраторам, або самостійно працюючи над поточними задачами. Перелічені співробітники компанії також ділять третій поверх зі співробітниками відділу логістики, який включає наступний персонал: головного менеджера з відділу логістики, менеджера з приймання товарів, 2 спеціаліста з закупівель, та 9 спеціалістів з доставки.

Орендовані офіси третього поверху будівлі були також призначенні для спеціалістів частини IT відділу, а саме відділу технічного обслуговування сайту Інтернет-магазину, у який входять: спеціаліст напрямку DevOps, по 30 програмістів зі спеціалізацією Backend, стільки ж зі спеціалізацією Frontend, а також 25 тестувальників, на тому ж поверсі також призначено місце для Product Owner-a, Team Lead-a, та системного адміністратора.

Також, на поверху були розміщені спеціалісти одного з підрозділів IT відділу, а саме підрозділ з просування сайту, а також спеціалісти відділу продажу. До складу спеціалістів підрозділ з просування сайту входять: головний маркетолог, 7 контент менеджерів, та 20 спеціалістів з наповнення сайтів. У відділ продажу входять співробітники наступних спеціалізацій: 47 спеціалістів з продажу, менеджера з продажу та системного адміністратора.

1.3 Завдання та мета роботи

Компанія замовник наразі має декілька фізичних магазинів у Дніпрі, з яких здійснюється реалізація товару. Через високий попит на товар, та наявні ресурсні можливості до розширення бізнесу, компанія розглядає можливість у розширенні клієнтської бази на інші регіони України, шляхом створення власного сервісу для Інтернет торгівлі. Для налаштування шляхів

обслуговування, модерації Інтернет-магазину та віддаленому обслуговуванні клієнтів, виникла необхідність у оренді офісних приміщень для розміщення співробітників компанії різних спеціалізацій та напрямків, задачею яких має бути налагодження та підтримка повного циклу розробки та функціонування Інтернет-магазину. Таким чином виникла необхідність у налаштуванні корпоративної мережі з встановленням обладнання яке може знадобитися спеціалістам для якісного виконання поставлених задач, та створити бекенд частину нтернет-магазину з базою даних для неї. Якісна розробка бекенд частини, налаштування інфраструктури та всіх необхідних засобів на початковому етапі розгортання нових потужностей компанії дасть надійний фундамент для подальшого розширення функціоналу Інтернет-магазину. Також задля забезпечення якості використовуваємого коду Інтернет-магазину, який будуть додавати або змінювати співробітники компанії необхідно створити конвеєр безперервної інтеграції коду, тобто забезпечити автоматичну перевірку до стандартів і тестування нового коду, щоразу як він буде додаватись на репозиторій проекту на GitHub [6].

1.4 Визначення можливих напрямків рішення поставлених завдань

Для створення комп'ютерної мережі для компанії Інтернет-магазину можна використовувати кабелі Ethernet Gigabit і FastEthernet для підключення таких пристроїв, як комп'ютери, принтери, сервери і комутатори та маршрутизатори.

Ethernet – це сімейство комп'ютерних мережевих технологій, які зазвичай використовуються в локальних мережах (LAN) і глобальних мережах (MAN). Це дротове з'єднання, яке дозволяє пристроям взаємодіяти один з одним шляхом передачі пакетів даних. Ethernet був вперше розроблений корпорацією Xerox в 1970-х роках і з тих пір став найбільш широко використовуваною технологією локальних мереж.

Ethernet використовує протокол під назвою Carrier Sense Multiple Access with Collision Detection (CSMA/CD) для уникнення колізій даних. Цей протокол

гарантує, що тільки один пристрій передає дані одночасно, що допомагає запобігти втраті та пошкодженню даних.

Fast Ethernet – це вдосконалена версія Ethernet, яка була представлена в 1990-х роках. Він має максимальну швидкість 100 Мбіт/с, що в десять разів швидше, ніж оригінальний Ethernet. Fast Ethernet використовує той самий протокол CSMA/CD, що і Ethernet, але має меншу затримку в обох напрямках на 100-500 біт.

Gigabit Ethernet – це ще одна вдосконалена версія Ethernet, яка була представлена в кінці 1990-х років. Він може забезпечити швидкість до 1 Гбіт/с, що в десять разів швидше, ніж Fast Ethernet. Gigabit Ethernet використовує інший протокол під назвою Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA), який допомагає запобігти колізіям даних, повністю їх уникаючи [9].

Окрім Ethernet кабілів, є й інші, як і інші середовища передачі інформації:

Wi-Fi – це технологія бездротових мереж, яка дозволяє таким пристроям, як комп'ютери (ноутбуки і настільні комп'ютери), мобільні пристрої (смартфони і носимі пристрої) та інше обладнання (принтери і відеокамери), взаємодіяти з Інтернетом.

Wi-Fi – це сімейство бездротових мережевих протоколів, заснованих на сімействі стандартів IEEE 802.11, які зазвичай використовуються для локального об'єднання пристроїв і доступу в Інтернет, дозволяючи розташованим поблизу цифровим пристроям обмінюватися даними за допомогою радіохвиль.

Wi-Fi – це протокол бездротової мережі, який пристрої використовують для зв'язку без прямих кабельних з'єднань. Це галузевий термін, який представляє тип протоколу бездротової локальної мережі (LAN), заснований на мережевому стандарті 802.11 IEEE.

Мережі Wi-Fi працюють у двох частотних діапазонах: 2,4 ГГц і 5 ГГц. Діапазон 2,4 ГГц є більш заповненим, ніж діапазон 5 ГГц, оскільки він

використовується багатьма іншими бездротовими пристроями, такими як бездротові телефони, радіоняні та Bluetooth-пристрої. Діапазон 5 ГГц менш заповнений і забезпечує вищу швидкість, але має менший радіус дії, ніж діапазон 2,4 ГГц [10].

Кабелі для обладнання передачі даних (DCE) і термінального обладнання (DTE) називаються послідовними кабелями. DCE – це пристрій, який підтримує передачу даних по послідовному телекомунікаційному каналу, в той час як DTE – це, як правило, або німий термінал, або послідовний порт на комп'ютері/робочій станції.

Послідовні кабелі використовуються для з'єднання пристроїв DCE і DTE між собою. Кабель має два кінці: один кінець підключається до пристрою DCE, а інший – до пристрою DTE. Кабель зазвичай називають послідовним кабелем або кабелем RS-232.

Стандарт RS-232 визначає електричні характеристики сигналів, значення сигналів, а також фізичні розміри і розташування роз'ємів. Стандарт визначає 25-контактний роз'єм (DB-25) для пристроїв DTE і 9-контактний роз'єм (DE-9) для пристроїв DCE.

Послідовний порт – це інтерфейс, який дозволяє передавати дані по одному біту за раз. Зазвичай він використовується для підключення до комп'ютера таких пристроїв, як модеми, принтери та миші. Послідовні порти також відомі як COM-порти.

Обладнання передачі даних (DCE) – це пристрій, який підтримує передачу даних через послідовний телекомунікаційний канал. Термін "DCE" конкретно відноситься до послідовної передачі, яка зазвичай відбувається по таких лініях, як локальна петля звичайного телефонного зв'язку (POTS), лінія цифрової мережі з інтегрованими послугами (ISDN) або лінія T1.

DCE – це, як правило, модем, DSU/CSU (Data Service Unit/Channel Service Unit) або інше обладнання для передачі даних. Пристрої DCE відповідають за забезпечення тактового сигналу, який керує передачею даних на шині.

Щоб об'єднати кілька пристроїв в одну мережу, можна використовувати комутатори. Комутатори дозволяють пристроям взаємодіяти один з одним, пересилаючи пакети даних між ними. Існує кілька типів комутаторів, зокрема некеровані, керовані та інтелектуальні комутатори. Некеровані комутатори прості та зручні у використанні, але мають обмежену функціональність. Керовані комутатори складніші, але пропонують більший контроль над мережею. Розумні комутатори є гібридом між некерованими і керованими комутаторами і пропонують деякі розширені функції, не будучи занадто складними.

Також можна використовувати локальні мережі (LAN) для підключення пристроїв у межах будівлі. Локальні мережі зазвичай використовуються для невеликих мереж, які охоплюють обмежену територію.

Для більш ефективного управління мережевим трафіком можна використовувати VLAN (віртуальні локальні мережі) . VLAN дозволяють групувати пристрої разом на основі їх функцій або розташування і відокремлювати їх від інших пристроїв в мережі. Це може допомогти зменшити перевантаження мережі та підвищити безпеку.

Якщо потрібна більша пропускна здатність, ніж може забезпечити один кабель Ethernet, можна скористатися агрегацією портів. Агрегація портів дозволяє об'єднати кілька портів Ethernet на комутаторі або маршрутизаторі в одне логічне з'єднання. Це може допомогти збільшити пропускну здатність і забезпечити резервування на випадок, якщо один з портів вийде з ладу.

Щоб об'єднати кілька мереж разом, можна використовувати маршрутизатори. Маршрутизатори дозволяють надсилати пакети даних між різними мережами, визначаючи найкращий шлях для даних. Існує кілька типів маршрутизаторів, зокрема дротові та бездротові. Дротові маршрутизатори підключаються до мережі за допомогою кабелів Ethernet і зазвичай використовуються для невеликих мереж. Бездротові маршрутизатори дозволяють пристроям підключатися до мережі бездротовим способом за допомогою Wi-Fi.

Для створення компютерної мережі можна обрати мережеве обладнання Cisco, адже Cisco це відомий бренд в мережевій індустрії, який пропонує широкий спектр обладнання для бізнесу будь-якого розміру. Серед основних переваг обладнання Cisco можна виділити наступні:

- обладнання Cisco відоме своєю надійністю та довговічністю. Воно розраховане на високі навантаження трафіку і може працювати протягом тривалого часу без будь-яких проблем;
- має високу масштабованість і може бути легко розширене в міру зростання бізнесу. Це робить його ідеальним вибором для підприємств, які прагнуть розширити свою мережеву інфраструктуру;
- постачається з вбудованими функціями безпеки, які допомагають захистити мережу від кіберзагроз. Воно також підтримує розширені протоколи безпеки, такі як VPN і брандмауери;
- Cisco пропонує відмінну підтримку клієнтів і має велику мережу сертифікованих фахівців, які можуть допомогти з будь-якими проблемами.

Серед основних недоліків обладнання Cisco можна виділити наступні:

- може бути дорогим у порівнянні з іншим мережевим обладнанням на ринку. Це може бути бар'єром для малого бізнесу або стартапів з обмеженим бюджетом;
- може бути складним у встановленні та налаштуванні, особливо для компаній, які не мають спеціального IT-персоналу;
- може бути несумісним з іншим мережевим обладнанням на ринку. Це може обмежити можливості при побудові мережевої інфраструктури.

Загалом, обладнання Cisco є чудовим вибором для підприємств, які потребують надійних, масштабованих та безпечних мережевих рішень. Однак воно може бути не найкращим вибором для малого бізнесу або стартапів, які мають обмежений бюджет або не мають спеціального IT-персоналу [9].

Для вирішення поставленої задачі та задоволення поставлених вимог можна розглянути декілька шляхів для налаштування інфрастуруктури Інтернет магазину:

- окрім створення офісної інфраструктури, створити власний сервер компанії для хостінгу усієї інфраструктури Інтернет-магазину. З плюсів: повний контроль створеним віртуальним майданчиком та даними, якими він оперує. З мінусів: збільшення витрат на встановлення та на підтримку, необхідність у самостійному створенні сталих умов для невідказної роботи серверів, складність у масштабуванні потужностей Інтернет-магазину;
- на додачу до офісної інфраструктури, розмістити увесь програмний пакет, необхідний для роботи Інтернет-магазину, та базу даних до нього, у хмарному середовищі. З плюсів: зменшення витрат на встановлення та на підтримку серверу, полегшення адміністрування сервером, більш сприятливі умови для масштабування потужностей Інтернет-магазину. З мінусів: створення умов, при яких бізнес буде певним чином залежним від послуг хостинг-сервісу.
- до офісної інфраструктури, встановити сервер, який буде обслуговувати частину сервісів, та, можливо базу даних Інтернет-магазину, а іншу частину передати на хостінг хмарним середовищам. З плюсів: помірний рівень витрат, в порівнянні зі сценарієм де треба повністю налагоджувати та підтримувати серверну інфраструктуру самостійно, отримання повного контролю, без посередників, над важливими частинами Інтернет-магазину. З мінусів: збільшений рівень витрат, в порівнянні зі сценарієм де треба повністю розміщувати увесь Інтернет-магазин у хмарі, все ще присутній певний рівень складності при масштабуванні власних серверних потужностей Інтернет магазину, все ще є певний рівень залежності від від послуг хостинг-сервісу, проте ця залежність вже не має бути критичною.

Налаштовуючи інфраструктуру для роботи спеціалістів компанї з розробки та модифікації коду Інтернет-магазину можна розглянути декілька шляхів та інструментів для обміну напрацюваннями та зберіганням написаного коду:

- обмін даними через переносний носій інформації;
- обмін даними з ПК на ПК через локалну мережу;

− сервіси контролю версій, тобто онлайн-сервіси, які дозволяють зберігати та керувати сховищами коду.

Існують наступні сервіси контролю версій:

− GitHub – це веб-хостинг для контролю версій за допомогою git. Здебільшого використовується для комп'ютерного коду. Він пропонує всі функції розподіленого контролю версій та управління вихідним кодом (SCM) Git'a, а також додає власні функції, такі як відстеження помилок, запити на функції, управління завданнями та вікі [10];

− GitLab – ще один веб-менеджер репозиторіїв Git, який забезпечує управління вихідним кодом, безперервну інтеграцію та розгортання тощо. Він пропонує такі функції, як відстеження проблем, вікі-документація та перегляд коду [11];

− Bitbucket – це веб-сервіс хостингу репозиторіїв контролю версій, що належить Atlassian, для вихідного коду та проектів розробки, які використовують системи контролю ревізій Mercurial або Git. Він пропонує такі функції, як pull requests, огляди коду та інтеграцію з іншими інструментами [12];

− SourceForge – це веб-сервіс, який пропонує розробникам програмного забезпечення централізоване онлайн місце для контролю та управління вільними та відкритими програмними проектами. Він надає такі функції, як контроль версій, відстеження помилок, інструменти управління проектами тощо [13].

Обираючи головний інструмент для бекенд частини Інтернет магазину можна розглянути ряд найбільш популярних мов програмування відповідного спрямування:

− Python – це мова програмування високого рівня, яка відома своєю простотою та читабельністю. Вона має велику бібліотеку з великою кількістю готового коду та фреймворків для бекенд-розробки [14];

− JavaScript – це популярна мова програмування, яка використовується як для фронтенд-, так і для бекенд-розробки. Вона відома своєю універсальністю та гнучкістю [15];

− PHP – це серверна мова сценаріїв, яка використовується для створення динамічних веб-сторінок. Відома своєю простотою використання та сумісністю з різними платформами [16];

− Java – це об'єктно-орієнтована мова програмування, яка використовується для створення масштабованих і безпечних веб-додатків. Відома своєю продуктивністю та надійністю [17];

− Ruby – це динамічна мова програмування з відкритим вихідним кодом, яка використовується для створення веб-додатків. Відома своєю простотою та продуктивністю [18].

Обираючи інструмент розробки більш детально, можна розлянути різні бекенд-фреймворки. Бекенд-фреймворки – це програмні бібліотеки, які надають розробникам набір інструментів і функцій для створення серверних додатків. Вони допомагають автоматизувати деякі аспекти веб-розробки, роблячи роботу швидшою та простішою. Ось кілька популярних фреймворків для бекенда: Django, Express, Laravel, Flask, Ruby on Rails, CakePHP та інші.

Коли мова йде про зберігання даних Інтернет-магазину, існує два основних підходи: SQL та NoSQL.

Бази даних SQL – це реляційні бази даних, які зберігають дані в таблицях з фіксованою схемою та структурою. Вони використовують мову SQL для взаємодії з даними і відомі своїм швидким зберіганням і відновленням даних. Бази даних SQL є вертикально масштабованими, що означає, що вони можуть обробляти більший трафік за рахунок модернізації апаратного забезпечення сервера.

З іншого боку, бази даних NoSQL – це нереляційні бази даних, які зберігають дані за гнучкою схемою. Вони використовують власний API для взаємодії з даними і є легко масштабованими та гнучкими. Бази даних NoSQL є

горизонтально масштабованими, що означає, що вони можуть обробляти більший трафік, додаючи більше серверів до кластера баз даних.

Обираючи між SQL та NoSQL для проекту, важливо враховувати такі фактори, як масштабованість, гнучкість, простота використання, підтримка спільноти та інтеграція з існуючими інструментами. SQL, як правило, краще підходить для проектів, які вимагають складних запитів або транзакцій, в той час як NoSQL краще підходить для проектів, які вимагають високої масштабованості або гнучкості.

1.5 Обґрунтування вибраного напрямку інженерного рішення

Враховуючи побажання замовника у гнучкості обслуговування створеної інфраструктури та відштовхуючись від ресурсних можливостей, було вирішено створити і налаштувати корпоративну мережу підприємства та розмістити в ній, на потужностях компанії, сервер, з я кого буде забезпечуватися доступ до Інтернет-магазину.

Було вирішено при побудуванні комп’ютерної мережі розбити її на декілька підмереж, аби забезпечити логічне розділення різних підрозділів підприємства. Таким чином було досянуто більш ефективно розподілення ресурсів, підвищений рівень безпеки мережі, та підвищення ефективності обміну інформацією між співробітниками.

Обираючи мережеве обладнання для мережі було вирішено надати перевагу обладнанню компанії Cisco. Так, для поєднання підмереж між собою було вирішено застосовувати маршрутизатори, також їх було використано таким чином, щоб створити додаткові, резервні маршрути зв’язку. Маршрутизатор забезпечує:

- з'єднання різних мереж. Маршрутизатор може об'єднувати мережі, які відрізняються за розміром, типом або протоколом;
- надсилання пакетів в потрібне місце. Маршрутизатор зчитує адресу призначення пакета і вирішує, куди його відправити далі. Він використовує таблицю маршрутизації, яка показує найкращі шляхи до різних мереж. Він

також може спілкуватися з іншими маршрутизаторами та оновлювати свою таблицю маршрутизації;

− фільтрування та захист мережевого трафіку. Маршрутизатор може перевіряти пакети і блокувати або дозволяти їх на основі правил. Він також може використовувати NAT, щоб дозволити багатьом пристроям використовувати одну публічну IP-адресу і приховати їхні приватні IP-адреси;

− надання мережевих послуг. Маршрутизатор може надавати різні послуги мережі, такі як DHCP, DNS, та інше. DHCP надає IP-адреси пристроям у мережі. DNS переводить доменні імена в IP-адреси;

− оптимізацію продуктивності мережі. Маршрутизатор може підвищити продуктивність мережі за допомогою різних методів, таких як балансування навантаження, кешування, стиснення або формування трафіку. Балансування навантаження розподіляє трафік між кількома шляхами або серверами. Кешування зберігає дані, до яких часто звертаються, локально, щоб зменшити пропускну здатність і затримки. Стиснення робить пакети даних меншими, щоб заощадити смугу пропускання і прискорити передачу. Формування трафіку контролює швидкість і обсяг трафіку, щоб запобігти заторам або забезпечити справедливість.

Окрім HTTP сервера для Інтернет-магазину було вирішено встановити DNS сервер для IT відділу, який дозволить співробітникам швидше отримувати доступ до тестових-сторінок, над якими ведеться розробка, а також до внутрішньої документації, та директив, які можуть в подальшому встановлюватись в компанії.

Також було вирішено використовувати комутатори, тобто пристрій, який з'єднує декілька пристроїв у мережі та пересилає пакети даних між ними. Призначення комутатора в мережі полягає в тому, що він забезпечує наступне:

− підключення декількох хостів. Зазвичай комутатор має велику кількість портів для кабельних з'єднань, що дозволяє здійснювати маршрутизацію за топологією "зірка";

− преадресацію повідомлень на певні хости. Як і міст, комутатор використовує однакову логіку переадресації або фільтрації на кожному порту. Коли будь-який хост в мережі або комутатор надсилає повідомлення іншому хосту в тій же мережі або тому ж комутатору, комутатор отримує і декодує кадри, щоб прочитати фізичну (MAC) частину повідомлення. Потім він пересилає повідомлення тільки на порт, до якого підключений хост-адресат, замість того, щоб транслювати його на всі порти. Це зменшує перевантаження мережі та підвищує безпеку;

− керування трафіком. Комутатор в мережі може керувати трафіком, що надходить в мережу або виходить з неї, і може легко підключати такі пристрої, як комп'ютери і точки доступу. Він також може надавати пріоритет одним пакетам над іншими на основі налаштувань якості обслуговування (QoS), що може підвищити продуктивність чутливих до затримок додатків, таких як голос і відео;

− зберігання електричного сигналу неспотвореним. Коли комутатор пересилає кадр, він регенерує неспотворений прямокутний електричний сигнал. Це допомагає зберегти цілісність даних і запобігти деградації сигналу на великих відстанях;

− збільшення пропускної здатності локальної мережі. комутатор розділяє локальну мережу на кілька доменів колізій з незалежною широкосмуговою смугою пропускання, таким чином значно збільшуючи пропускну здатність локальної мережі. Домен колізій – це сегмент мережі, де пакети даних можуть зіткнутися один з одним при одночасному надсиланні. Зменшуючи кількість пристроїв у кожній області колізій, комутатор знижує ймовірність колізій і підвищує ефективність мережі.

Для налаштування ефективного процесу розробки командою розробників було вирішено розміщувати код і файли від яких залежить код на репозиторії GitHub.

