МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: «Web-додаток для конфігурування ПК інтернет магазину «PC Configurer»»

за спеціальністю 122 «Комп'ютерні науки»
освітньо-професійна програма «Інформаційні технології проектування»

Виконавець роботи: студент групи ІТ-81 Беккер Дмитро Олександрович

Кваліфікаційна робота бакалавра
захищена на засіданні ЕК
з оцінкою _____«____»_____ 2022 р.

Науковий керівник _____ к.т.н., доц., Марченко А. В.
(підпис) (науковий ступінь, вчене звання, прізвище та ініціали)

Засвідчую, що у цій дипломній роботі немає
запозичень з праць інших авторів
без відповідних посилань.

Студент _____
(підпис)

Суми-2022

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра інформаційних технологій
Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма «Інформаційні технології проектування»

ЗАТВЕРДЖУЮ
Зав. кафедри ІТ

В.В. Шендрик
«___»_____2022 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА СТУДЕНТУ

Беккер Дмитро Олександрович
(прізвище, ім'я, по батькові)

1 Тема роботи *Web-додаток для конфігурування ПК інтернет магазину "PC Configurer"*

керівник роботи__*Марченко Анна Вікторівна, к.т.н., доцент*

затверджені наказом по університету від «27» квітня 2022 р. №0301-IV

2 Строк подання студентом роботи «10» червня 2022 р.

3 Вхідні дані до роботи технічне завдання на розробку web-додатку

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) аналіз предметної області, проектування web-додатку, розробка web-дода

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) постановка задачі, дослідження продуктів-аналогів, вимоги до додатку, діаграма IDEF0, діаграма декомпозиції, діагарма використання web-додатку, діагарма структурного аналізу, діагарма послідовності, діаграма

комопонентів, даіграма розгортання, діагарма потоків даних, моделі бази даних, архітектура додатку, PIM система Google Firebase, пошуковий двигун Algolia Search Engine, використання web-додатку

6. Консультанти розділів роботи:

Розділ	**Консультант**	**Підпис, дата**	
		Завдання видав	**Завдання прийняв**

7.Дата видачі завдання_____05.10.2021_____

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Постановка задачі проекту	20.10.2021-04.11.2021	
2	Аналіз додатків-аналогів	05.11.2021-09.11.2021	
3	Складання технічного завдання	10.11.2021-20.11.2021	
4	Вибір засобів реалізації додатку	21.11.2021-21.11.2021	
5	Створення прототипу клієнтської частини додатку	22.11.2021-29.11.2021	
6	Розробка архітектури додатку	30.11.2021-15.12.2021	
7	Написання коду для додатку	16.01.2022-14.03.2022	
8	Тестування web-додатку	15.03.2022-18.04.2022	
9	Підготовка до випуску	19.04.2022-01.05.2022	
10	Оформлення документації	02.05.2022-09.06.2022	

Студент _____ Беккер Д.О.
(підпис)

Керівник роботи _____ к.т.н., доц. Марченко А.В.
(підпис)

РЕФЕРАТ

Тема кваліфікаційної роботи бакалавра «Web-додаток для конфігурування ПК інтернет магазину «PC Configurer»».

Пояснювальна записка складається зі вступу, 3 розділів, висновків, списку використаних джерел із 20 джерелами та додатків. Загальний обсяг роботи – 102 сторінка, у тому числі 72 сторінок тексту без додатків, 3 сторінки використаних джерел та 22 сторінок додатків.

Кваліфікаційну роботу бакалавра присвячено розробці web-додатку для конфігурування ПК інтернет-магазину «PC Configurer».

У роботі проведено дослідження видів архітектури, виконано аналіз додатків-аналогів, написано технічне завдання на розробку, побудовано інфраструктуру роботи додатку в мережі сервісів та 3d party бібліотек, описано архітектуру та продемонстровано роботу додатку.

Результатом зробленої роботи є розроблений web-додаток для конфігурування ПК.

Ключові слова: web-додаток, мікросервісний, нереляційна БД, конфігурування ПК, web, Java, Spring Boot, React, Algolia, Search Engine, PIM System, Firebase.

ЗМІСТ

ВСТУП

Зараз досить помітна тенденція, коли велика кількість робіт на планеті діджиталізуються. Це викликає потребу для стрімкого зростання апаратного забезпечення для людей. Усе більше працівників потребують якісного hardware для їхньої роботи. Такий стрімкий ріст породжує велику кількість потреби в правильному обладнанні. Хоч і більшість персональний комп'ютер (ПК) уже продаються зібраними, та все ж для кожної задачі потрібні різні комплектуючі. Тому більшість людей хочуть взяти процес зборки ПК у власні руки, або передати цю роботу спеціалістам. Досить часто користувачам не потрібен занадто потужний, або дорогий комп'ютер, то у такому випадку треба обирати з не такої вже великої кількості. Можливість конфігурувати персональний комп'ютер дає можливості для вибору саме того, що буде використовуватися й за розумну ціну.

Як кажуть багато заголовків сервісів хмарних технологій – «Плати за те, що будеш використовувати». Так само потрібно обирати правильний ПК. Не має сенсу купувати дорогий та високопродуктивний комп'ютер для задач, які пов'язані із офісом. Або, наприклад, для деяких завдань потрібна висока продуктивність саме графічного процесора, але не потрібно так багато оперативної пам'яті. Хоч як би не старалися зібрати оптимальний персональний комп'ютер в магазинах, усе одно вони не знають, для яких саме задач буде він використаний.