Для розробки бекенд частини Інтернет магазину та забезпечення якості та надійності коду до нього було обрано ряд технологій, які повинні бути

об’єднанні у структуру наведену на рисунку 1.4. Так, згідно зі структурою, коли розробник завантажує зміни до репозиторію на GitHub повинна автоматично пройти перевірка коду на якість і відповідність стандартам, повинно бути проведено автоматичне тестування, розрахування відсотку покриття коду тестами та передати результат перевірок до репозиторію.

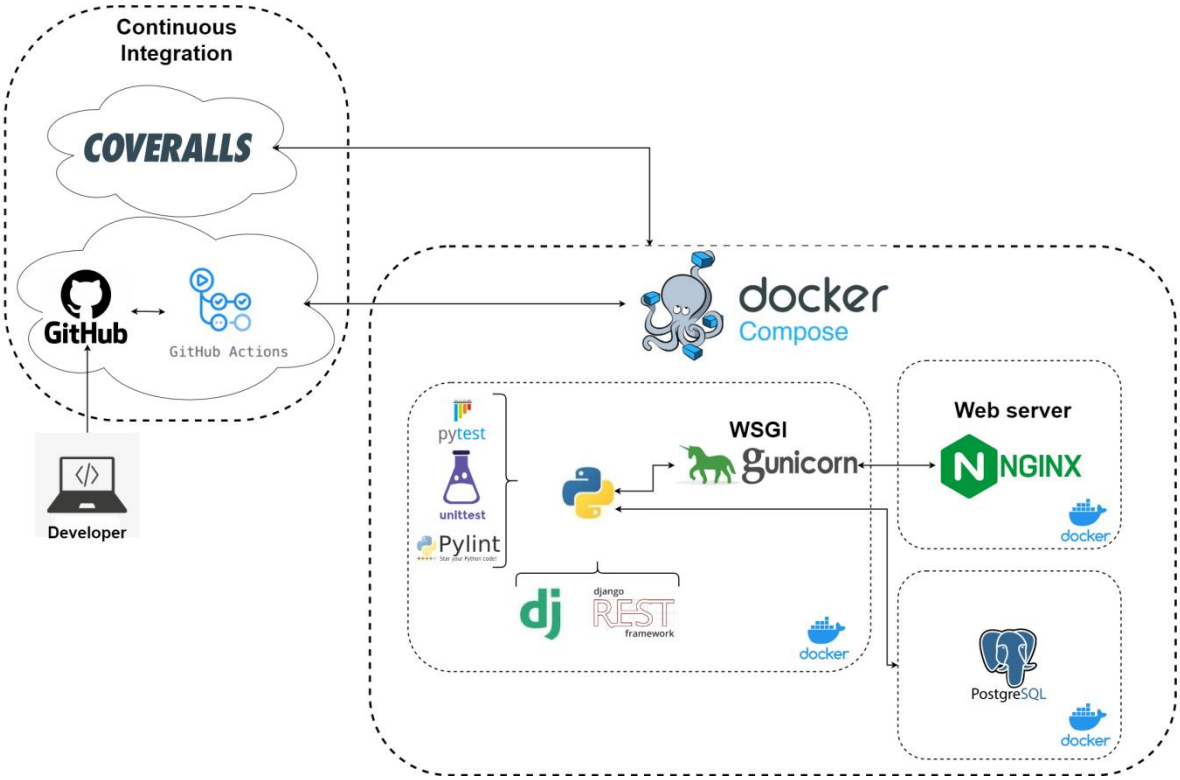

Рисунок 1.4 – Схема структури застосування основних технологій для бекенд частини Інтернет-магазину

Для розробки бекенд частини Інтернет магазину, було обрано мову програмування Python. Python – це мова програмування, яка дозволяє швидко працювати та ефективніше інтегрувати системи. Вона легка для вивчення та використання, вона підтримує кілька парадигм програмування, таких як об'єктно-орієнтоване, імперативне, функціональне та процедурне. Python добре підходить для багатьох цілей, таких як веб-розробка, розробка графічних інтерфейсів, наукові та чисельні обчислення, розробка програмного забезпечення та системне адміністрування. Python має велику та активну

спільноту розробників і користувачів, а також підтримується багатьма бібліотеками та фреймворками, які розширюють її функціональність [14].

В якості фреймворку мови програмування Python, для розробки Інтернет-магазину, було обрано фреймворк Django у поєднанні з Djano REST Framework (DRF). Django – це веб-фреймворк, який допомагає створювати веб-сайти та веб-додатки за допомогою Python. Це безкоштовний, відкритий і зрілий фреймворк, який працює за архітектурним шаблоном Model-View-Template (MVT). Він зменшує клопоти з веб-розробкою, щоб була можливість зосереджуватися на написанні програми замість того, щоб винаходити велосипед. Django має багато вбудованих функцій, які роблять веб-розробку простішою та швидшою, наприклад, автентифікація, авторизація, пагінація, сортування, форми, інтерфейс адміністратора та інструменти для тестування [19].

DRF, або Django REST Framework – це інструментарій, який допомагає створювати веб-API за допомогою Django. Веб-аплікатор – це спосіб для систем надавати корисні функції та дані іншим системам за допомогою форматів JSON або XML. DRF – це потужний і гнучкий інструментарій, який значно спрощує серіалізацію, десеріалізацію та валідацію даних. DRF також має багато функцій, які роблять розробку веб-API більш зручною та ефективною, наприклад, API, що переглядаються, API з гіперпосиланнями, користувацькі рендери, фільтрація та керування версіями [20]. Django і DRF добре працюють разом, тому що вони мають багато спільних функцій і концепцій, таких як моделі, представлення, серіалізатори і валідатори. Моделі – це класи, які визначають структуру і поведінку даних у базі даних. Представлення – це функції або класи, які обробляють запити та відповіді для веб-додатку або API. Серіалізатори – це класи, які перетворюють складні дані, такі як набори запитів і екземпляри моделей, у власні типи даних Python, які потім можна легко конвертувати в JSON, XML або інші типи вмісту. Валідатори – це функції або класи, які перевіряють дійсність вхідних даних перед тим, як зберегти їх у базі даних або повернути клієнту. Django і DRF також мають велику і активну спільноту

розробників і користувачів, які надають підтримку, зворотній зв'язок і роблять внесок у проекти.

В якості бази даних було обрано систему управління базами даних (СУБД) PostgreSQL. PostgreSQL – це система управління базами даних з відкритим вихідним кодом, яка підтримує об'єктно-реляційну модель даних та багато розширених функцій. Це одна з найстаріших і найдосконаліших систем баз даних у світі, що має понад 35 років активного розвитку та велику й активну спільноту розробників і користувачів. Вона має добру репутацію завдяки надійності, продуктивності та функціональній стійкості, оскільки може працювати з високим рівнем паралелізму, складними запитами та великими наборами даних. PostgreSQL може працювати на різних платформах, таких як Linux, macOS, Windows, BSD і Solaris, і може бути легко інтегрована з багатьма мовами програмування, такими як Python, Java, Ruby і C#. PostgreSQL може обробляти різні типи даних, такі як текст, числа, масиви, JSON, XML і геометричні типи, а також підтримує багато типів даних, які недоступні в інших базах даних, наприклад, UUID, HSTORE і RANGE. PostgreSQL також підтримує багато розширень та інструментів, які розширюють її функціональність, наприклад, pgAdmin для графічного адміністрування та psycopg2 для підключення Python [21].

Для створення документації для API було обрано бібліотеку drf-spectacular для автоматичного створення документації на основі написаного коду, у вигляді схеми OpenAPI 3.0 для Django REST Framework [22].

Для забезпечення зручної, автоматизованої та уніфікованої збірки усіх програмних компонентів Інтернет-магазину було вирішено створити інструкції до створення та об’єднання сервісів у контейнери. Для цього було вирішено застосувати Docker-compose. Docker – це інструмент, який допомагає створювати, запускати та керувати контейнерами. Контейнери – це ізольовані середовища, в яких можна запускати програми та їхні залежності, не впливаючи на хост-систему. Контейнери схожі на віртуальні машини, але вони легші та ефективніші, оскільки мають спільне ядро операційної системи і не

потребують завантаження цілої операційної системи для кожної програми. Docker дозволяє легко створювати, ділитися та розгортати контейнерні програми на різних платформах і в різних середовищах. Docker може бути використаний для створення образів, які є знімками програми та її залежностей. Згодом, можна використовувати Docker для запуску контейнерів, які є екземплярами образів. Також можна використовувати Docker для виштовхування і витягування образів з реєстрів, які є сховищами образів. Docker також надає інструменти для керування життєвим циклом контейнерів, таких як запуск, зупинка, перевірка, ведення журналів та вилучення. Docker Compose – це інструмент, який допомагає у визначені і запуску багатоконтейнерних програм за допомогою Docker. За допомогою Docker Compose можна використовувати YAML-файл для налаштування служб програми, таких як веб-сервери, бази даних, кеші тощо. Також можна вказати, як служби пов'язані між собою, як монтуються томи даних і як відкриваються порти. Потім, за допомогою однієї команди, можна створити і запустити всі сервіси зі створеної конфігурації. Docker Compose працює у всіх середовищах: виробництво, постановка, розробка, тестування, а також робочі процеси CI. Ви можете використовувати Docker Compose, щоб спростити розробку і тестування додатку, запускаючи кілька сервісів однією командою. Також можливо використовувати Docker Compose для масштабування додатку, змінюючи кількість реплік для кожного сервісу.

Для зберігання даних які створюють Docker контейнери було вирішено налаштувати використання Docker volumes (томи Docker). Відповідні налаштування вказуються у docker-compose.yml файлі.

Томи Docker – це спосіб збереження даних, створених і використовуваних контейнерами Docker. Вони повністю управляються Docker і не залежать від структури каталогів та операційної системи хост-машини. Це означає, що не потрібно турбуватися про те, де зберігаються дані на диску, як отримати до них доступ з різних контейнерів або як перемістити їх між

машинами. Docker зробить це все самостійно, тож користувачу або розробнику не доведеться про це турбуватися.

Томи мають кілька переваг над прив'язками, які є ще одним способом приєднання зовнішніх даних до контейнерів. Приєднання покладається на файлову систему хост-машини і може спричинити проблеми сумісності або ризики для безпеки. З іншого боку, томи мають такі переваги:

- томи легше створювати резервні копії або мігрувати, ніж прив'язані з'єднання. Для керування томами можна використовувати команди Docker або Docker API, а також драйвери томів для зберігання томів на віддалених хостах або у хмарних провайдерів;
- томи можна безпечніше розподіляти між декількома контейнерами. Можна приєднувати том з одного контейнера до іншого, або використовувати іменовані томи, щоб посилатися на один і той самий том у різних контейнерах;
- томи можуть мати вміст, попередньо заповнений контейнером. Можна скористатися інструкцією Volume у докер-файлі, щоб вказати один або декілька каталогів, які слід створити як томи під час запуску контейнера з цього образу;
- томи Docker Desktop мають значно вищу продуктивність, ніж прив'язані монтування з комп'ютерів Mac і Windows. Це пов'язано з тим, що томи використовують власні події файлової системи на хост-машині, тоді як прив'язування покладається на механізми опитування, які споживають ресурси центрального процесора.

Для налаштування обміну даними між контейнерами Docker було використано мережу Docker, що є функцією, яка дозволяє з'єднувати контейнери між собою та з робочими навантаженнями, що не належать до Docker. Це означає, що можливо створювати програми, які складаються з декількох контейнерів, що взаємодіють між собою або взаємодіють із зовнішніми системами чи сервісами. Наприклад, таким чином було вирішено підключити контейнер веб-сервера до контейнера бази даних.

Мережа Docker також дозволяє керувати мережею на основі діагностики платформи, незалежно від того, чи працюють Docker-хости під управлінням Linux, Windows або їх комбінації. Це означає, що можна використовувати ті самі команди та інструменти для створення та налаштування мереж на різних операційних системах, і що не потрібно турбуватися про реалізацію або деталі базової мережі. Наприклад, можна використовувати одну і ту ж команду docker network для створення мостової мережі у Linux і Windows, а Docker впорається з відмінностями самостійно [23].

Для забезпечення якості коду при розробці Інтернет-магазину було вирішено налагодити безперервну інтеграцією (CI) для проведення автоматичної перевірки якості коду та проходження тестування за завчасно написаними тестами. Також було вирішено забезпечити створення звітності з того який відсоток коду буде покритий тестами. GitHub – це платформа, яка допомагає розміщувати, керувати та співпрацювати зі сховищами коду. Сховища коду – це колекції файлів і каталогів, які містять розміщений на ньому вихідний код, документацію, тести та інші ресурси для проектів користувачів [10]. GitHub також надає безліч функцій та інструментів, які допомагають з налаштуванням безперервної інтеграції (CI), що є практикою автоматизації тестування та розгортання коду. GitHub Actions – це функція, яка дозволяє створювати робочі процеси для автоматизації конвеєра безперервної інтеграції та безперервної доставки (CI/CD) за допомогою YAML-файлів [24].

Coveralls – це інструмент, який допомагає вимірювати і покращувати покриття тестами написаного коду. Покриття тестами – це відсоток коду, який покритий тестами. Coveralls працює з GitHub і CI-сервером для збору та аналізу даних про покриття і показує, які частини коду не покриваються тестами. Coveralls також надає такі функції, як бейджі, коментарі та сповіщення, які допоможуть відстежувати та повідомляти про стан і зміни у тестовому покритті [25].

GitHub Actions і Coveralls можуть працювати разом, щоб допомогти забезпечити якість і надійність коду. Можливо використовувати GitHub для

зберігання та зміни версій коду, а також використовувати GitHub Actions для запуску тестів для кожного коміту або pull-запиту. Згодом можеливо використовувати Coveralls, щоб публікувати дані про покриття тестами на GitHub і бачити, як відсоток покриття змінюється з часом.

Для написання тестів для програмного коду Інтернет-магазину на мові Python було вирішено використовувати такі фреймворки як pytest та unittest.

Pytest – це потужний і гнучкий фреймворк, який допомагає писати і запускати тести для програм на Python. Pytest підтримує різні типи тестів, такі як модульні, функціональні, інтеграційні та інші. Можна використовувати pytest для тестування будь-якого типу коду Python, від простих функцій до складних додатків і бібліотек. Pytest також надає багато функцій і плагінів, які роблять тестування простішим і ефективнішим, наприклад fixtures: fixtures– це багаторазові функції або об'єкти, які можна використовувати для налаштування або очищення тестового середовища. Fixtures можна використовувати для різних областей видимості (наприклад, модуль, клас або сеанс) і використовувати їх у декількох тестах. Fixtures також можуть залежати один від одного або мати різні параметри [26].

Unittest – це фреймворк, який допомагає писати і запускати модульні тести для програм на Python. Unittest використовує деякі об'єктно-орієнтовані концепції для організації та виконання тестів [27].

Для перевірки якості коду та відповідності до стандартів було вирішено використати pylint.

Pylint – це потужний і всеосяжний інструмент, який аналізує код Python і перевіряє його на наявність різних типів проблем, які можуть вплинути на якість і читабельність вашого коду. Pylint є частиною проекту PyCQA, метою якого є надання спільного інтерфейсу та ресурсів для інструментів якості коду Python. Pylint доступний на PyPI і може бути встановлений за допомогою pip або інших менеджерів пакетів.

Мета Pylint полягає у тому щоб допомогти покращити якість та читабельність коду шляхом виявлення різних типів проблем, таких як:

− синтаксичні помилки, тобто помилки, які перешкоджають виконанню або компіляції коду, наприклад, пропущені дужки, невірні відступи або неправильні ключові слова. Наприклад, Pylint може виявити, якщо використовується print як інструкція замість функції у Python 3, або якщо використовується зарезервоване ключове слово як ім'я змінної;

− логічні помилки, тобто помилки, які призводять до неправильної або неочікуваної поведінки коду, наприклад, використання невизначених змінних, доступ до недійсних атрибутів або виклик некоректних методів. Наприклад, Pylint може виявити, що використовується змінна до присвоєння їй значення, або намагаєтеся отримати доступ до атрибуту, який не існує в об'єкті;

− помилки стилю, тобто помилки, які порушують PEP 8, керівництво по стилю Python, яке надає набір угод і кращих практик для написання коду на Python. Наприклад, Pylint може виявити, якщо використовується непослідовні угоди про імена для змінних, функцій або класів, або якщо перевищується максимальна довжина рядка в 79 символів;

− пропозиції щодо рефакторингу, тобто пропозиції щодо покращення коду шляхом застосування деяких поширених технік рефакторингу, таких як вилучення методів, заміна літералів константами або спрощення логічних виразів. Наприклад, Pylint може запропонувати виділити спільний фрагмент коду в окрему функцію або замінити жорстко закодоване значення на константу.

Pylint також може генерувати звіти, які показують загальну якість коду, наприклад, кількість знайдених помилок, попереджень і рефакторингів, а також цикломатичну складність. Цикломатична складність – це міра складності коду, яка базується на кількості гілок і циклів у ньому. Pylint може допомогти визначити області коду, які потребують покращення [28].

Для забезпечення надійності та безпеки ресурсів Інтернет-магазину було вирішено замінити стандартний інтерфейс шлюзу веб-сервера(WSGI) Django на Gunicorn, який рекомендований офіційною документацією фреймворка.

Gunicorn – це Python WSGI HTTP сервер для Unix. WSGI розшифровується як Web Server Gateway Interface, що є стандартним інтерфейсом між веб-серверами та веб-додатками. WSGI дозволяє веб-серверам спілкуватися з веб-додатками за допомогою загального протоколу, незалежно від базового фреймворку або мови. Gunicorn є сервером моделі pre-fork worker, що означає, що він розгалужує декілька робочих процесів від головного процесу для обробки запитів. Це дозволяє Gunicorn використовувати переваги декількох процесорів і ядер, а також ізолювати кожен запит в окремому процесі [29].

Gunicorn можна інтегрувати з різними інструментами та сервісами. Запустити Gunicorn можна з командного рядка, скрипта або інструменту з графічним інтерфейсом. Gunicorn також можна використовувати з іншими інструментами, такими як Nginx.

В якості веб-серверу було обрано Nginx. Nginx – це веб-сервер, який відомий своєю високою продуктивністю, масштабованістю, надійністю та низьким споживанням ресурсів. Nginx – це проект з відкритим вихідним кодом, який підтримується спільнотою волонтерів. Nginx може допомогти запускати веб-додатки швидше і ефективніше, надаючи такі функції, як:

− обслуговування веб-сторінок, тобто може обслуговувати статичний і динамічний контент, використовуючи різні протоколи, такі як HTTP, HTTPS та інші. Nginx підтримує різні веб-фреймворки, такі як Django, Flask і Ruby on Rails. Nginx може обробляти різні типи запитів, такі як GET, POST, PUT, DELETE і PATCH. Nginx також може обробляти різні типи відповідей, такі як HTML, JSON, XML і двійкові дані;

− зворотний проксінг, тобто може виступати в ролі зворотного проксі-сервера для інших веб-серверів або додатків. Nginx також може змінювати заголовки, переписувати URL-адреси і фільтрувати запити на основі різних критеріїв. Наприклад, Nginx може перенаправляти запити на основі імені хоста або шляху, або відхиляти запити на основі IP-адреси або агента користувача;

– балансування навантаження, тобто може розподіляти вхідний трафік між декількома висхідними серверами або додатками. Nginx також може виконувати перевірку працездатності, обхід відмов і резервне копіювання висхідних серверів. Наприклад, Nginx може перевірити, чи приймає запити висхідний сервер, відправивши запит ping або спеціальний запит. Якщо висхідний сервер виходить з ладу або стає перевантаженим, Nginx може переключитися на інший сервер або повернути відповідь про помилку;

– HTTP-кешування, тобто може кешувати статичний і динамічний контент з попередніх серверів або додатків, зменшуючи навантаження на них і покращуючи час відгуку для клієнтів. Nginx може кешувати вміст на основі різних факторів, таких як URL, заголовки або файли cookie. Nginx також може анулювати або очищати кешований вміст, коли це необхідно. Наприклад, Nginx може тривалий час кешувати зображення або CSS-файли з попереднього сервера, але при їх оновленні анулювати їх. Nginx також може очистити кешований вміст за допомогою спеціального запиту або інструменту командного рядка [30].

2 РОЗРОБКА АПАРАТНОЇ ЧАСТИНИ КОМП'ЮТЕРНОЇ СИСТЕМИ

2.1 Технічні вимоги до інформаційно-комп'ютерної системи

2.1.1 Вимоги до системи в цілому

Комп'ютерна система підтримки роздрібної онлайн-торгівлі (далі КС СПРОТ) призначена для підтримки прийняття рішень, а також здійснення торгівельних операцій через мережу Інтернет компанією ТОВ «АВР Solutions», яка здійснює торгівлю електронними виробами.

2.1.1.1 Вимоги до структури і функціонування системи

Комп'ютерна мережа має бути поділена на 5 підсистем:

− підсистема відділу продажу. Через дану підсистему мають здійснюватися продажі, через відео, аудіо, а також текстові канали комунікації з клієнтами. Повинні бути забезпеченні мінімальні характеристики мережевого з'єднання з доступом до глобальної мережі для кожного окремого вузла, а саме: мінімальна пропускна здатність має бути не менше 1Мб/с, затримка має бути не більше 150мс, втрата пакетів не 5%;

− адміністративно-бухгалтерська підсистема, з доданням персоналу з відділу продажів. У дану підсистему має надходити текстові звіти та статуси з усіх підмереж компанії, а також з цієї підсистеми має надсилатися інструкції у текстовому форматі.Через дану підсистему мають здійснюватися, продажі, через відео, аудіо, а також текстові канали комунікації з клієнтами. Повинні бути забезпеченні мінімальні характеристики мережевого з'єднання з доступом до глобальної мережі для кожного окремого вузла, а саме: мінімальна пропускна здатність має бути не менше 1Мб/с, затримка має бути не більше 150мс, втрата пакетів не 5%;

− підсистема відділу логістики Ця підсистема, окрім робочих станцій має бути забезпечена принтером та TFTF сервером. Мережа підсистеми має

задовольняти мінімальним вимогам на мережу, що включає: мінімальну швидкість у 45 Мб/с, затримку до 100мс, втрату пакетів не більше 1%;

− перша підсистема відділу IT. Підсистема має бути розбита на окремі підмережі для ізоляції робочих станцій спеціалістів різної спеціалізації. Повинні бути забезпеченні мінімальні характеристики мережевого з'єднання з доступом до глобальної мережі для кожного окремого вузла, а саме: мінімальна пропускна здатність має бути не менше 1Мб/с, затримка має бути не більше 150мс, втрата пакетів не 5%;

− друга підсистема відділу IT. Окрім робочих станцій підсистема має бути забезпечена DNS сервером. Повинні бути забезпеченні мінімальні характеристики мережевого з'єднання з доступом до глобальної мережі для кожного окремого вузла, а саме: мінімальна пропускна здатність має бути не менше 1Мб/с, затримка має бути не більше 150мс, втрата пакетів не 5%.

В системі має бути встановлений HTTP сервер, на якому буде розгорнутий Інтернет-магазин, доступ до якого буде надаватися через глобальну мережу.

Розроблена комп'ютерна мережа має бути встановлена та налаштована таким чином, аби була можливість розширення, тобто, повинна бути можливість додання нових робочих станцій чи серверів до існуючих підсистем, так і додання нових підсистем.

Функціонал бекенд частини Інтернет-магазину має мати таку структуру коду і архітектуру, яка дозволить, за потребою, розширити його шляхом додання нового функціоналу, як з бекенд, так і з фронтенд напрямку. Також, в базу даних Інтернет-магазину має бути можливість постійно додавати нову інформацію щодо нових товарів, категорій товарів, тощо.

2.1.1.2 Вимоги до показників призначення

В підмережах компанії має бути забезпечена максимальна швидкість передачі інформації до 100 мегабайт на секунду. Максимальна швидкість передачі даних між підмережами має бути не менше 1000 мегабайт на секунду.

Необхідно забезпечити технічне обслуговування робочих станцій, серверів, та мережевого обладнання кожні 3-6 місяців. Робота мережі та серверів має бути забезпечена в 99% часу, без збоїв та переривань.

Система повинна забезпечувати функціювання обладнання підключеного до комп’ютерної мережі. Комп’ютерна мережа повинна бути під'єднана до глобальної мережі. Інтернет-магазин має бути успішно розгорнутий на HTTP сервері корпоративної мережі, та до нього має бути доступ, як у клієнтів, так і у адміністраторів, з глобальної мережі та з мережі компанії для адміністративних потреб.

2.1.1.3 Вимоги до експлуатації, технічного обслуговування, ремонту і збереження компонентів системи

Умови і регламент (режим) експлуатації:

- система повинна забезпечувати виконання своїх функцій з високою надійністю і безпекою;
- система повинна бути захищена від несанкційованого доступу, втручання або пошкодження;
- система повинна мати автоматичні механізми діагностики, контролю, корекції та відновлення при виникненні помилок або збоях;
- система повинна мати резервні копії даних та програмного забезпечення для запобігання втрати інформації;
- система повинна мати можливість оновлюватися та модернізуватися без порушення її роботи.

Вимоги до параметрів мереж енергопостачання (живлення та заземлення):

- система повинна мати стабільне живлення від мережі 220 В з частотою 50 Гц;
- система повинна мати захист від перепадів напруги, короткого замикання, перегрівання та перевантаження;

− система повинна мати аварійне живлення від акумуляторних батарей або генератора на випадок відключення мережі. Час роботи системи від аварійного живлення повинен бути не менше 2 годин;

− система повинна мати надійне заземлення для запобігання статичної електрики та електромагнітних перешкод.

Регламенту обслуговування:

− система повинна підлягати плановому обслуговуванню кожні 6 місяців. Планове обслуговування повинно включати такі операції: перевірка стану апаратної частини, очищення вентиляторів та фільтрів, заміна застарілих або пошкоджених компонентів, перевірка параметрів живлення та заземлення, перевірка роботи програмного забезпечення, оновлення драйверів та антивірусних баз, дефрагментація жорсткого диску тощо;

− система повинна підлягати неплановому обслуговуванню у разі виявлення помилок або збоях у роботі системи. Непланове обслуговування повинно включати такі операції: детальна діагностика причин помилок або збоях, усунення несправностей або пошкоджень, встановлення нових компонентів або програмного забезпечення, виконання тестування та налагодження системи.

Система повинна мати комплект запасних виробів і приладів для проведення ремонту та обслуговування. Комплект повинен містити такі елементи: запасні модулі, плати, кабелі, конектори, індикатори, перемикачі, тощо. Комплект запасних виробів і приладів повинен бути під постійним обліком та інвентаризацією. Персонал повинен вести журнал видачі та повернення запасних виробів і приладів.

2.1.1.4 Вимоги до кількості, кваліфікації обслуговуючого персоналу і режимам його роботи

Система повинна мати достатню кількість обслуговуючого персоналу для забезпечення її нормальної роботи. Мінімальний склад персоналу повинен складатися з 2-х системних адміністраторів.

Обслуговуючий персонал повинен мати високу кваліфікацію та досвід роботи з комп’ютерними системами. Персонал повинен пройти спеціальне навчання та атестацію перед початком роботи з системою.

Обслуговуючий персонал повинен працювати за графіком, що забезпечує постійну наявність необхідного складу персоналу. У разі наявності лише 2-х системних адміністраторів, забезпечити позмінний графік на ніч та день. Також персонал повинен бути доступний для виклику в разі аварійних ситуацій.

2.1.1.5 Додаткові вимоги

Система повинна працювати в умовах підвищеної вологості, до 60 відсотків, та температури до 45°C.

Кабель-канали повинні бути металевими з можливістю закриття кришкою. Кабель-канали повинні мати захист за стандартом IP65. Кабель-канали повинні мати отвори для виходу кабелю в місцях підключення пристроїв.

Інформаційні розетки повинні бути типу RJ-45, або DB-9, в з залежності від потреби, з позначенням номера порту. Інформаційні розетки повинні бути розташовані на висоті не менше 0,5 м від підлоги. Інформаційні розетки повинні бути розташовані на відстані не більше 2 м від пристроїв, що підключаються до них.

Електричні розетки повинні бути типу Schuko з позначенням напруги та струму. Електричні розетки повинні бути розташовані на висоті не менше 0,5 м від підлоги. Електричні розетки повинні бути розташовані на відстані не більше 2 м від пристроїв, що підключаються до них.

Комунікаційне обладнання повинно бути розташоване в спеціальній шафі, що відповідає стандарту захисту IP65. Приміщення або шафа повинні мати замки та системи контролю доступу.

Комунікаційне обладнання повинно бути розміщене в металевих шафах з можливістю закриття дверцятами. Шафи повинні мати вентиляцію та освітлення.

2.1.2 Вимоги до функцій, виконуваних системою

Для однієї з підсистем IT відділу належать наступні функції:

− розробка нового функціоналу для Інтернет-магазину;

− тестування нового функціоналу за допомогою автоматичних та ручних тестів для перевірки його якості, надійності, безпеки та зручності.

Для другої підсистеми IT відділу притаманні наступні функції:

− аналіз ефективності Інтернет-магазину за різними показниками (трафік, конверсія, продажі, лояльність тощо);

− розробка та реалізація стратегії просівання Інтернет-магазину в Інтернеті (SEO, SMM, PPC, email-маркетинг тощо);

− створення та оновлення контенту для Інтернет-магазину (тексти, фото, відео, банери, лендінги тощо);

− керування DNS сервером для забезпечення належної роботи доменних імен Інтернет-магазину.

Для підсистемі продажів призначені наступні функції:

− обробка замовлень клієнтів Інтернет-магазину (прийом, підтвердження, відправлення, відстеження тощо);

− надання консультацій та підтримки клієнтам Інтернет-магазину (телефон, email, чат тощо);

− здійснення продажів додаткових товарів та послуг клієнтам Інтернет-магазину;

− збір та аналіз відгуків та скарг клієнтів Інтернет-магазину (опитування, форми зворотного зв’язку, соціальні мережі тощо).

Для підсистем логістики призначені наступні функції:

− приймання товарів від постачальників Інтернет-магазину (перевірка кількості, якості, терміну придатності тощо);

− закупівля товарів для Інтернет-магазину (планування, вибір постачальників, укладання договорів, оплата тощо);

− доставка товарів до клієнтів Інтернет-магазину (формування замовлень, пакування, вибір перевізників, підправка, відстеження тощо);

− керування TFTP сервером для забезпечення належної роботи принтера.

Для підсистеми основної підсистеми були призначені наступні функції:

− ведення бухгалтерського обліку і звітності Інтернет магазину (облік доходів і витрат, нарахування податків і зарплати, складання балансу та інших звітів тощо);

− контроль фінансової дисципліни і безпеки Інтернет магазину (перевірка оплати замовлень, запобігання шахрайству та втратам тощо);

− планування та аналіз фінансових показників Інтернет магазину (бюджетування, прогнозування, оцінка рентабельності та ефективності тощо).

2.1.3 Вимоги до видів забезпечення

2.1.3.1 Вимоги до інформаційного забезпечення

Структура даних у Системі повинна бути реляційною та нормалізованою до третьої нормальної форми. Дані повинні бути розподілені за сутностями та атрибутами та зв’язані за допомогою первинних та зовнішніх ключів.

Способи організації даних у Системі повинні включати:

− використання Систем керування базами даних (СКБД) для збереження та обробки даних. СКБД повинна бути розподіленою та масштабованою для забезпечення високої продуктивності та надійності Системи;

− використання стандартних форматів для представлення даних. Формати повинні бути сумісними з існуючими стандартами і протоколами. Наприклад, для представлення даних у вигляді тексту можна використовувати формат JSON або XML; для представлення даних у вигляді фото або відео можна використовувати формат JPEG або MP4 тощо.

Системи керування базами даних (СКБД) повинні бути використані для збереження та обробки даних у Системі. СКБД повинні мати наступні характеристики:

- реляційність, тобто СКБД повинні підтримувати реляційну модель даних, що дозволяє зберігати та маніпулювати даними за допомогою таблиць, рядків, стовпців та зв’язків між ними;
- розподіленість, тобто СКБД повинні підтримувати розподілену архітектуру, що дозволяє розміщувати та обробляти дані на різних вузлах мережі для покращення продуктивності та надійності Системи;
- масштабованість, тобто СКБД повинні підтримувати масштабовану архітектуру, що дозволяє збільшувати або зменшувати обсяг та швидкість обробки даних в залежності від потреб Системи;
- безпека, тобто СКБД повинні підтримувати безпечну архітектуру, що дозволяє захищати дані від несанкційованого доступу, модифікації або втрати за допомогою шифрування, аутентифікації, авторизації тощо.

Дані повинні бути контрольовані за допомогою різних методів і засобів для перевірки їх правильності, повноти, актуальності та консистентності. Наприклад, для контролю даних можна використовувати перевірки на коректність формату і типу даних; перевірки на наявність обов’язкових атрибутів і значень; перевірки на унікальність дублювання і суперечливості даних тощо.

2.1.3.2 Вимоги до лінгвістичного забезпечення

При розробці проекту бекенд частини Інтернет-магазину Python має бути використаний в якості основної мови програмування. Так, при використанні фреймворку Django, для отримання доступу до бази даних можна уникнути використання напряму мовою маніпулювання даними SQL, натомість можна використати Django ORM.

Для шифрування та дешифрування паролів користувачів можна використати алгоритм PBKDF2. Для кодування і декодування текстових даних можна використовувати кодову таблицю UTF-8.

Уся технічна документація до проекту має надаватися Англійською мовою.

2.1.3.3 Вимоги до технічного забезпечення

HTTP сервер має мати хоча б 128 Гб оперативної пам’яті, та 2 Тб основної, хоча б один процесор на 2 ядра з тактовою частотою від 1.7 Гц.

TFTP сервер має мати хоча б 16 Гб оперативної пам’яті, та 2 Тб основної, хоча б один процесор на 2 ядра з тактовою частотою від 1.7 Гц.

Обираємий маршрутизатор від компанії Cisco має мати не менше 4-х портів RJ-45 та мати підтримку розширення для модулів NIM-1T або NIM-4T.

Мережа підприємства має бути забезпечена 5-ма комутаторами на 24 порти RJ-45, та 3-ма на 48 відповідно.

Робочі станції мають мати як мінімум: 16 Гб оперативної пам’яті DDR4, 240 Гб SSD з форм-фактором у 2.5".

2.1.3.4 Вимоги до організаційного забезпечення

Новий персонал має бути має пройти через етап навчання по завчасно підготовленим інструкційним матеріалам та документацією, як до обладнання з яким він буде взаємодіяти, так і з інформаційним забезпеченням системи, до того як приступити до роботи.

Повинні проводитися регулярні інструктажі з розбором частих помилок, та повторенням матеріалу документацій та інструкційних матеріалів для співробітників компанії.

2.1.3.5 Вимоги до методичного забезпечення

Повинні бути створені нормативи експлуатації, які повинні визначати правила та рекомендації щодо експлуатації комп’ютерної системи та її

компонентів, а також щодо запобігання та усунення несправностей та аварій. Нормативи експлуатації повинні містити такі розділи: технічне обслуговування, діагностика, ремонт, модернізація тощо.

Повинна бути створена документація для API, куди повинні бути включені: перелік всіх кінцевих точок, їх адреси, опис, методи запитів за протоколом HTTP, та приклади запитів з вказанням типів даних для запиту.

2.2 Розробка апаратної частини комп'ютерної системи

2.2.1 Визначення та обґрунтування структурної схеми комплексу технічних засобів комп'ютерної системи на основі врахування структури та топологічних особливостей об'єкту розробки

Відповідно до організаційної структури підприємства та розроблених технічних вимог було спроектовано топологічну схему розміщення структурних підрозділів підприємства (рис. 2.1).

Рисунок 2.1 – Топологічна схема розміщення структурних підрозділів підприємства

Таким чином, кожна підсистема була виділена в окрему підмережу. Було вирішено розмістити маршрутизатори та сервери у серверній кімнаті у спеціальних шафах. Кімната серверної, як і всі шафи, мають дверці, які закриваються на замок, доступ до них мають системні адміністратори та директор. У сусідній кімнаті до серверної розміщена кімната головного системного адміністратора з його робочою станцією, яка підключена до маршрутизатора провайдера. Також до маршрутизатора провайдера під'єднаний HTTP сервер.

Підмережу LAN 3, де розміщені робітники підрозділу розробки Інтернет-магазину виділено 3 офісних приміщення, де в одному з них було розміщено 3 комутатори, 2 на цю саму підмережу, LAN 3, і 1 для іншої частини IT підрозділу компанії, який був розміщений у окремому офісному приміщенні, цій підмережі присвоєно назву LAN 5. Комутатори розміщені у спеціальній шафі, яка закривається на замок, ключ до якого є у системних адміністраторів та директора. До комутатора підмережі LAN 5 під'єднаний DNS сервер, який розміщений в серверній кімнаті.

У сусідньому офісному приміщеннях було розміщено підмережу LAN 2, для бухгалтерського підрозділу, директора та частини робітників підрозділу з продажу. У цьому ж офісі було розміщено 2 комутатори, які розміщені у спеціальній шафі, яка закривається на замок, ключ до якого є у системних адміністраторів та директора. 1 комутатор призначений для підмережі LAN 2, а другий для підмережі LAN 4.

Підмережа LAN 4, яка розміщена в сусідньому офісному приміщені, яка прилигає до сходів, призначена для підрозділу логістики. У підрозділі логістики окрім робочих станцій розміщено принтер. В серверній розміщено TFTF сервер, який підключається до комутатора, який відповідає до підмережі логістики.

LAN 1 – це підмережа яка розміщена в 3-х приміщеннях того ж поверху, 2 з яких це офісні приміщення, а одне друга серверна. Підмережа LAN 1 призначена для відділу продажів. В другій серверній знаходиться мережеве

обладнання для підмережі LAN 1, а саме 3 комутатори, які розподіляють навантаження між вузлами підмережі.

Усі сервери, комутатори, маршрутизатори та вузли комп’ютерної мережі з’єднанні між собою кабелями, які протягнуті вздовж стінок у спеціальному металевому кабель-каналі.

2.2.2 Розробка специфікації апаратних засобів комп’ютерної системи

В якості серверів для комп’ютерної системи було обрано 2 Cisco UCS C220 M3 LFF в різних конфігураціях, наведених у таблиці 2.1, де Cisco UCS C220 M3 LFF – це сервер для встановлення в стійку, призначений для віртуалізованих і невіртуалізованих додатків. Він забезпечує високу продуктивність, масштабованість і надійність у компактному форм-факторі. C220 M3 LFF підтримує до двох процесорів серії Intel Xeon E5-2600 і до 512 ГБ пам'яті. Він також підтримує до восьми 3,5-дюймових жорстких дисків або твердотільних накопичувачів (SSD) для зберігання даних. C220 M3 LFF ідеально підходить для широкого спектру застосувань, включаючи сервери баз даних, веб-сервери та сервери віртуалізації. Він забезпечує високу продуктивність і надійність у компактному форм-факторі, що робить його ідеальним вибором для центрів обробки даних та інших середовищ, де простір має першочергове значення.

На додаток до високої продуктивності та надійності, C220 M3 LFF також пропонує ряд розширених функцій, які полегшують керування та обслуговування. Ці функції включають підтримку Cisco UCS Manager, який забезпечує централізоване управління декількома серверами з одного інтерфейсу, а також підтримку Cisco Integrated Management Controller (IMC), який надає можливості віддаленого управління [9].

В якості маршрутизаторів було обрано 7 Cisco Catalyst 8300-1N1S-4T2X в різних конфігураціях, де Cisco Catalyst 8300-1N1S-4T2X – це модульний корпоративний маршрутизатор для філій, який є частиною периферійних платформ Cisco Catalyst серії 8300. Він призначений для прискореного надання

послуг. Маршрутизатор має 1 стійку, 10Gig WAN Cisco SD-WAN з підтримкою 5G/LTE, модульний корпоративний маршрутизатор для філій з 1 PIM, 1 NIM і 1 SM слотами. Додаткові його характеристики та перелік портів були наведені у таблиці 2.1. Маршрутизатор підтримує статичну та динамічну маршрутизацію та має підтримку DHCP [9].

У випадках, де є необхідність з'єднання маршрутизаторів послідовним кабелем, було встановлено додаткові модулі NIM-1T або NIM-4T. Cisco NIM-1T – це 1-портова послідовна інтерфейсна карта WAN, тоді як Cisco NIM-4T – це 4-портова послідовна інтерфейсна карта WAN [9].

Було обрано 5 комутаторів Cisco Catalyst 9300 IE-9320-24P4S-E, де Cisco Catalyst 9300 IE-9320-24P4S-E – це захищений комутатор, який забезпечує високошвидкісне підключення Ethernet в компактному форм-факторі. Він має 24 порти гігабітних інтерфейсів Ethernet. Його пропускна здатність комутації при максимальному навантажені в повнодуплексному режимі досягає 56 Гбіт/с, що повною мірою покриває потреби підприємства [9].

Для більш великих підмереж було обрано 2 комутатора Meraki MS355-48X-HW, який є стековим комутатором доступу, який забезпечує високопродуктивне підключення для корпоративних мереж. Він має 48 портів RJ45 з максимальною пропускною здатністю 1 Гб/с та 2 порти QSFP+. Комутатор має пропускну здатність 544 Гбіт/с і пропускну здатність стекування 400 Гбіт/с. Він також підтримує маршрутизацію 3-го рівня і UPoE [31].

Було обрано комплектуючі до персональних комп'ютерів співробітників. Так було обрано наступні комплектуючі до персональних комп'ютерів: AMD Ryzen 3 2100GE PRO [32], MSI B450M-A PRO MAX [33], 2 плашки оперативної пам'яті DDR4 8GB 2666 MHz Goodram [34], накопичувач SSD 2.5" 512GB AS350X Apacer [35]. Таким чином процесор з 2 ядрами та 4-ма потоками на 3.2 GHz, 16 Гб оперативною пам'яттю, та інтегрованою відеокартою мають забезпечити комфортну роботу та базові технічні потреби для необхідного програмного забезпечення.

В якості принтера для логістичної підмережі було обрано Canon i-SENSYS LBP710CX, бо цей кольоровий лазерний принтер, який забезпечує високу швидкість друку та велику місткість паперу. Він має частоту процесора 792 МГц, загальну вхідну ємність 200 аркушів і підтримує двосторонній друк. Він ідеально підходить для малої мережі, де потрібен високоякісний кольоровий друк. Він має невелику площу і легко вписується в обмежений простір. Крім того, Canon i-SENSYS LBP710CX простий у використанні та обслуговуванні. Через ці переваги він може повністю задовільнити потреби невеличкої підсистеми, на яку розрахований [36].

Було підібрано Ethernet кабель від компанії ATcom [37], який відноситься до категорії 6 і призначений для використання в локальних обчислювальних мережах (LAN) та інших системах передачі даних. Він має діаметр провідника 0,51 мм і складається з чотирьох пар скручених мідних проводів. Кабель також екранований екраном FTP (Foil Twisted Pair) для зменшення електромагнітних перешкод. Кабель має ПВХ оболонку, що робить його придатним для використання всередині приміщень. Кабель здатний передавати дані на високих швидкостях і може підтримувати смугу пропускання до 250 МГц. Він також сумісний з кабелями категорії 5 і категорії 5e. Кабель ідеально підходить для використання в таких додатках, як Gigabit Ethernet, Fast Ethernet. Всього передбачається до закупівлі 610м кабелю, де 215м має залишитися запасними для проведення ремонтних робіт, розширення мережі, чи будь-яких інших непередбачуваних ситуацій.

Також для поєднання частини маршрутизаторів було обрано послідовний кабель Smart Serial Back-to-Back Smart Serial DCE DTE, тобто кабель, який з'єднує два маршрутизатори Cisco з інтерфейсами Smart Serial. Кабель Smart Serial використовує роз'єм 12-в-1 Smart Serial, розроблений Cisco. Мережевий кінець кожного кабелю має фізичні роз'єми, які найчастіше використовуються для інтерфейсу [9].

Таблиця 2.1 – Таблиця специфікації обраних апаратних засобів комп’ютерної мережі

Позиція	Найменування	Технічна характеристика	Кількість	Порти	Примітки
1	Cisco UCS C220 M3 LFF	2 процесори E5-2650 v2 1.7-2.1 GHz, 8 модулів оперативної пам’яті на 32 GB 1333Mhz, 1 SSD на 100GB, 2 жорстких диски на 3 TB кожен	1	2 RJ-45	HTTP сервер
2	Cisco UCS C220 M3 LFF	2 процесори E5-2650 v2 1.7-2.1 GHz, 4 модулів оперативної пам’яті на 4 GB 1333Mhz, 1 SSD на 100GB, жорсткий диск на 2 TB	1	2 RJ-45	TFTP сервер
3	Cisco Catalyst 8300-1N1S-4T2X	2 джерела живлення, 8 GB оперативної пам’яті, 16 GB постійної пам’яті M2 SSD	7	4 RJ-45, 3 NIM	Додано 3 модуля NIM-1T та 1 модуль NIM-4T
4	Cisco Catalyst 9300 IE-9320-24P4S-E	4 GB оперативної пам’яті, 8 GB флеш-пам’яті	5	24 RJ-45, 4 SFP	Для підмереж
5	Meraki MS355-48X-HW	-	3	48 RJ-45, 2 QSFP+	Для підмереж

Продовження таблиці 2.1

Позиція	Найменування	Технічна характеристика	Кількість	Порти	Примітки
6	PC	AMD Ryzen 3 2100GE PRO, Інтегрована відеокарта, 16 GB оперативної пам'яті DDR4 - 2666 МГц, 512 GB (SSD)	204	1 RJ-45	Робочі станції
7	Canon i-SENSYS LBP710CX	Роздільна здатність: 9600x600 dpi, формат: A4, технологія друку: лазерна, кількість картреджів: 4	1	RJ-45, USB	Принтер
8	ATcom FTP cat.6 CCA 4 pairs 0.51 mm, internal, PVC	250Мг, 4 пари по 0.51 мм	610м	RJ-45	Для поєднання мережевих пристроїв
9	Smart Serial Back-to-Back Smart Serial DCE DTE Cable	0.75 м	3	DB-25	Для поєднання маршрутизаторів

Було сформовано загальну структуру обладнання мережі підприємства, згідно з якої для забезпечення надійності мережі, а також розподілення навантаження, було вирішено створити додаткові маршрути між маршрутизаторами, так було додано додаткові міжмережеві маршрутизатори. Мережа підприємства поділяється на внутрішню та зовнішню мережу. Поділевачем слугує маршрутизатор провайдера. До нього під'єднана внутрішня мережа, до якої відносяться LAN 2-5, а також WAN 1, WAN 3-8. Також під'єднана зовнішня мережа LAN 1.

Загальна структура обладнання мережі підприємства представлена на рисунку 2.2.

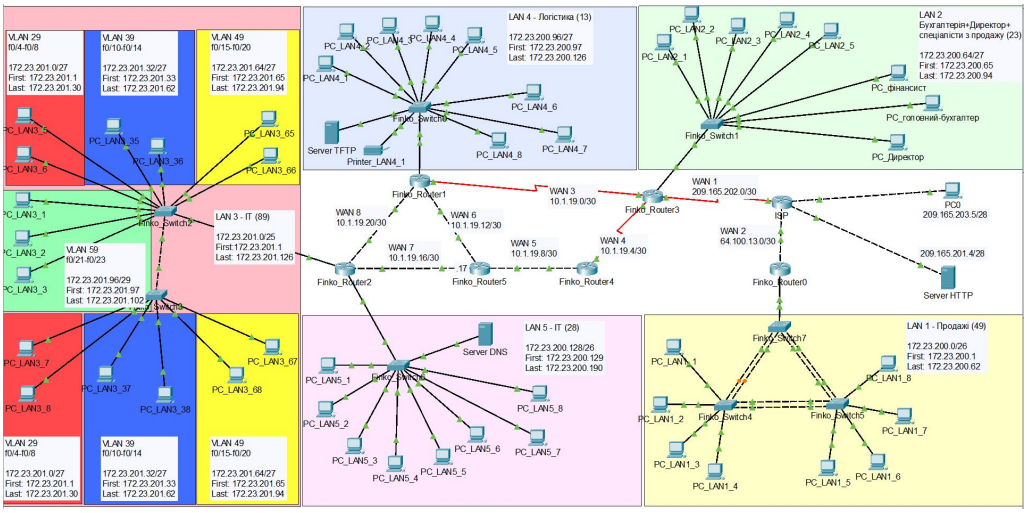

Рисунок 2.2 – Структурна схема обладнання мережі підприємства

2.2.3 Розрахунок інтенсивності трафіку вихідного трафіку найбільшої локальної мережі підприємства

Відповідно до технічних вимог, і структури підприємства компанії ТОВ «ABP Solutions», найбільшою локальною мережею має бути підмережа LAN 3, де на етапі розгортання має бути розміщено 89 ПК для частини IT відділу.

Для проведення розрахунків відомі наступні дані:

– середня довжина вихідного повідомлення в локальній мережі LAN 3 (l) дорівнює 650 байтам, або 5200 бітам;

– затримка передачі пакету в локальній мережі LAN 3 ≤6 мс;

– середня інтенсивність вихідного трафіку в локальній мережі LAN 3 (μ) = 108 кадрів/с;

– кількість вузлів в локальній мережі LAN 3 (N) = 89;

– кількість портів в комутаторів рівня доступу в локальній мережі LAN 3 (n) = 90;

– пропускна здатність ліній передачі даних в локальній мережі LAN 3 (C) = 100000000 біт/секунду.

Спершу була розрахована пропускна здатність локальної мережі на рівні доступу за формулою 2.1.

$$P_{\text{р.р}} = \mu * l * n, \quad (2.1)$$

де μ – середня інтенсивність вихідного трафіку в локальній мережі LAN 3;

l – середня довжина вихідного повідомлення в локальній мережі LAN 3;

n – Кількість портів в комутаторі рівня доступу в локальній мережі LAN 3

Тобто Рр.р = 108 * 5200 * 90 = 50544000 (біт/с) = (50.544 Мбіт/с)

Наступним кроком було розраховано значення інтенсивності виходу за формулою 2.2.

$$\mu_{\text{вих}} = C / l, \quad (2.2)$$

де C – пропускна здатність лінії передачі даних в локальній мережі LAN 3;

l – середня довжина вихідного повідомлення в локальній мережі LAN 3.

Тобто $\mu_{\text{вих}}$= 1000000000 біт/с / 5200 біт = 192307 (пакетів/с)

Далі було розраховано максимальну кількість кінцевих вузлів, яких локальна мережа LAN 3 зможе обслуговувати, враховуючи встановлені раніше комутатори до неї. Для цього розрахунку було використано формулу 2.3.

$$N = \mu_{\text{вих}} / \mu, \quad (2.3)$$

де $\mu_{\text{вих}}$ – значення інтенсивності виходу пакетів на секунду в локальній мережі LAN 3;

μ – середня інтенсивність вихідного трафіку в локальній мережі LAN 3.

Тобто N = 192307 / 108 ≈ 1780 одиниць.

Було розраховано інтенсивність вихідного трафіку для всіх вузлів у локальній мережі LAN 3 за формулою 2.4.

$$\lambda = N * \mu, \quad (2.4)$$

де N– кількість вузлів в мережі в локальній мережі LAN 3;

μ – середня інтенсивність вихідного трафіку в локальній мережі LAN 3.

Тобто λ = 89 * 108 = 9612 пакетів/секунду.

2.2.4 Розрахунок основних характеристик трафіку з метою підтвердження надійної роботи мережі

Для розрахування коефіцієнту затримки на рівні розподілу в локальній мережі LAN 3 було застосовано формулу 2.5.

$$\rho = \lambda / \mu_{вих}, \quad (2.5)$$

де λ– інтенсивність вихідного трафіку для всіх вузлів у локальній мережі LAN 3;

$\mu_{вих}$ – значення інтенсивності виходу пакетів на секунду в локальній мережі LAN 3.

Тобто ρ = 9612 / 192307 ≈ 0,04998.

Було розраховано коефіцієнт зайнятості комутаторів в локальній мережі LAN 3, для цього було застосовано формулу 2.6.

$$r = \rho / (1 - \rho), \quad (2.6)$$

де ρ – коефіцієнту затримки на рівні розподілу в локальній мережі LAN 3.

Тобто r = 0,04998 / (1 - 0,04998) ≈ 0.0526.

Було розраховано значення середньої затримки кадру в локальній мережі LAN 3, для цього було застосовано формулу 2.7.

$$T = 1 / (\mu_{вих} - \lambda), \quad (2.7)$$

де $\mu_{вих}$ – значення інтенсивності виходу пакетів на секунду в локальній мережі LAN 3;

λ– інтенсивність вихідного трафіку для всіх вузлів у локальній мережі LAN 3.

$T = 1 / (192307 – 9612) \approx 55 * 10^{-7}$ секунд

Використовуючи попередні розрахунки, було розраховано середню довжину черги в локальній мережі LAN 3. Для цього було використано формулу 2.8.

$$L_{черги} = \rho^2 / (1 - \rho), \quad (2.8)$$

де ρ – коефіцієнту затримки на рівні розподілу в локальній мережі LAN 3.

Тобто $L_{черги} = (0,04998)^2 / (1 - 0,04998) \approx 0.00263$.

Було розраховано середній час перебування пакета в черзі в локальній мережі LAN 3 за допомогою формули 2.9.

$$T_{очік} = L_{черги} / \lambda, \quad (2.9)$$

де $L_{черги}$ – середня довжина черги в локальній мережі LAN 3;

λ– інтенсивність вихідного трафіку для всіх вузлів у локальній мережі LAN 3.

Тобто $T_{очік} = 0.00263 / 9612 = 2.73616313*10^{-7}$ (секунди) ≈ 0.0007 мілісекунд

Було розраховано пропускну здатність каналу в локальній мережі LAN 3, для цього було застосовано формулу 2.10.

$$b = \lambda * l, \quad (2.10)$$

де λ– інтенсивність вихідного трафіку для всіх вузлів у локальній мережі LAN 3;

l – середня довжина вихідного повідомлення в локальній мережі LAN 3.

Тобто b = 9612 * 5200 = 49982400 біт/с ≈ 49.982 Мбіт/с.

В результаті проведення розрахунків було з’ясовано що усі значення розрахованих показників задовільняють технічні вимоги.

3 РОЗРОБКА КОРПОРАТИВНОЇ МЕРЕЖІ ПІДПРИЄМСТВА

3.1 Розрахунок налаштувань для заданої топології мережі

3.1.1 Розрахунок схеми адресації корпоративної мережі

Згідно з даними приведеними у попередніх розділах було визначено кількість вузлів в кожній з підсистем та розподілено їх відповідним чином у підмережі (табл. 3.1).

Таблиця 3.1 – Розподіл вузлів по підмережам комп'ютерної системи.

LAN 1	LAN 2	LAN 3	LAN 4	LAN 5
49	23	89	13	28

До маршрутизатора провайдера було додано HTTP сервер, до LAN 4 було додано TFTF сервер та принтер, і до LAN 5 було додано DNS сервер.

Базовою адресою для підмереж було обрано 172.23.200.0, з маскою /21, що у повному вигляді записується як 255.255.248.0. При подальшому розподілу адрес було враховано необхідну необхідну кількість вузлів у кожній підмережі, а також апаратні обмеження щодо кількості портів у мережевих пристроях, та кількість самих мережевих пристроїв. З уже відомими кількостями вузлів у кожній підмережі було розраховано та обрано необхідні маски підмереж.

Для підмережі LAN 1 була обрана маска /26, що записується як 255.255.255.192. З маскою /26 в рамках підмережі можна призначити 62 вузлів, щоб це визначити, спочатку було розраховано кількість вільних бітів (32 - 26 = 6, де 32 – це максимальна доступна кількість бітів), використовуючи результат попереднього розрахунку, було визначено кількість доступних вузлів у підмережі (2^6 - 2 = 62). При розрахунку кількості доступних вузлів у підмережі, після піднесення двійки до степеня, що було зроблено для визначення максимальної можливості комбінацій, було віднято з цього результату двійку для того щоб унеможливити для призначення використання першої і останньої адреси підмережі, адже вони є адресою самої мережі та

адреса широкомовна передачі в підмережі. Так, 62 адреси задовольняють необхідність у призначенні адрес 49 вузлам для LAN 1.

Для LAN 2 була обрана маска /27, що записується як 255.255.255.224. З маскою /26 в рамках підмережі можна призначити 30 вузлів. Розрахунки були проведені таким же чином як і у випадку з підмережею LAN 1. Так, 30 адрес задовольняють необхідність у призначенні адрес 23 вузлам для LAN 2.

Для LAN 3 була обрана маска /25, що записується як 255.255.255.128. З маскою /26 в рамках підмережі можна призначити 126 вузлів. Розрахунки були проведені таким же чином як і у випадку з підмережею LAN 1. Так, 126 адрес задовольняють необхідність у призначенні адрес 89 вузлам для LAN 3.

Для LAN 4 була обрана маска /27, що записується як 255.255.255.224. З маскою /26 в рамках підмережі можна призначити 30 вузлів. Розрахунки були проведені таким же чином як і у випадку з підмережею LAN 1. Так, 30 адрес задовольняють необхідність у призначенні адрес 13 вузлам для LAN 4.

Для LAN 5 була обрана маска /26, що записується як 255.255.255.192. З маскою /26 в рамках підмережі можна призначити 62 вузлів. Розрахунки були проведені таким же чином як і у випадку з підмережею LAN 1. Так, 62 адреси задовольняють необхідність у призначенні адрес 28 вузлам для LAN 5.

Відповідно до встановлених масок та базової адреси, було розраховано діапазони доступних адрес та адреси мереж. Для кожної підмережі, починаючи з базової адреси було додано доступну кількість вузлів, відповідно до обмежень накладених маскою підмережі. Тобто маска переводилась у двійкову систему числення і ті біти які мали значення 0, могли бути зміненими на адресу вузла. Відповідно, та частина яка є маскою в підмережі, але не є маскою в базовій мережі – може бути адресою підмережі.

Було вирішено розбити найбільшу підмережу LAN 3 на віртуальні підмережі: VLAN 29, VLAN 39, VLAN 49 та VLAN 59. Розподіл вузлів здійснювався по спеціалізаціям, і таким чином, у VLAN 29 було вирішено віднести 30 вузлів, у VLAN 39 було вирішено віднести 30 вузлів, у VLAN 49 було вирішено віднести 25 вузлів, та у у VLAN 59 було вирішено віднести 4

вузла. Віртуальним підмережам було призначено маски відповідно до розподілу адрес та маски підмережі в якій вони розміщені, а саме LAN 3.

Між внутрішньою мережею та маршрутизатором провайдера (WAN 1) було встановлено 209.165.202.0, як адресу мережі, та /30 у якості маски.

Між зовнішньою мережею та маршрутизатором провайдера (WAN 2) було встановлено 64.100.13.0, як адресу мережі, та /30 у якості маски.

Між іншими маршрутизаторами внутрішньої системи комп'ютерної мережі було обрано базу адресу та маску: 10.1.19.0/24.

Оскільки для поєднання 2 маршрутизаторів потрібно встановити лише 2 адреси на їх портах, то було вирішено для кожної мережі WAN (WAN 3-8) встановити /30 маскою підмережі. Адреси підмереж WAN були розраховані відповідно до відштовхуючись від базової адреси і наданих масок, додаючи до базової адреси максимальну кількість вузлів до кожної наступної підмережі WAN, але уникаючи адресу широкомовної передачі.

Результати усіх проведених розрахунків по підмережам представлені у таблиці 3.2.

Таблиця 3.2 – Схема адресації мережі

Назва мережі	Кількість вузлів	Адреса мережі	Маска мережі	Діапазон доступних адрес
LAN 1 (Продажі)	49	172.23.200.0	/26	172.23.200.1-172.23.200.62
LAN 2 (Бухгалтерія, Директор, продажі)	23	172.23.200.64	/27	172.23.200.65-172.23.200.94
LAN 4 (Логістика)	15	172.23.200.96	/27	172.23.200.97-172.23.200.126
LAN 5 (IT)	29	172.23.200.128	/26	172.23.200.129-172.23.200.190

Продовження таблиці 3.2

Назва мережі	Кількість вузлів	Адреса мережі	Маска мережі	Діапазон доступних адрес
LAN 3 IT	89	172.23.201.0	/25	172.23.201.1- 172.23.201.126
VLAN 29 Backend	30	172.23.201.0	/27	172.23.201.1- 172.23.201.30
VLAN 39 Frontend	30	172.23.201.32	/27	172.23.201.33- 172.23.201.62
VLAN 49 Тестувальники	25	172.23.201.64	/27	172.23.201.65- 172.23.201.94
VLAN 59 Адміністратори	4	172.23.201.96	/29	172.23.201.97- 172.23.201.102
WAN 1	2	209.165.202.0	/30	209.165.202.1- 209.165.202.2
WAN 2	2	64.100.13.0	/30	64.100.13.1-64.100.13.2
WAN 3	2	10.1.19.0	/30	10.1.19.1-10.1.19.2
WAN 4	2	10.1.19.4	/30	10.1.19.5-10.1.19.6
WAN 5	2	10.1.19.8	/30	10.1.19.9-10.1.19.10
WAN 6	2	10.1.19.12	/30	10.1.19.13-10.1.19.14
WAN 7	2	10.1.19.16	/30	10.1.19.17-10.1.19.18
WAN 8	2	10.1.19.20	/30	10.1.19.21-10.1.19.22

3.1.2 Розрахунок схеми адресації пристроїв у корпоративній мережі

Відповідно до розрахунків у попередньому підпункті, 3.1.1, та 2 розділу кваліфікаційної роботи, було надано адреси та маски всім пристроям мережі, на їх портах. Усі записи розподілення адрес між пристроями мережі приведені у таблиці 3.3.

Таблиця 3.3 – Схема адресації пристроїв мережі

Назва мережі	Пристрій	Інтерфейс	IP-адреса	Маска підмережі	Шлюз
LAN 1 Продажі	Finko_Router0	Gig8/0	172.23.200.1	/26	-
	Finko_Switch7	VLAN1	172.23.200.2	/26	172.23.200.1
	Finko_Switch5		172.23.200.3		
	Finko_Switch4		172.23.200.4		
	PC_LAN1_1-PC_LAN1_49	Fa0	172.23.200.14-172.23.200.62	/26	172.23.200.1
LAN 2 Бухгалтерія, Директор, продажі	Finko_Router3	Gig9/0	172.23.200.65	/27	-
	Finko_Switch1	VLAN1	172.23.200.66	/27	172.23.200.65
	PC_фінансист	Fa0	172.23.200.94	/27	172.23.200.65
	PC_головний-бухгалтер	Fa0	172.23.200.93	/27	172.23.200.65
	PC_Директор	Fa0	172.23.200.92	/27	172.23.200.65
	PC_LAN2_1-PC_LAN2_20	Fa0	172.23.200.72-172.23.200.91	/27	172.23.200.65
LAN 4 Логістика	Finko_Router1	Gig7/0	172.23.200.97	/27	-
	Finko_Switch0	VALN1	172.23.200.98	/27	172.23.200.97
	PC_LAN4_1-PC_LAN4_13	Fa0	172.23.200.114-172.23.200.126	/27	172.23.200.97
	Printer_LAN4_1	Fa0	172.23.200.113	/27	172.23.200.97
	Server TFTP	Fa0	172.23.200.112	/27	172.23.200.97

Продовження таблиці 3.3

Назва мережі	Пристрій	Інтерфейс	IP-адреса	Маска підме-режі	Шлюз
LAN 5 IT	Finko_Router2	Gig8/0	172.23.200.129	/26	-
	Finko_Switch6	VALN1	172.23.200.130	/26	172.23.200.129
	PC_LAN5_1-PC_LAN5_28	Fa0	172.23.200.163-172.23.200.190	/26	172.23.200.129
	Server DNS	Fa0	172.23.200.162	/26	172.23.200.129
LAN 3 IT	Finko_Router2	VLAN29	172.23.201.1	/27	-
		VLAN39	172.23.201.33	/27	-
		VLAN49	172.23.201.65	/27	-
		VLAN59	172.23.201.97	/29	-
	Finko_Switch2	VLAN 99	172.23.201.125	/29	172.23.201.121
	Finko_Switch3	VLAN 99	172.23.201.124	/29	172.23.201.121
VLAN 29	PC_LAN3_5-PC_LAN3_34	Fa0	172.23.201.1-172.23.201.30	/27	172.23.201.1
VLAN 39	PC_LAN3_35-PC_LAN3_64	Fa0	172.23.201.33-172.23.201.62	/27	172.23.201.33
VLAN 49	PC_LAN3_65-PC_LAN3_91	Fa0	172.23.201.65-172.23.201.94	/27	172.23.201.65
VLAN 59	PC_LAN3_1-PC_LAN3_3	Fa0	172.23.201.97-172.23.201.102	/27	172.23.201.97
WAN 1	Finko_Router3	Se6/0	209.165.202.2	/30	-
	ISP	Se6/0	209.165.202.1		
WAN 2	ISP	Gig9/0	64.100.13.1	/30	-
	Finko_Router0	Gig9/0	64.100.13.2		
WAN 3	Finko_Router3	Se7/0	10.1.19.1	/30	-
	Finko_Router1	Se6/0	10.1.19.2		
WAN 4	Finko_Router3	Se8/0	10.1.19.5	/30	-
	Finko_Router4	Se8/0	10.1.19.6		
WAN 5	Finko_Router4	Gig9/0	10.1.19.9	/30	-
	Finko_Router5	Gig7/0	10.1.19.10		

Продовження таблиці 3.3

Назва мережі	Пристрій	Інтерфейс	IP-адреса	Маска підме-режі	Шлюз
WAN 6	Finko_Router5	Gig8/0	10.1.19.13	/30	-
	Finko_Router1	Gig8/0	10.1.19.14		
WAN 7	Finko_Router5	Gig9/0	10.1.19.17	/30	-
	Finko_Router2	Gig6/0	10.1.19.18		
WAN 8	Finko_Router2	Gig7/0	10.1.19.22	/30	-
	Finko_Router1	Gig9/0	10.1.19.21		

3.2 Налаштування та перевірка роботи комп’ютерної системи

3.2.1 Базове налаштування конфігурації пристроїв

Для прикладу виконання команд для застосування базового налаштування мережевих пристроїв було взято Finko_Switch0, що був призначений для відділу логістики, LAN 4. При налаштуванні кожного мережевого пристрою було перейдено до режиму конфігурації. Це було зроблено наступними командами:

Switch>enable

Switch#configure terminal

Було призначено унікальні назви для всіх мережевих пристроїв, використовуючи наступний формат: “Finko_тип пристрою_номер”. Це має дозволити легко розрізняти пристрої за їх типом і адміністратором. З цією метою було застосовано наступну команду:

Switch(config)#hostname Finko_Switch0

Було забезпечено захист доступу до консолі і vty на кожному пристрої, встановивши до них пароль cisco. Це має запобігти несанкціонованому доступу до налаштувань пристроїв. Проводячи відповідні налаштування, було почергово під’єднано до консольної ліні, та після виконання необхідних команд,

і до ліній vty. Налаштування на мережевих пристроях булі виконані за наступним прикладом:

Finko_Switch0(config)#line console 0

Finko_Switch0(config-line)#password cisco

Finko_Switch0(config-line)#login

Finko_Switch0(config-line)#line vty 0 15

Finko_Switch0(config-line)#password cisco

Finko_Switch0(config-line)#login

Після завершення налаштувань на консольній лінії, та на лініях vty, було повернуто до режиму конфігурації, для цього було застосовано наступну команду:

Було задано пароль class для переходу в привілейований режим на кожному пристрої. Це дало можливість виконувати розширені команди для налаштування мережі. Встановлення відповідного налаштування було виконано з використанням наступної команди:

Finko_Switch0(config)#enable password class

Було зашифровано усі паролі, які зберігалися у відкритому вигляді, під час налаштування моделі комп’ютерної системи, для цього було використано наступну команду:

Finko_Switch0(config)#service password-encryption

Було розроблено банер MOTD для відображеня привітального повідомлення, яке містить інформацію про назву пристрою. Для встановлення банера на кожному пристрою було виконано наступну команду:

Finko_Switch0(config)#banner motd #Welcome message for the user in Finko_Switch0!#

Для забезпечення шифрування даних і аутентифікацію користувачів було налаштовано використання протоколу ssh для безпечної передачі даних по vty лініях на всіх пристроях, для цього було виконано наступні команди:

Finko_Switch0(config)#line vty 0 15

Finko_Switch0(config-line)#transport input ssh

Finko_Switch0(config-line)#login local

Після завершення налаштувань використання протоколу ssh для безпечної передачі даних по vty лініях, було повернуто до режиму конфігурації, для цього було застосовано наступну команду:

Було створено користувача для кожного пристрою, де “12320sk_Finko” – це ім’я користувача, а “admincisco” – пароль. Встановлення користувача було здійснено для кожного мережевого пристрою наступною командою:

Finko_Switch0(config)#username 12320sk1_Finko password admincisco

Для запобіганню конфліктам доменних імен, було встановлено ім’я домена для кожного пристрою, яке співпадає з його назвою, для цього було застосована наступна команда:

Finko_Switch0(config)#ip domain-name Finko_Switch0

Для покращення шифрування даних було згенеровано ключ RSA довжиною 1024 біт. Для створення ключа RSA було використано наступну команду на кожному мережевому пристрої:

Finko_Switch0(config)#crypto key generate rsa

Для синхронізації передачі даних між маршрутизаторами, було задано значення тактової частоти – 128000 на DCE-інтерфейсах маршрутизаторів. Налаштування були виконані з використанням типових команд, які мають наступний вигляд:

Finko_Router1(config)#interface Serial6/0

Finko_Router1(config-if)#clock rate 128000

Оскільки в подальшому на маршрутизаторах буде налаштовано DHCP, а серверам потрібно мати статичні адреси, то було заздалегідь налаштовано їх статичні адреси використовуючи інтерфейси операційних систем на серверах. Адреси були призначені відповідно до проведеним розрахункам у таблиці 3.3

3.2.2 Налаштування комутаторів корпоративної мережі

Для комутаторів мережі, для vlan1, було призначено адреси відповідно до таблиці 3.3, зі схемою адресації пристроїв мережі.

Для налаштування комутатора Finko_Switch0, з підмережі LAN 4, було використано наступні команди з режима конфігурації:

Finko_Switch0(config)#int vlan1

Finko_Switch0(config-if)#no shutdown

Finko_Switch0(config-if)#ip address 172.23.200.98 255.255.255.224

Finko_Switch0(config-if)#ip default-gateway 172.23.200.97

Таким же чином було призначено адреси і для інших комутаторів мережі, для vlan1.

3.2.3 Налаштування агрегації каналів

Відповідно до вимог, для об'єднання декількох фізичних каналів між комутаторами в одні логічний канали в LAN 1, було використано агрегацію каналів. Таким чином, використовуючи дану технологію, було збільшено пропускну здатність між комутаторами.

Для налаштування комутації в умовах агрегації каналів було використано протокол Port Aggregation Protocol (PAgP). PAgP використовується для автоматизованого об'єднання портів комутаторів Ethernet, відомого як EtherChannel. Так, з його допомогою можна забезпечити балансування навантаження даних/трафіку.

В підмережі LAN 1 всього 3 комутатора. Для кожного з них були встановлені налаштування, які забезпечують роботу за протоколом PAgP.

Так, для Finko_Switch7, для портів fastethernet0/1-2 був поставлений режим auto та налаштовані у першу групу, в той час як для fastethernet0/3-4 був поставлений режим auto та налаштовані у третю групу.

Для кожної встановленої на порти групи, було призначено режим роботи trunk, як і для групи портів, для яких були призначені вищевказані налаштування.

Налаштування для Finko_Switch5, для портів fastethernet0/1-2 був поставлений режим desirable та налаштовані у третю групу, в той час як для fastethernet0/3-4 був поставлений режим desirable та налаштовані у другу групу.

Для кожної встановленої на порти групи, було призначено режим роботи trunk, як і для групи портів, для яких були призначені вищевказані налаштування.

Налаштування для Finko_Switch4 було виконано аналогічним чином, проте були співставленні порти з встановленою групою, та був встановлений режим desirable, там, де на іншому кінці кабелю було встановлено режим auto.

Для кожної встановленої на порти групи, було призначено режим роботи trunk, як і для групи портів, для яких були призначені вищевказані налаштування.

3.2.4 Налаштування мереж VLAN

Згідно до вимог до корпоративної мережі, пункту 2 пояснювальної записки, було створено віртуальні підмережі, адресація до яких була наведена у таблиці 3.2. Також назви VLAN приведені у таблиці 3.4.

Таблиця 3.4 – Назви VLAN для комп’ютерної мережі компанії ТОВ «ABP Solutions»

Номер VLAN	Ім’я VLAN	Примітка
1	Default	Не використовується
29	Backend	Для потреб бекенд розробників
39	Frontent	Для потреб фронтенд розробників
49	QA	Тестувальники
59	Core	Менеджери відділу
99	Management	Для адміністраторських потреб
100	Native	Власна

При налаштуванні комутаторів, для інтерфейсів комутаторів, які використовуються для з’єднання між собою та для з’єднання з маршрутизаторами, було встановлено режим роботи trunk, для того щоб одним кабелем передавати пакети з усіх віртуальних підмереж одночасно. А для інтерфейсів які використовуються для з’єднання з вузлами, було встановлено режим access. Було створено віртуальні підмережі, відповідно до таблиці 3.4, на кожному з комутаторів в підмережі LAN 3.

Встановлюючи адреси на комутаторах для vlan99, була надана можливість адміністраторам віддалено підключатися до консолі кожного з комутаторів.

Для комутаторів Finko_Switch2 та Finko_Switch3 були створені віртуальні підмережі, та їм були надані назви, які перелічені в таблиці 3.4. Після створення віртуальних підмереж, та надання їм імен, було призначено для кожної з групи портів, відповідні віртуальні підмережі.

3.2.5 Налаштування маршрутизаторів корпоративної мережі

3.2.5.1 Налаштування адрес на портах маршрутизаторів

Були встановленні налаштування для всіх портів маршрутизаторів, де були вказані адреси підмереж. Адреси визначаються відповідно до таблиці 3.3. Для DCE кабелів було встановлено пропускну спроможність 128 Кб/с, та вартість метрики 7500. Також для vlan-ів було створено віртуальні під-інтерфейси, і окрім адрес, було вказано протокол інкапсуляції dot1Q.

3.2.5.2 Налаштування DHCP

На маршрутизаторах було налаштовано динамічну конфігурацію хостів (DHCP), так щоб автоматично призначалися IP-адреси, маски підмереж, шлюз та DNS сервер вузлам, які під’єднанні до однієї з ними підмережі.

DNS сервер було вказано для того, аби автоматично було надано адресу для всіх вузлів, щоб користувачі могли отримувати доступ до файлів

розповсюджуваних HTTP сервером, для зручності, використовуючи лише назву ресурсу, а не його адресу.

Приклад налаштування для маршрутизатора Finko_Router0 наведений нижче:

```
Finko_Router0(config)#ip dhcp pool LAN_1
Finko_Router0(dhcp-config)#network 172.23.200.0 255.255.255.192
Finko_Router0(dhcp-config)#default-router 172.23.200.1
Finko_Router0(dhcp-config)#dns-server 172.23.200.162
```

Для DHCP в LAN 1 була вказана інструкція щодо утримання від надання вузлам адрес 172.23.200.112, та 172.23.200.98, тому що їх вже займають VLAN 1 на комутаторах:

```
Finko_Router0(config)#ip dhcp excluded-address 172.23.200.2 172.23.200.4
```

Таким же чином були налаштовані і інші маршрутизатори. Для DHCP в LAN 4 була вказана інструкція щодо утримання від надання вузлам адрес 172.23.200.112, та 172.23.200.98, тому що їх вже займає сервер та VLAN 1 на комутатор, для LAN 2: 172.23.200.66, та адреси з проміжку з 172.23.200.92 по 172.23.200.95, тому що їх вже займають вузли зі статичними налаштуваннями та VLAN 1 на комутаторі.

Для маршрутизатора Finko_Router2 було вказано декілька DHCP-пулів, оскільки до нього під’єднанні одразу декілька підмереж з вузлами, та ще й в одній з підмереж була потреба у налаштуванні DHCP для віртуальних підмереж.

Для DHCP в LAN 5 була вказана інструкція щодо утримання від надання вузлам адрес 172.23.200.130, та 172.23.200.162, бо їх займає сервер та VLAN 1 на комутаторі.

В результаті виконаних налаштувань було автоматизоване та централізоване розподілення адресної конфігурації для вузлів мережі, що суттєво зменшило вірогідність помилки.

3.2.5.3 Налаштування маршрутизації у внутрішній мережі

Для налаштування маршрутизації у внутрішній мережі було вирішено використати протокол Open Shortest Path First (OSPF), тобто один з протоколів внутрішнього шлюзу (IGP), який допомагає знайти найкращий шлях маршрутизації між маршрутизатором-джерелом і маршрутизатором-приймачем, використовуючи власний алгоритм найкоротшого шляху (SPF). Це протокол внутрішнього шлюзу (IGP) для маршрутизації пакетів Інтернет-протоколу (IP) в межах одного домену маршрутизації, наприклад, автономної системи. Він збирає інформацію про стан з'єднання з доступних маршрутизаторів і будує топологічну карту мережі. OSPF дозволяє здійснювати автентифікацію маршрутизації за допомогою різних методів парольної автентифікації. OSPF дозволяє передавати і тегувати зовнішні маршрути, що вводяться в автономну систему.

Для зменшення навантаження на порти маршрутизаторів, які з’єднують його з підмережами, було оголошено що вони будуть пасивними. Тобто на цих портах не будуть надсилатися або оброблятися пакети Hello OSPF, які використовуються для виявлення сусідів та утримання сусідства. Таким чином, на цих портах не будуть формуватися сусідства OSPF з іншими маршрутизаторами. Однак, мережа, підключена до цього порту, все ще буде оголошена через інші активні інтерфейси OSPF. Також були оголошені, в налаштуванні маршрутизації через протокол OSPF, усі мережі підключені до маршрутизатора, але з інверсними масками. Приклад налаштування для маршрутизатора Finko_Router1 наведені нижче:

```
Finko_Router1(config)#router ospf 1
Finko_Router1(config-router)#passive-interface GigabitEthernet7/0
Finko_Router1(config-router)#network 172.23.200.96 0.0.0.31 area 0
Finko_Router1(config-router)#network 10.1.19.0 0.0.0.255 area 0
```

Таким чином було налаштовано маршрутизацію у внутрішній мережі, і завдяки цьому будь-які вузли підмереж внутрішньої мережі можуть надсилати пакети до будь-яких інших вузли підмереж внутрішньої мережі.

3.2.6 Налаштування роботи Інтернет

Для входу в Інтернет, на пограничному маршрутизаторі Finko_Router3, був налаштований динамічний NAT. Динамічний NAT забезпечує зіставлення приватних IP-адрес з публічною IP-адресою з пулу публічних IP-адрес, який називається пулом NAT. Тобто користуючись динамічним NAT, велика кількість вузлів, які не повинні бути доступні з Інтернету, зможуть використовувати малу кількість публічних IP-адрес для виходу в Інтернет. Динамічний NAT створює тимчасові записи в таблиці NAT і змінює їх залежно від доступності публічних IP-адрес у пулі.

Налаштування для маршрутизатора Finko_Router3 наведені нижче:

Finko_Router3(config)#ip access-list standard LANs

Finko_Router3(config-std-nacl)#permit 172.23.200.0 0.0.7.255

Finko_Router3(config)#ip nat pool Internet 209.165.202.5 209.165.202.30 netmask 255.255.255.0

Finko_Router3(config)#ip nat inside source list LANs pool Internet

Також було налаштовано напрям портів, до, та з внутрішньої мережі:

Finko_Router3(config)#interface Serial6/0

Finko_Router3(config-if)#ip nat outside

Finko_Router3(config-if)#interface Serial7/0

Finko_Router3(config-if)#ip nat inside

Finko_Router3(config-if)#interface Serial8/0

Finko_Router3(config-if)#ip nat inside

Finko_Router3(config-if)#interface GigabitEthernet9/0

Finko_Router3(config-if)#ip nat inside

Використовуючи такий підхід, було заощаджено публічні IP-адреси для використання великою кількістю вузлів компанії. Також була поліпшена безпека внутрішньої мережі, постійно змінюючи IP-адреси.

Коли динамічний NAT був успішно налаштований, потрібно налаштувати яким чином пакет з Finko_Router3 будуть передаватися на маршрутизатор провайдера. З цією ціллю було вирішено налаштувати також і

статичний NAT на Finko_Router3. Статичний NAT створює постійний запис у таблиці NAT і не змінюється, якщо його не налаштувати вручну.

Налаштування статичного NAT для маршрутизатора Finko_Router3 наведені нижче:

Finko_Router3(config)#ip route 0.0.0.0 0.0.0.0 209.165.202.1

Таблиця маршрутизації на налаштованому маршрутизаторі Finko_Router3 представлена на рисунку 3.1.

Routing Table for Finko_Router3

Type	Network	Port	Next Hop IP	Metric
S	0.0.0.0/0	---	209.165.202.1	1/0
C	10.1.19.0/30	Serial7/0	---	0/0
C	10.1.19.4/30	Serial8/0	---	0/0
O	10.1.19.8/30	Serial8/0	10.1.19.6	110/7501
O	10.1.19.12/30	Serial7/0	10.1.19.2	110/7501
O	10.1.19.16/30	Serial8/0	10.1.19.6	110/7502
O	10.1.19.16/30	Serial7/0	10.1.19.2	110/7502
O	10.1.19.20/30	Serial7/0	10.1.19.2	110/7501
C	172.23.200.64/27	GigabitEthernet9/0	---	0/0
O	172.23.200.96/27	Serial7/0	10.1.19.2	110/7501
O	172.23.200.128/26	Serial7/0	10.1.19.2	110/7502
O	172.23.201.0/27	Serial7/0	10.1.19.2	110/7502
O	172.23.201.32/27	Serial7/0	10.1.19.2	110/7502
O	172.23.201.64/27	Serial7/0	10.1.19.2	110/7502
O	172.23.201.96/29	Serial7/0	10.1.19.2	110/7502
C	209.165.202.0/30	Serial6/0	---	0/0

Рисунок 3.1– Таблиця маршрутизації на Finko_Router3

Таким же чином було налаштовано статичну маршрутизацію для виходу в Інтернет на всіх маршрутизаторах внутрішньої мережі, але посилаючись на наступний маршрут до пограничного маршрутизатора. Також було налаштовано напрям портів, до (inside), та з (outside) внутрішньої мережі на

кожному з маршрутизаторів внутрішньої мережі. Приклад налаштування для маршрутизатора Finko_Router1 наведені нижче:

```
Finko_Router1(config)#ip route 0.0.0.0 0.0.0.0 10.1.19.1
Finko_Router1(config)#interface Serial6/0
Finko_Router1(config-if)#ip nat outside
Finko_Router1(config-if)#interface GigabitEthernet7/0
Finko_Router1(config-if)#ip nat inside
Finko_Router1(config-if)#interface GigabitEthernet8/0
Finko_Router1(config-if)#ip nat inside
Finko_Router1(config-if)#interface GigabitEthernet9/0
Finko_Router1(config-if)#ip nat inside
```

Таким же чином були налаштовані і інші маршрутизатори внутрішньої мережі. А на маршрутизаторі провайдера було налаштовано зворотній шлях:

```
Finko_ISP(config)#ip route 209.165.202.0 255.255.255.0 209.165.202.2
```

Таблиця маршрутизації на налаштованому маршрутизаторі ISP представлена на рисунку 3.2.

Routing Table for IPS

Type	Network	Port	Next Hop IP	Metric
C	64.100.13.0/30	GigabitEthernet9/0	---	0/0
C	209.165.201.0/28	GigabitEthernet7/0	---	0/0
S	209.165.202.0/24	---	209.165.202.2	1/0
C	209.165.202.0/30	Serial6/0	---	0/0
C	209.165.203.0/28	GigabitEthernet8/0	---	0/0

Рисунок 3.2 – Таблиця маршрутизації на ISP

Для віддаленої мережі LAN 1 було налаштовано PAT для виходу в Інтернет. PAT є типом динамічного NAT, який дозволяє декільком пристроям в одній приватній мережі підключатися до загальнодоступного Інтернету,

використовуючи одну і ту ж публічну IP-адресу. PAT додає номер порту в кінець IP-адреси, щоб розрізняти різні пристрої та сеанси.

Налаштування PAT, та напряму портів, до, та з LAN 1 до ISP, для маршрутизатора Finko_Router0 наведені нижче:

Finko_Router0(config)#ip access-list standard LAN1_list

Finko_Router0(config-std-nacl)#permit 172.23.200.0 0.0.0.63

Finko_Router0(config)#ip nat inside source list LAN1_list interface GigabitEthernet9/0 overload

Finko_Router0(config)#ip route 0.0.0.0 0.0.0.0 64.100.13.1

Finko_Router0(config)#interface GigabitEthernet8/0

Finko_Router0(config-if)#ip nat inside

Finko_Router0(config-if)#interface GigabitEthernet9/0

Finko_Router0(config-if)#ip nat outside

Таблиця маршрутизації на налаштованому маршрутизаторі Finko_Router0 представлена на рисунку 3.2.

Routing Table for Finko_Router0

Type	Network	Port	Next Hop IP	Metric
S	0.0.0.0/0	---	64.100.13.1	1/0
C	64.100.13.0/30	GigabitEthernet9/0	---	0/0
C	172.23.200.0/26	GigabitEthernet8/0	---	0/0

Рисунок 3.2 – Таблиця маршрутизації на Finko_Router0

Було налаштовано HTTP сервер, так щоб за його адресою користувачу відкривався веб-сайт з відомостями про тему та завдання на кваліфікаційну роботу. Також, у внутрішній мережі було налаштовано DNS сервер, який дозволяє користувачам внутрішньої мережі, замість адреси HTTP серверу, вводити текстову адресу, яка за його допомогою переводиться у IP адресу HTTP серверу. Вміст файлу index.html наведений у додатку А.

3.2.7 Перевірка роботи комп’ютерної системи

Схема моделі комп’ютерної системи представлена у розділі 2, як рисунок 2.2.

Було перевірено роботу налаштованого сервісу DHCP на маршрутизаторах. Приклад розподілу адрес за допомогою DHCP представлений на рисунку 3.3.

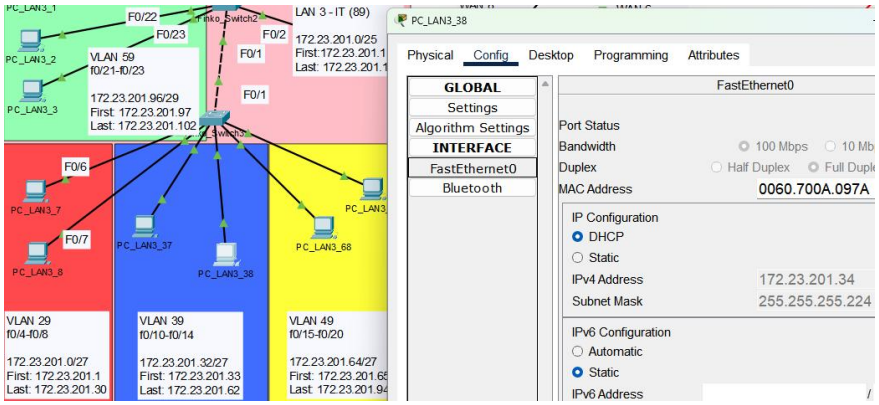

Рисунок 3.3 – Встановлення динамічного розподілення IP-конфігурації для вузла у VLAN 39

Отже, на рисунку вище видно що DHCP для віртуальної підмережі, VLAN 39, налаштовано коректно.

Було перевірено налаштування маршрутизації у внутрішній мережі шляхом виконання команди “ping” з LAN 3 до вузла внутрішньої підмережі LAN 2. Результат виконання команди наведено на рисунку 3.4.

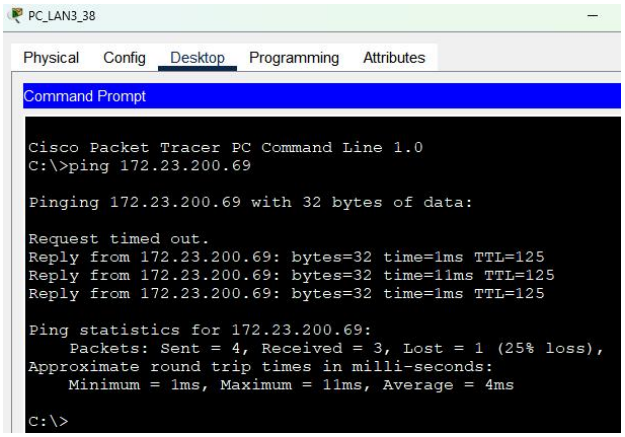

Рисунок 3.4 – Виконання команди ping з вузла LAN 3 до вузла у LAN 2

Було перевірено налаштування IP конфігурації у зовнішній мережі LAN 1, шляхом виконання команди “ping” з одного вузла LAN 1 до іншого. Результат виконання команди наведено на рисунку 3.5.

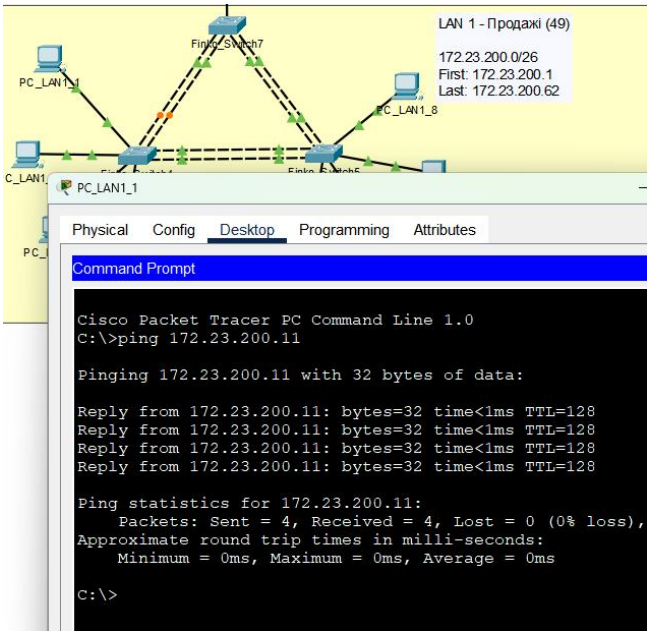

Рисунок 3.5 – Виконання команди ping між вузлами в зовнішній мережі LAN 1

Було перевірено налаштування маршрутизації у зовнішній мережі шляхом виконання команди “ping” з LAN 1 до HTTP сервера, під'єднаного до ISP. Результат виконання команди наведено на рисунку 3.6.

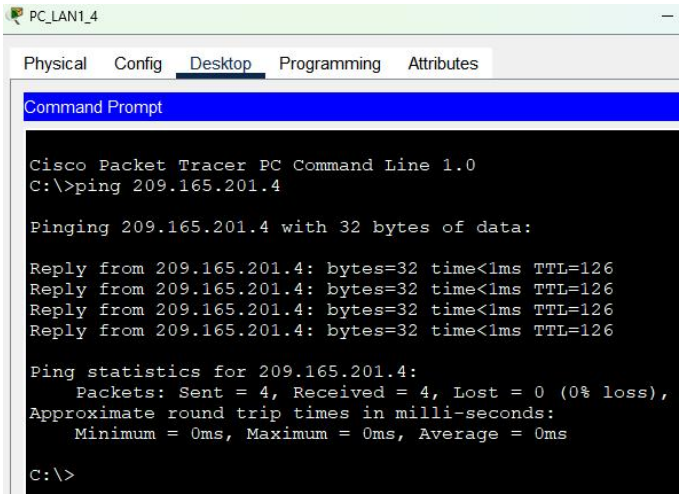

Рисунок 3.6 – Виконання команди ping з вузла LAN 1 до HTTP сервера

Було перевірено роботу DNS сервера у зв’язці з HTTP сервером, виконуючи HTTP запит до головної веб-сторінки, використовуючи текстову адресу до HTTP сервера, з вузла у внутрішній мережі. Результат виконання HTTP запиту наведено на рисунку 3.4.

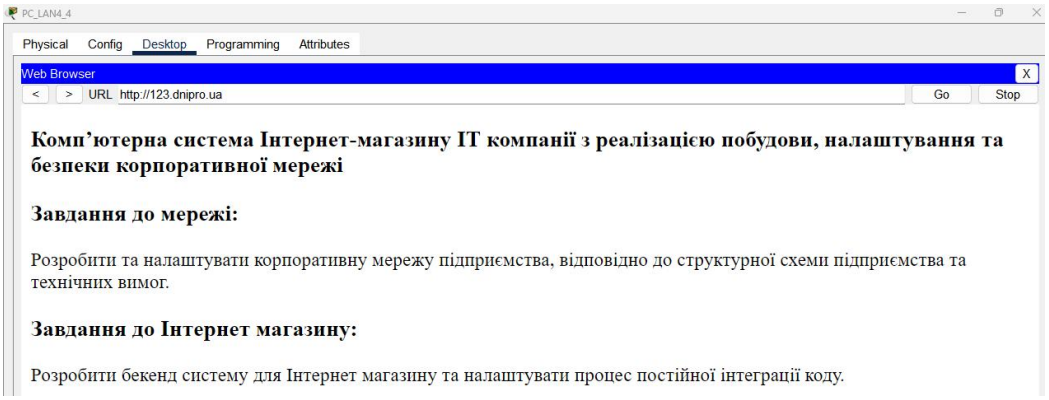

Рисунок 3.7 – Виконання HTTP запиту до HTTP серверу, використовуючи текстову адресу з вузла у LAN 4

Було перевірено заміну IP-адрес, з приватної до публічної, при надсиланні пакету з внутрішньої мережі до зовнішньої через ISP. Перевірка пакету, при його надсиланні до Інтернету, у симуляції за мережевою моделлю OSI представлена на рисунку 3.8.

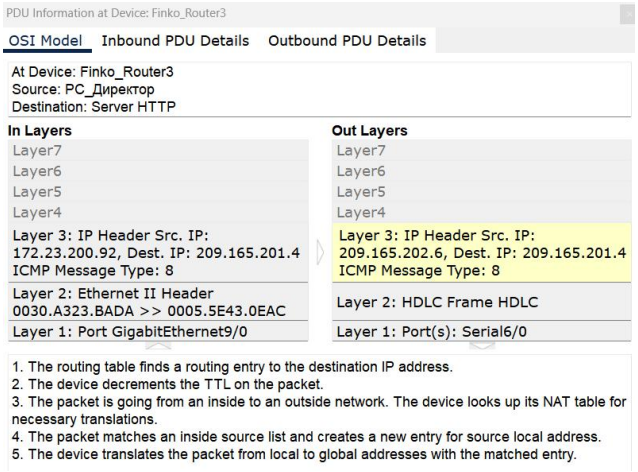

Рисунок 3.8 – Перевірка пакету, при його надсиланні до Інтернету, у симуляції за мережевою моделлю OSI

Було перевірено заміну IP-адрес, з публічної до приватної, при надсиланні пакету з Інтернету до внутрішньої мережі. Перевірка пакету, при його надсиланні до Інтернету, у симуляції за мережевою моделлю OSI представлена на рисунку 3.9.

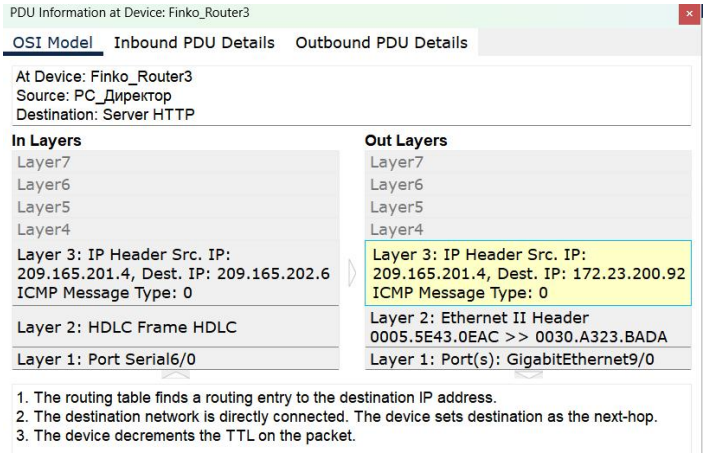

Рисунок 3.9 – Перевірка пакету, при його надсиланні з Інтернету, у симуляції за мережевою моделлю OSI

4 РОЗРОБКА КОМПОНЕНТА СИСТЕМИ

4.1 Налаштування контейнерізації

Було створено docker-compose.yml файл в директорії проекту. Призначенням цього файлу є розгортання, поєднування та налаштовування кілька контейнерів Docker одночасно. У цьому файлі одразу було вказано версію Docker Compose, та оголошено блок сервісів, директорій, та мереж. В блоці мереж було вказана назву мережі для додатку, “e-shop-network”. В блоці томів було вказано наступні три томи: “postgres_data”, “coverage”, та “static”. На рисунку 4.1 представлено схему поєднання контейнерів Docker, томів та мереж, за допомогою Docker Compose для бекенд частини Інтернет-магазину.

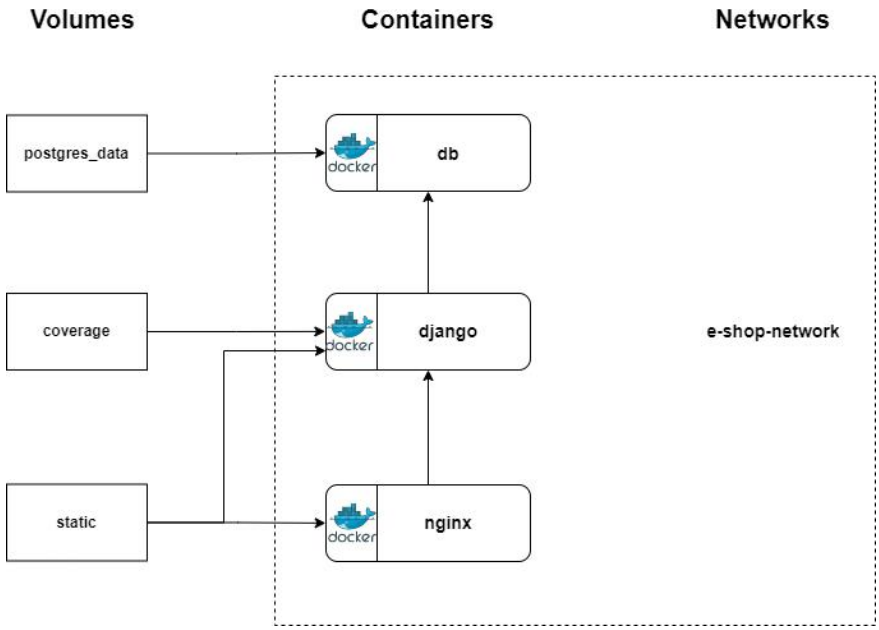

Рисунок 4.1 – Схема поєднання томів, контейнерів та мереж Docker

4.1.1 Налаштування контейнера для бази даних

Першим, було створено контейнер для бази даних. В якості конфігураційних даних для контейнера бази даних було вказано назву використовуваного загальнодоступного зображення, а саме: postgres; назву контейнера, змінні середовища, які були вказані таким чином, що значення змінних не було видно у файлі, а будуть імпортуватися з файлу для зберігання

змінних оточення, “.env”. Також було вказано: порти та том “postgres_data”, примонтований до Docker, аби зберігати базу даних навіть після перезапуску контейнера. Останнє що було вказано для контейнера бази даних – це мережа для додатку, “e-shop-network”.

4.1.2 Налаштування контейнера для веб-додатка

Було створено контейнер для веб-додатку, “django”, який будує образ контейнера з поточної директорії, використовуючи файл Dockerfile, який знаходиться в ./dockerfiles/django/Dockerfile. Було передано аргумент DEV у файл Dockerfile, для того щоб в залежності від того, чи має ця змінна значення “true” або “false”, відповідним чином налаштовувати образ, завантажуючи необхідні пакети для різного режиму роботи з веб-додатком.

Усі змінні оточення, які були передані до конфігураційного файлу були передані до нього через файл для зберігання змінних оточення, “.env”, тобто значення не були вписані у конфігураційний файл на пряму. Був відкрити порт 8000 на хості, та був прив’язаний до порту 8000 у контейнері.

В якості вхідної точки для запуску контейнера веб-додатку, було написано sh скрипт, який виконує наступні дії: переходить до директорії “backend”, чекає, поки база даних буде готова, застосовує міграції, збирає статичні файли та запускає сервер gunicorn для веб-додатка.

Було змонтовано наступні томи:

- том “./backend” до шляху /home/backend-user/app/backend у контейнері;
- том “coverage” до шляху /home/backend-user/app/coverage у контейнері;
- том “static” до шляху /home/backend-user/app/static у контейнері.

Було вказано, що сервіс “django” залежить від сервісу “db” і запускається після нього. Також було приєднано сервіс “django” до мережі “e-shop-network” і додано йому псевдонім “django”.

4.1.3 Налаштування контейнера для веб-сервера

Було створено сервіс “nginx”, який виконує роль веб-сервера для веб-додатку. Для цього було використовуємо готовий образ “nginx:1.23.3-alpine-slim” з Docker Hub[38], який базується на легкій версії Alpine Linux.

Було змонтовано том ./dockerfiles/nginx/default.conf до шляху /etc/nginx/conf.d/default.conf у контейнері. Цей файл містить конфігурацію Nginx, яка складається з двох частин:

- “upstream django-app”, який визначає групу серверів, до яких nginx буде перенаправляти запити. Тобто один сервер “django”, який працює на порту 8000 у контейнері “django”;
- “server”, який визначає параметри веб-сервера Nginx. Він слухає порт 80 на хості, та має два розділи location: “location /”, який перенаправляє всі запити до upstream django-app, та “location /static/”, який обслуговує статичні файли з директорії /home/backend-user/app/static/.

Було змонтовано том “static” до шляху /home/backend-user/app/static/ у контейнері. Це дозволило зберегти статичні файли поза контейнером і спільно використовувати їх між сервісами “nginx” і “django”.

Було відкрито порт 80 на хості та прив’язано його до порту 80 у контейнері. Це дозволило звертатися до веб-додатку за адресою “http://localhost:80” або просто “http://localhost”.

Сервіс “nginx” був приєднаний до мережі “e-shop-network”, яка була створена у файлі docker-compose.yml. Це дозволяє забезпечити зв’язок між сервісами “nginx” і “django” за допомогою їх псевдонімів.

Встановлено залежність сервіса “nginx” від сервісу “django” для того, щоб він запускався після нього, для того щоб веб-додаток був готовий приймати запити від nginx.

Після успішного налаштувананя контейнерів Docker, після виконання команди “docker-compose --env-file .dev.env up --build”, виконується запуск контейнерів, з їх попередньою побудовою. За допомогою ключа “--env-file .dev.env” було передано значення змінних оточення, які були попередньо

записані в цей файл. Ці змінни оточення необхідні для того щоб коректно налаштовувались контейнери та веб-додаток, відповідно до поточних потреб. Для наведення прикладу нижче, команда для запуску та побудови контейнерів Docker була виконана з імпортуванням змінних оточення з відповідного файлу для налаштування розробницького середовища, проте для побудови контейнерів для розгортання сервісів для кінцевого споживача варто встановлювати інші налаштування, забезпечуючи більший рівень захисту та оптимізуючи ресурси, шляхом зменшення кількості завантажених пакетів, у яких немає потреби для кінцевого користувача. Також змінні середовища разом з їх ключами, які будуть використовуватися для розгортання проекту для використання кінцевим споживачем, варто зберігати в секреті, та не викладати у відкритий доступ.

На рисунку 4.2 зображено повідомлення в консолі щодо проходження процесу побудови контейнерів, після чого вони поєднуються, та починають свій запуск з бази даних.

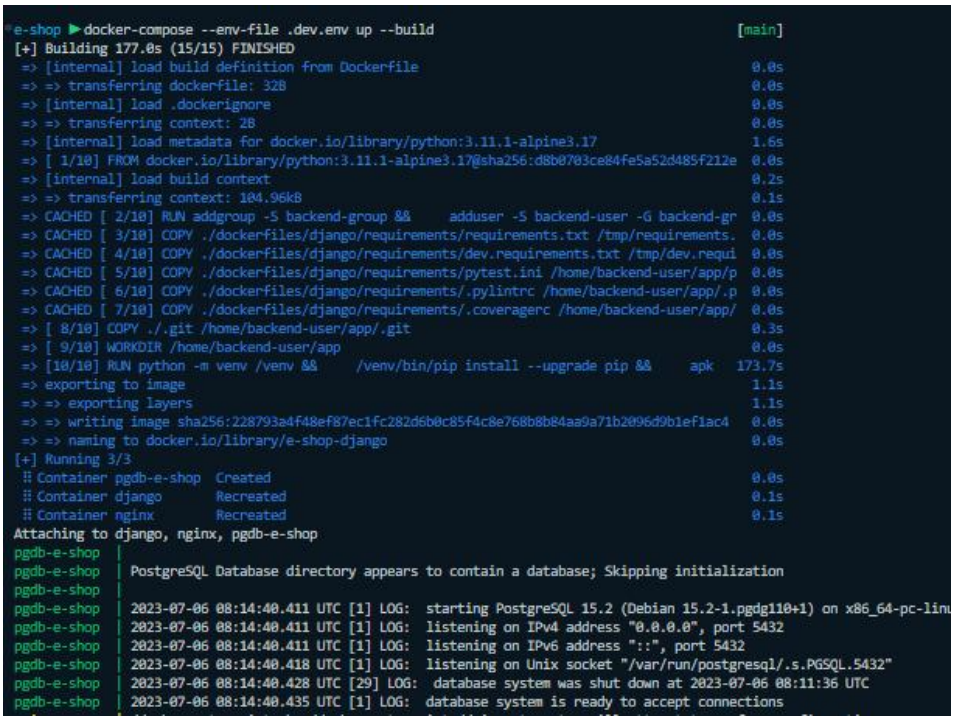

Рисунок 4.2 – Процес побудови контейнерів Docker, та запуск першого з них

На рисунку 4.3 зображено повідомлення в консолі щодо запуску контейнерів веб-серверу та веб-додатку.

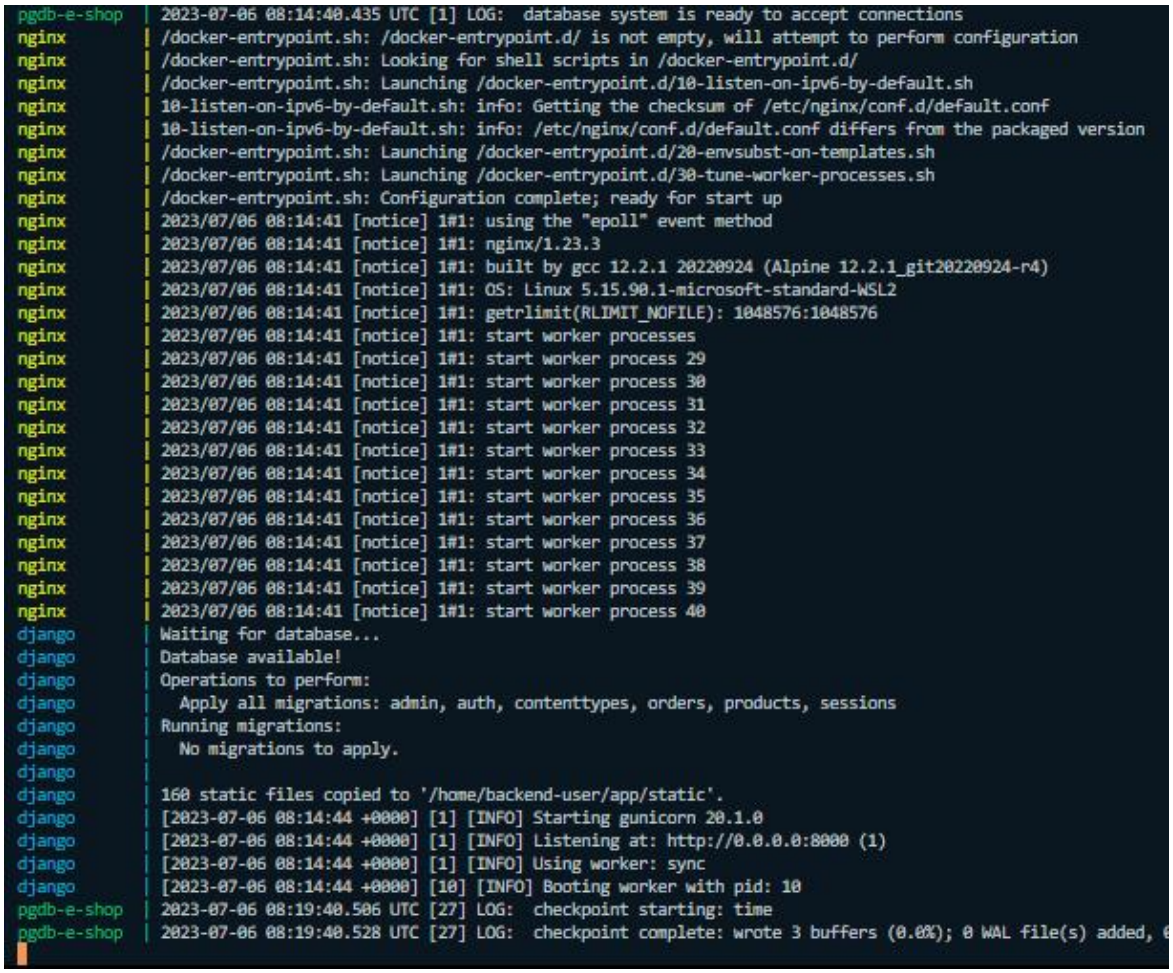

Рисунок 4.3 – Процес запуску контейнерів Docker веб-серверу та веб-додатку

4.2 Налаштування системи автоматичної перевірки коду на якість

Було створено “checks.yml” файл у директорії “.github/workflows”. Цей файл відповідає за налаштування GitHub Actions, або те для чого воно було використано в цьому додатку, для постійних інтеграцій (CI).

У створеному файлі було визначено назву робочого процесу “test & lint”, який буде виконувати тестування та перевірку коду. Було визначено які події будуть запускати робочий процес, а саме: події push та pull_request до гілки main в репозиторії проекту в GitHub [6].

Було оголошено блок завдань, які будуть виконуватися в рамках робочого процесу. В блоку завдань було визначено завдання “test-lint”, яке буде мати назву “Test and Lint” і буде виконуватися на середовищі “ubuntu-20.04”. Завдання “test-lint” буде відповідальне за тестування та перевірку коду додатку, проводячи ці перевірки за допомогою Docker.

Було визначено кроки, які будуть виконуватися в рамках завдання test-lint:

– крок “Login to Docker Hub”, який буде використовувати дію “docker/login-action@v1” для аутентифікації у Docker Hub [38]. Для того, щоб мати доступ до образів Docker Hub [38], потрібно увійти за допомогою імені користувача та пароля. Тому, було передано параметри username та password, які беруться з змінних середовища репозиторію GitHub [6];

– крок “Checkout”, який буде використовувати дію “actions/checkout@v2” для отримання коду з репозиторію GitHub. Це дозволить побудувати образ Docker з файлу Dockerfile, який знаходиться у репозиторії [6].

На рисунку 4.4 представлено виконання описаних вище кроків для завдання “test-lint” у Github Actions.

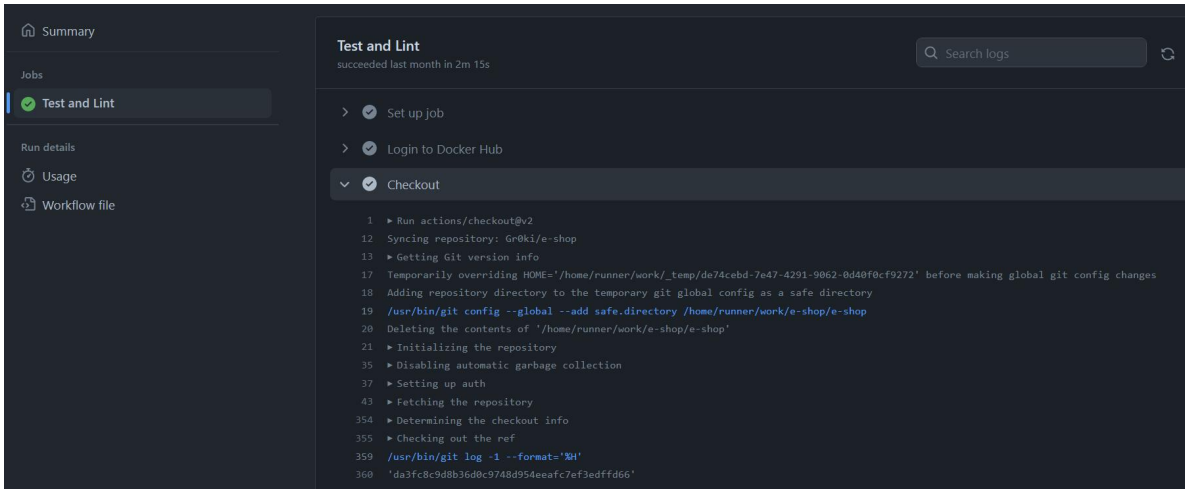

Рисунок 4.4 – Виконання входження в Docker Hub та отримання коду з репозиторію GitHub у GitHub Actions

4.2.1 Налаштування перевірки коду на відповідність стандартам

Було визначено крок “Lint”, який буде виконувати перевірку коду на стиль та якість за допомогою інструменту статичного аналізу коду Python, “pylint”. В цьому кроці було виконано команду, яка буде використовувати docker-compose для запуску сервісу “django” в новому контейнері і видаляємо його після завершення команди, також було імпортовано до запускаємого середовища змінні оточення з файлу “.dev.env”. Сервіс “django” буде виконувати команду “sh -c”. Команда “sh -c” дозволяє виконувати рядок команд у оболонці sh. В контейнері “django” було перейдено до директорії “backend”, де зберігається весь код веб-додатку, та виконано перевірку коду у директорії “src” за допомогою команди pylint, використовуючи комбінацію ключей до неї, які забезпечують:

- імпортування правил та налаштувань для перевірки коду;
- вмикання плагіну pylint_django, який допомагає pylint розуміти специфіку Django, таку як: моделі, форми, views тощо;
- передачу шляху до модуля налаштувань для Django, для того щоб раніше підключений плагін працював коректно. На рисунку 4.5 було представлено результат виконання перевірки коду на відповідність стандартів за допомогою “pylint”.

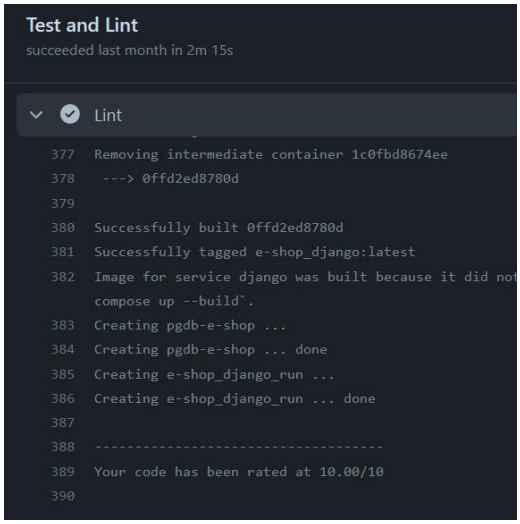

Рисунок 4.5 – Результат виконання перевірки коду на відповідність стандартів за допомогою “pylint” через GitHub Actions

4.2.2 Налаштування автоматичного тестування коду з фіксуванням його покриття тестами

Було створено крок наступний крок завдання “test-lint”, “Run tests and measure code coverage”, який виконує команди з Docker контейнера “django”, а саме:

- контейнер “django” очікує на базу даних;
- виконуються усі тести з каталогу “./backend”, та розраховується ряд показників щодо покриття коду тестами, які зберігаються у примонтованому до контейнера томі “coverage”.

На рисунку 4.6 наведено результат виконання кроку “Run tests and measure code coverage” завдання “test-lint”.

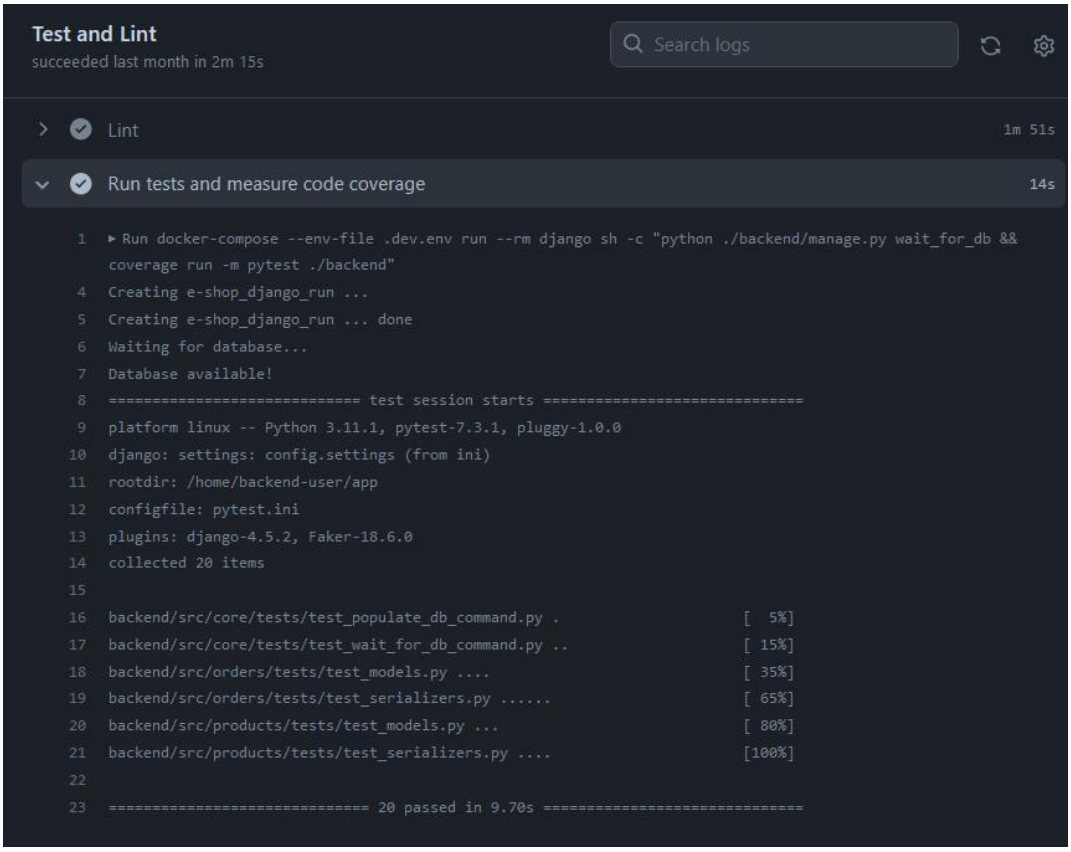

Рисунок 4.6 – Результат виконання кроку “Run tests and measure code coverage” завдання “test-lint” через GitHub Actions

4.2.3 Налаштування звітності з результатом проходження перевірок коду на якість

Було створено крок “Publish coverage results via coveralls.io”, який має виконувати публікацію результатів покриття коду за допомогою сервісу coveralls.io [25]. У цьому кроці виконуються команда з додавання змінної середовища “COVERALLS_REPO_TOKEN” до файлу .dev.env. Змінна “COVERALLS_REPO_TOKEN” містить токен для аутентифікації репозиторію на coveralls.io [25]. Токен береться з змінних середовища налаштувань GitHub.

Також з Docker контейнера “django” виконується команда “coveralls”, за допомогою якої буде надіслано дані про покриття коду на coveralls.io.

Для коректної роботи команди “coveralls”, в конфігураційний файл, для сервісу “django”, “Dockerfile” було додано інструкцію щодо копіювання з хоста каталогу “.git” до контейнеру.

Також, було завчасно проведено авторизацію на сервіс coveralls.io, за допомогою того ж GitHub акаунта, на якому і розміщений репозиторій проекту [6]. Після проведеної авторизації, на сервісі coveralls.io було підключено проект з GitHub репозиторію [6] до сервісу, завдяки чому було отримано токен для налаштування зв’язку, який був доданий в якості змінної середовища в налаштуваннях GitHub. На рисунку 4.7 представлено результат виконання кроку “Publish coverage results via coveralls.io.” завдання “test-lint”.

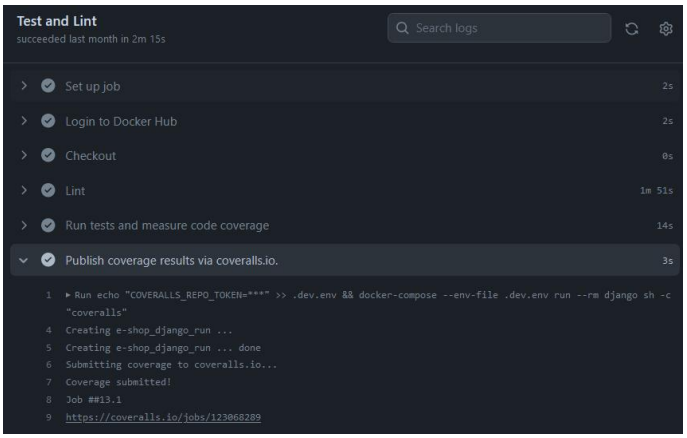

Рисунок 4.7 – Результат виконання кроку “Publish coverage results via coveralls.io.” завдання “test-lint” через GitHub Actions

В разі успішного проходження усіх перевірок, на головній сторінці репозиторію GitHub [6], з’являється зелена галочка, яка сповіщає про успішне проходження усіх перевірок.

На рисунку 4.8 представлено сторінку репозиторію GitHub [6] з виконаною успішною перевіркою коду.

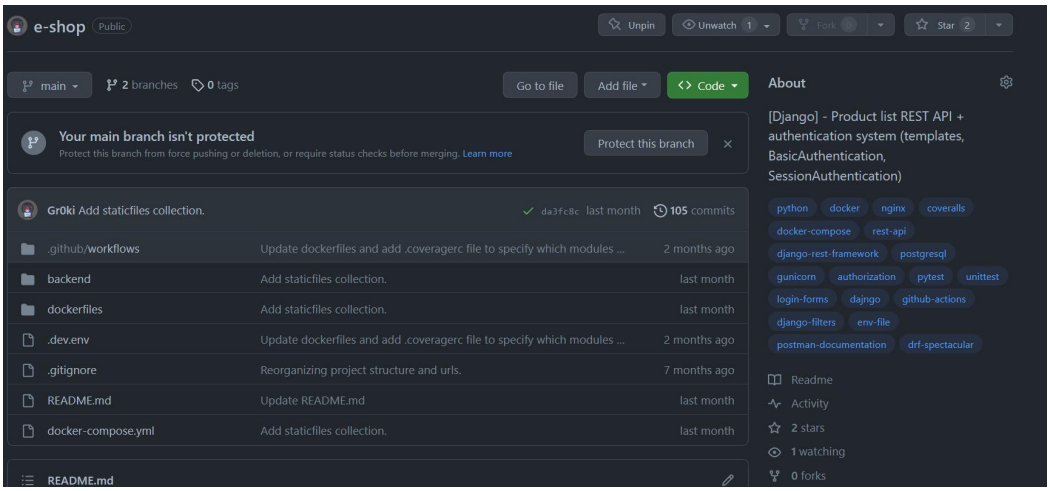

Рисунок 4.8 – Сторінка репозиторію GitHub з виконаною успішною перевіркою коду

4.3 Налаштування системи автентифікації для веб-додатку

Після створення додатку Django, і налаштування структури проекту, було використано вбудовану в Django систему автентифікації, до якої було створено власні форми входу, виходу з акаунту користувача, реєстрації, та зміни паролю.

До форм входу та виходу з акаунту користувача було додано налаштування до проходженням користувачем reCaptcha v3 [39], від компанії Google. Якщо користувач її проходить при виконанні переходу на наступну сторінку, веб-додаток дозволяє йому це зробити, в іншому випадку, користувач лише побачить повідомлення про помилку з відповідним текстом.

До всіх форм входу додані повідомлення про помилки які можуть виникнути в ході взаємодії користувача з цими формами. Вони відображаються лише тоді коли результат дії користувача приводить до відповідної помилки.

На рисунку 4.9 приведено сторінку входу.

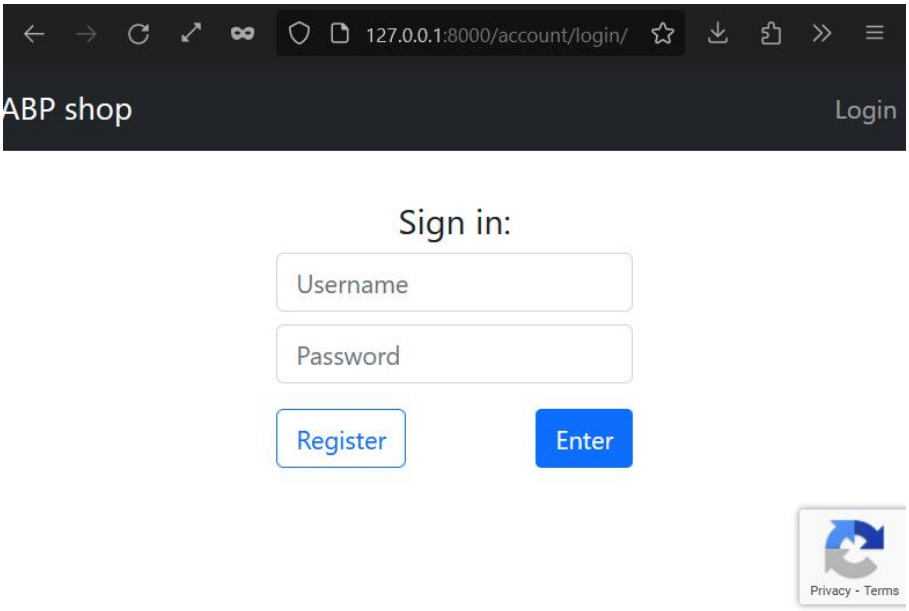

Рисунок 4.9 – Сторінка входу

Якщо вхід здійснює адміністратор, то на сторінці акаунту, окрім стандартних опцій йому буде ще доступна адмін-панель, та сторінка документації до API. На рисунку 4.10 приведено сторінку адміністратора.

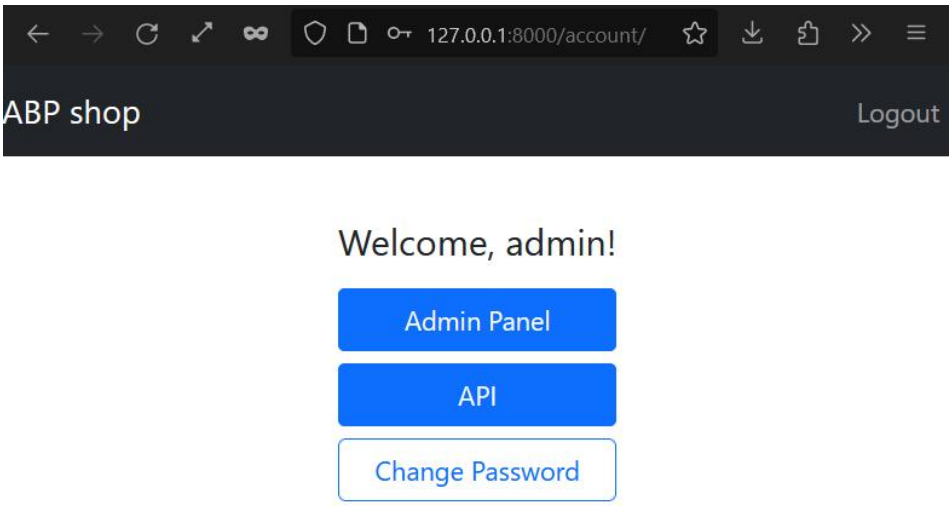

Рисунок 4.10 – Сторінка адміністратора

На рисунку 4.11 приведено сторінку звичайного користувача.

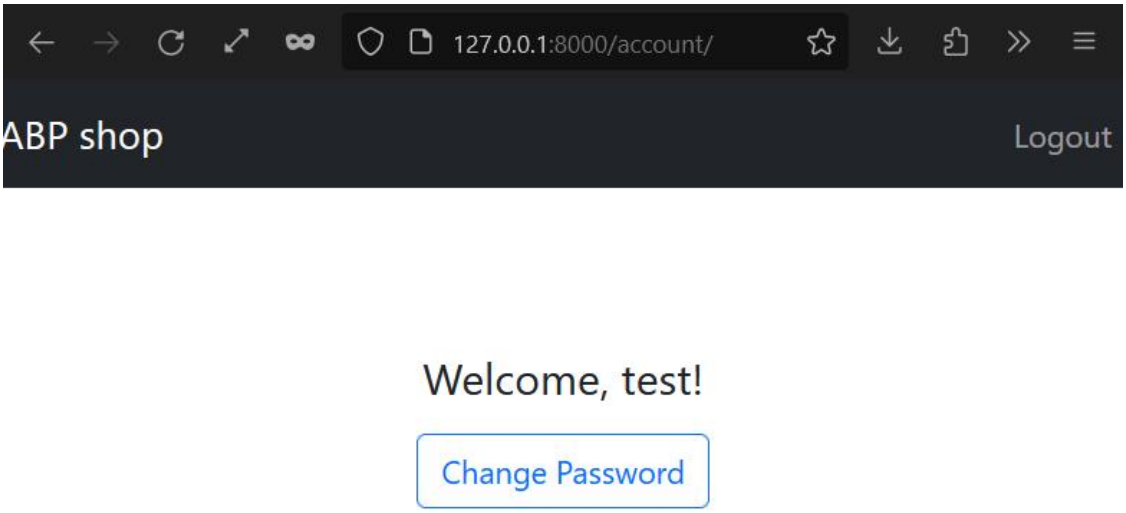

Рисунок 4.11 – Сторінка звичайного користувача

Тобто до форм була додана логіка, та призначені шляхи до кожної з форм.

Стандартна система автентифікації Django, забезпечує збереження даних користувача, з паролями у хешованому вигляді, та іншими полями, які слугують зручними інструментами для авторизації користувача.

Весь код та налаштування проекту приведені на віддаленому репозиторію GitHub.

4.4 Налаштування бекенд частини веб-додатку

Було створено 5 моделей бази даних засобами фреймворку Django, які описують продукти, їх категорії, замовлення, статус замовлення, та допоміжна модель товар замовлення.

4.4.1 Створення REST API

Було створено кінцеві точки для проведення запитів POST, PUT, UPDATE, GET, DELETE, за протоколом HTTP, до кожної з створених моделей системи. Кінцевим точкам було надано відповідний шлях, який обов’язково починається з “/api/v1”, де “v1” вказує на версію API.

Також було налаштовано рівні доступу до функціоналу REST API. Так, не авторизований користувач може отримати доступ лише до списку товарів, та їх категорій. Він не зможе щось видаляти, модифікувати, або навіть отримувати доступ до даних замовлень інших користувачів.

Авторизований користувач, в свою чергу, може отримати доступ лише до публічних даних описаних вище, або ж до даних, які відносяться безпосередньо до нього. Наприклад, авторизований користувач може отримати дані щодо своїх замовлень, але не зможе отримати дані щодо замовлень інших користувачів.

Адміністратор має доступ до всіх даних в системі, як і до операцій з ними.

До деяких кінцевих точок було додано систему фільтрування за різними показниками, як от пошук замовлень по якомусь користувачу, що доступно для адміністратора. Та додано систему пошуку товарів в рамках API, де виконується пошук по ключових словах у назвах та описам товарів.

Весь код та налаштування проекту приведені на віддаленому репозиторію GitHub [6].

4.4.2 Налаштування автодокументації для кінцевих точок API

Було налаштовано автогенерування документації для створеного REST API. Для цього було встановлено пакет Python, drf-spectacular, оновлені налаштування фреймворку Django, з доданням завантаженого пакету до її додатків, та налаштований шлях для отримання доступу до документації. Таким чином він став доступний за шляхом “/api/docs/”.

Сторінка документації також підтримує 2 види авторизації: за cookie, та звичайну, за логіном та паролем. На рисунку 4.12 представлене вікно авторизації.

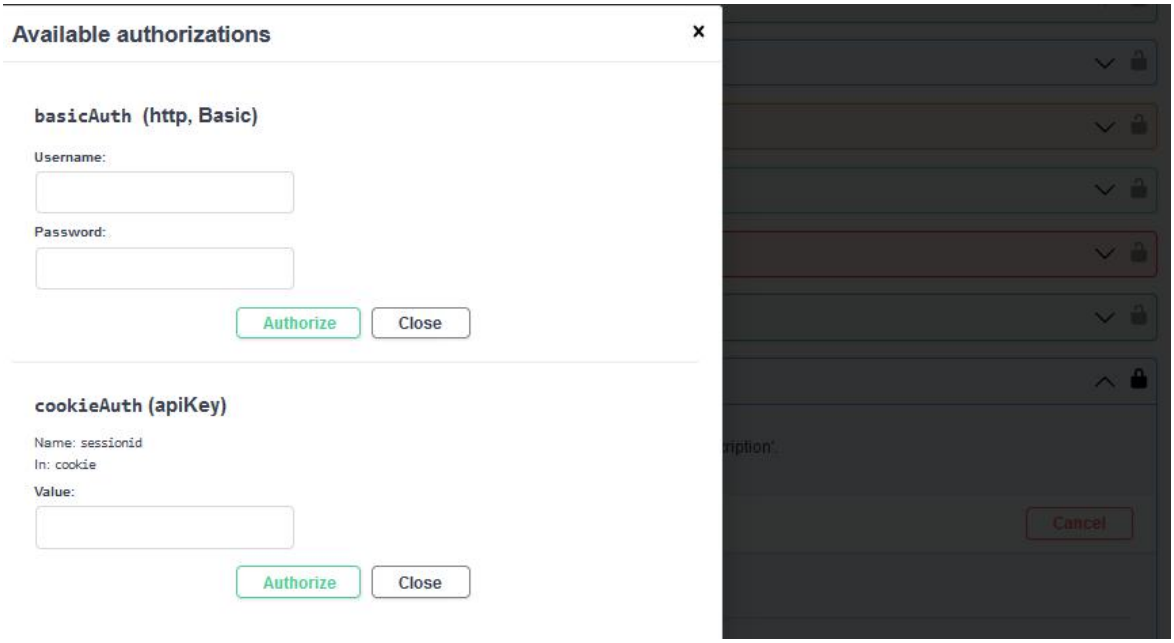

Рисунок 4.12 – Вікно авторизації для кінцевої точки на сторінці документаці

На сторінці автодокументації відображені всі кінцеві точки одним списком.

На сторінці документації також можна протестувати кожну з кінцевих точок, тобто відправити запит до неї, використовуючи інтерфейс, який надає сторінка документації. Окрім передання значень до ключів, для зміни чи створення нової сутності, можна також передавати аргументи запиту, наприклад для фільтрування або пошуку по ключовим словам.

На рисунку 4.13 представлено частину записів на сторінці автозгенерованої документації до створеної REST API.

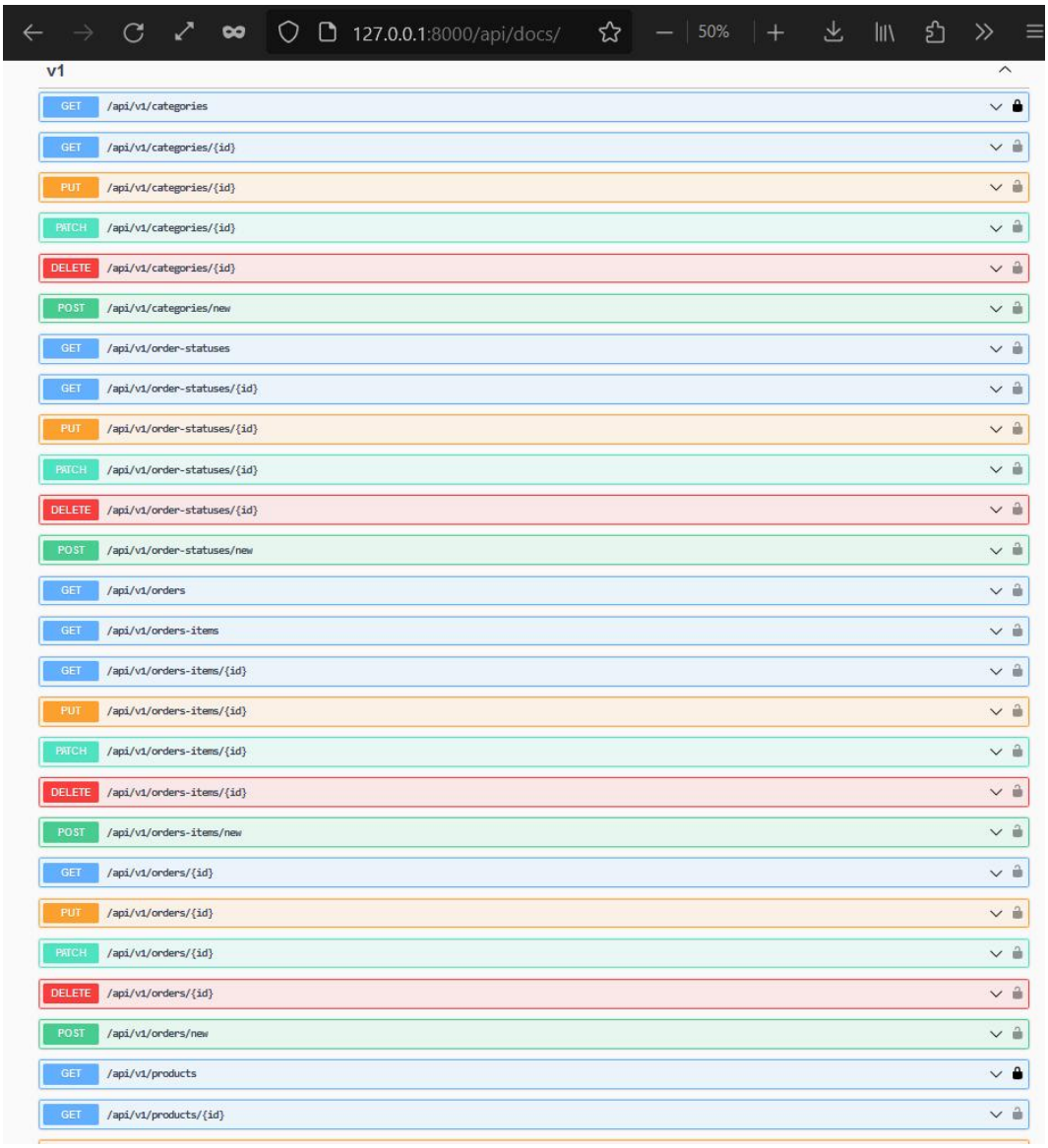

Рисунок 4.13 – Сторінка автозгенерованої документації до створеної REST API

На рисунку 4.14 представлено інтерфейс для відправлення запиту до кінцевої точки REST API.

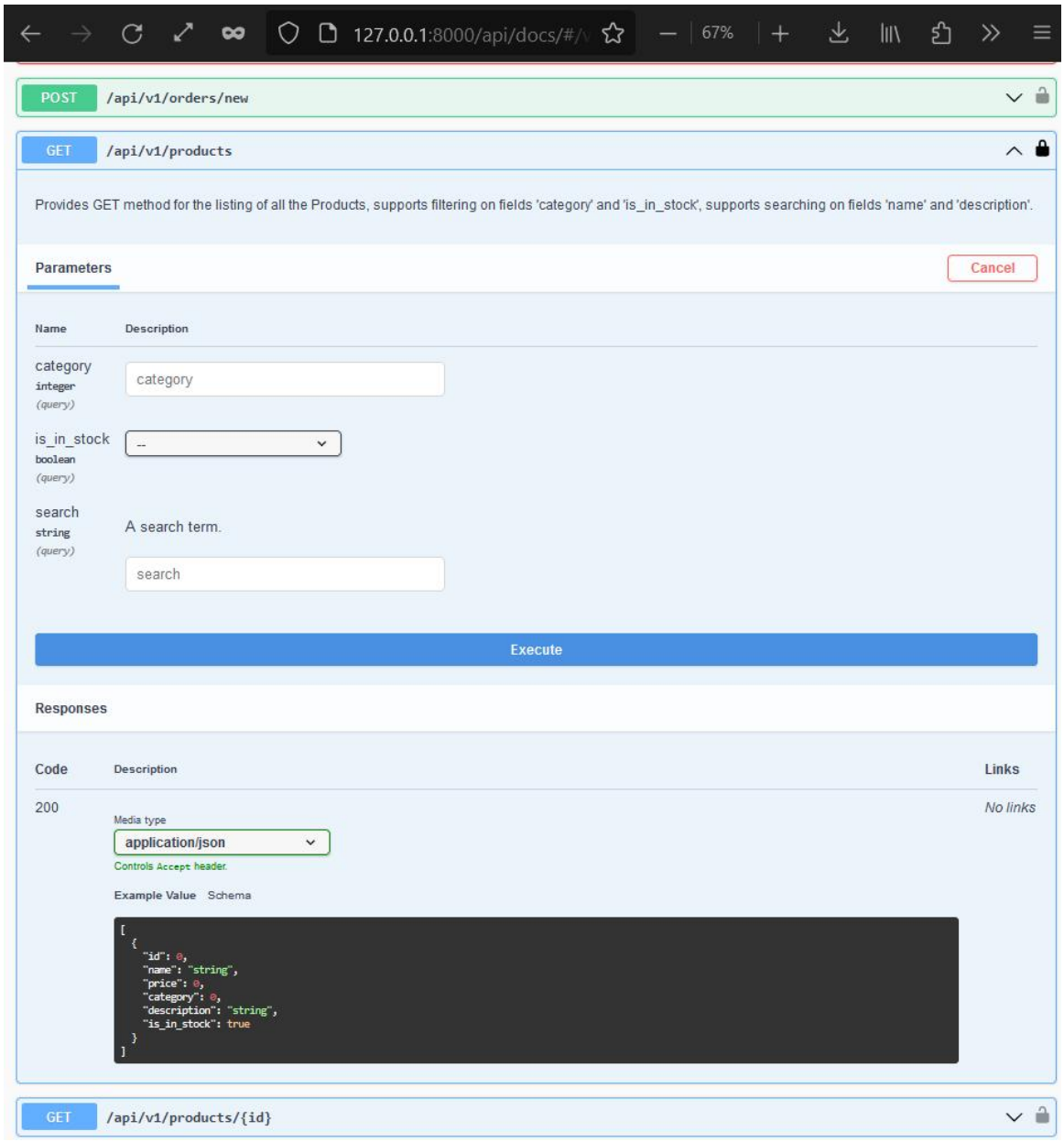

Рисунок 4.14 – Інтерфейс для відправлення запиту до кінцевої точки REST API

ВИСНОВКИ

В рамках кваліфікаційної роботи було розроблено корпоративну мережу для компанії ТОВ «ABP Solutions», згідно з якою, може бути надано обладнанні робочі місця для кожного співробітника компанії.

Також було налаштовано хмарну інфраструктуру для циклу постійної інтеграції написаного програмного коду співробітниками компанії, з автоматичним проходженням тестів та перевірок на відповідність коду загальноприйнятим стандартам.

Було розроблено бекенд частину Інтернет-магазину з налаштованими рівнями доступу до кінцевих точок API, та додано інтерфейс авторизації користувача. Структура веб-додатку була спроектована з урахуванням подальшого масштабування сервісів, та додання нового функціоналу.

Створений проект може бути використаний компанією ТОВ «ABP Solutions» для доопрацювання веб-додатку у частині фронтенду, під свої потреби, та налагодження роботи команди для підтримки роботи Інтернет-магазину на потужностях створеної комп’ютерної мережі.

ПЕРЕЛІК ПОСИЛАНЬ

1. COVID-19 Coronavirus Pandemic [Електронний ресурс] – Режим доступу до ресурсу: COVID-19 Coronavirus Pandemic

2. Global Growth to Slow through 2023, Adding to Risk of ‘Hard Landing’ in Developing Economies. PRESS RELEASE January 11, 2022 [Електронний ресурс] – Режим доступу до ресурсу: https://www.worldbank.org/en/news/press-release/2022/01/11/global-recovery-economics-debt-commodity-inequality

3. Бізнес під час війни [Електронний ресурс] – Режим доступу до ресурсу: https://chm-s.com/biznes-vo-vremya-vojny/

4. Вплив війни на інтернет-торгівлю: як змінювалися онлайн-продажі рітейлерів протягом I півріччя 2022 року [Електронний ресурс] – Режим доступу до ресурсу: https://rau.ua/novyni/vpliv-vijni-na-internet/

5. IT компанія Yalantis [Електронний ресурс] – Режим доступу до ресурсу: https://yalantis.com/

6. GitHub репозиторій Інтернет-магазину компанії ТОВ «АВР Solutions» [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/Gr0ki/e-shop

7. Стаття в Wikipedia: Ethernet [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Ethernet

8. Стаття в Wikipedia: Wi-Fi [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Wi-Fi

9. Транснаціональна компанія Cisco [Електронний ресурс] – Режим доступу до ресурсу: https://www.cisco.com/

10. GitHub. Головна сторінка [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/

11. Gitlab. About [Електронний ресурс] – Режим доступу до ресурсу: https://about.gitlab.com/

12. Bitbucket. Головна сторінка [Електронний ресурс] – Режим доступу до ресурсу: https://bitbucket.org/

13. Sourceforge. Головна сторінка [Електронний ресурс] – Режим доступу до ресурсу: https://sourceforge.net/

14. Python. Головна сторінка [Електронний ресурс] – Режим доступу до ресурсу: https://www.python.org/

15. JavaScript . Сторінка документації від Mozilla [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Web/JavaScript

16. PHP. Головна сторінка [Електронний ресурс] – Режим доступу до ресурсу: https://www.php.net/

17. Java. Головна сторінка [Електронний ресурс] – Режим доступу до ресурсу: https://www.java.com/en/

18. Ruby. Головна сторінка [Електронний ресурс] – Режим доступу до ресурсу: https://www.ruby-lang.org/en/

19. Django. Сторінка документації [Електронний ресурс] – Режим доступу до ресурсу: https://www.djangoproject.com/

20. Django REST Framework. Сторінка документації [Електронний ресурс] – Режим доступу до ресурсу: https://www.django-rest-framework.org/

21. PostgreSQL. Головна сторінка [Електронний ресурс] – Режим доступу до ресурсу: https://www.postgresql.org/

22. drf-spectacular. Сторінка документації [Електронний ресурс] – Режим доступу до ресурсу: https://drf-spectacular.readthedocs.io/en/latest/

23. Docker. Головна сторінка [Електронний ресурс] – Режим доступу до ресурсу: https://www.docker.com/

24. GitHub Actions. Сторінка документації [Електронний ресурс] – Режим доступу до ресурсу: https://docs.github.com/en/actions

25. Coveralls. Головна сторінка [Електронний ресурс] – Режим доступу до ресурсу: https://coveralls.io/

26. Pytest. Сторінка документації [Електронний ресурс] – Режим доступу до ресурсу: https://docs.pytest.org/

27. Unittest. Сторінка документації [Електронний ресурс] – Режим доступу до ресурсу: https://docs.python.org/3/library/unittest.html

28. Pylint. Сторінка документації [Електронний ресурс] – Режим доступу до ресурсу: https://pylint.readthedocs.io/en/latest/

29. Gunicorn. Головна сторінка [Електронний ресурс] – Режим доступу до ресурсу: https://gunicorn.org/

30. Nginx. Головна сторінка [Електронний ресурс] – Режим доступу до ресурсу: https://nginx.org/

31. Meraki . Головна сторінка [Електронний ресурс] – Режим доступу до ресурсу: https://meraki.cisco.com/

32. Amd. Головна сторінка [Електронний ресурс] – Режим доступу до ресурсу: https://www.amd.com

33. Msi. Головна сторінка [Електронний ресурс] – Режим доступу до ресурсу: https://ua.msi.com/

34. Goodram. Головна сторінка [Електронний ресурс] – Режим доступу до ресурсу: https://www.goodram.com/

35. Apacer. Головна сторінка [Електронний ресурс] – Режим доступу до ресурсу: http://www.apacer.com/

36. Canon. Головна сторінка [Електронний ресурс] – Режим доступу до ресурсу: https://global.canon/

37. Atcom. Головна сторінка [Електронний ресурс] – Режим доступу до ресурсу: http://atcom.ua/

38. Docker Hub. Образ nginx:1.23.3-alpine-slim [Електронний ресурс] – Режим доступу до ресурсу: https://hub.docker.com/layers/library/nginx/1.23.3-alpine-slim/images/sha256-60a7532e3b954c902cb651aa29a2c757c495e11c264368fdf77b139985b923c6?context=explore

39. Google reCAPTCHA [Електронний ресурс] – Режим доступу до ресурсу: https://www.google.com/recaptcha/

ДОДАТОК А

Код HTML сторінки з Cisco PT

```
<!DOCTYPE html>
<html>
<head>
</head>
    <div>
        <h3>Комп’ютерна система Інтернет-магазину IT компанії з реалізацією побудови, налаштування та безпеки корпоративної мережі</h3>
    </div>
    <div>
        <h3>Завдання до мережі:</h3>
        <p>Розробити та налаштувати корпоративну мережу підприємства, відповідно до структурної схеми підприємства та технічних вимог.</p>
    </div>
    <div>
        <h3>Завдання до Інтернет магазину:</h3>
        <p>Розробити бекенд систему для Інтернет магазину та налаштувати процес постійної інтеграції коду.</p>
    </div>
</body>
</html>
```