Також через великий попит існує велика кількість комплектуючих для ПК. Це дає ринку добре розвиватися, але в той же час через велику кількість цих компонентів важко обрати ті, котрі потрібні. Тим паче уже давно є монополісти у власних сферах комплектуючих, через що обирати апаратні запчастини може стати важкою задачею.

Більшість сайтів, які мало спеціалізуються на комп'ютерах, а продають всю техніку не достатньо детально розробляють каталог подібних продуктів.

Через це важко шукати потрібні товари, без спеціальних вказівників. Також такі web-сайти не показують прогрес збору ПК, та сумісність компонентів між собою.

Тому метою даного дослідження є полегшення зборки комп'ютера для користувачів різного рівня підкованості та обізнаності в комп'ютерах за рахунок створеного web-додатку.

Саме з вище зазначеною ціллю надання свободи вибору, та контролю користувачу, який хоче зібрати саме те, для чого йому потрібен буде комп'ютер – створюються даний проект. Завдяки простому та інтуїтивно-зрозумілому інтерфейсу користувач буде легко орієнтуватися при повній та частковій зборці ПК. Також за допомогою підрахунку основних параметрів комп'ютера буде відразу зрозуміло сумарну кількість потрібної електроенергії, або якого об'єму потрібен акумулятор для існуючих комплектуючих. Додаток побудований таким чином, що спочатку ідуть основні компоненти, а вже потім залежні, для підрахунку метрик та оптимізації ПК.

Основними задачами є створення обширної системи для максимальної свободи та конфігурації web-додатку, його розширення. Важливим пунктом є надійність додатку, це означає високий рівень витримки, та максимальний фокус до виключних ситуації у web-додатку. Також важливим аспектом є швидкість роботи додатку, так як кількість товарів може сягати величезних об'ємів. Одним із невід'ємних частин додатку повинен бути інтуїтивно зрозумілий та зручний інтерфейс.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд останніх досліджень і публікацій

Монолітна архітектура web-додатку, суть якого, полягає в тому що всі функціональні можливості ресурсу є одним суцільний додатком. Структура всередині додатку сильно зв’язна та слабо розподілена за функціональними призначеннями. Проста структура, якщо додаток не великий, але знад-то громіздка, для великого масштабу ресурсу [1].

Можливості має всі ті, що і інші типи web-додатків. Перевагами є централізоване ядро всіх процесів додатку, що є спрощенням для розробки та розгортання додатку на сервері або у хмарі.

Мінусами є те, що такий додаток важко підтримувати, у випадку росту функціоналу. Також такий додаток складно тестувати, так як модулі не розділенні на сервіси, тому команда тестувальників повинна добре розуміти внутрішнє розміщення програмних модулів додатку. Через складність тестування зростає ризик знаходження несправностей уже в процесі в процесі експлуатації додатку на сервері. Через це складно буде шукати проблеми та усунути ці помилки в системі.

Onion Architecture (Розділена на шари архітектура). Додаток має структуру, яка розділена на шари відповідальності, що сильно пов’язані між собою. Але наприклад модулі верхнього рівня не можуть взаємодіяти із модулями центрального рівня, якщо вони не знаходяться на прямому контакті із ним. За часту додаток розділять на такі шари:

- Domain entities, бізнес моделі;
- Repository interface, рівень комунікації із базою даних;
- Service interface, сервісний рівень;
- Infrastructure (контролери для додатку), UI (рівень користувальницької частини), Unit Tests(тестування сервісного шару).

Всі вище зазначені рівні залежать одні від одного починаючи із рівня бізнес моделей і до рівня представлення. Наприклад рівень репозиторію, тобто рівень роботи із базою даних, залежить від рівня бізнес моделей [2].

Можливостями є те, що в такому додатку через розділення на шари можна легко організувати модулі, які будуть гнучкими для розширення функціоналу додатку. Додаток є простим у тестуванні через те, що є рівень сервісу і кожний компонент сервісного шару можна тестувати окремо. Через жорстку структуру шари не мають більше однієї залежності із іншим шаром ресурсу.

З недоліків додатку є те, що хоч розподілення в додатку є, це все-одно великий один великий додаток, який потрібно розгорнути на сервері чи у хмарі. Це означає, що у випадку, якщо один із модулів не зможе працювати, то цей суцільний додаток потрібно буде перезавантажити повністю і це у кращому випадку. Найгірший же випадок, що додаток буде далі працювати із частиною непрацюючого функціоналу [3].

Узагальнена характеристика підходів до проектування web-додатку наведена у таблиці 1.1.

Таблиця 1.1 – Узагальнена характеристика

Назва	Основні характеристики	Переваги	Недоліки
Monolith	Суцільність, висока зв’язність компонентів,	Швидка розробка, легкість у написанні для малого розміру додатку	Модулі сильно зв’язані, через це важно вносити зміни в існуючий функціонал, та додавати новий. При несправності важко знайти помилку. Важко тестувати додаток
Onion	Середня зв’язність компонентів, чітка структура, розподілені на шари модулі	Середній час розробки, легке тестування, висока гнучкість при додаванні нового функціоналу до системи, або модифікації старого функціоналу	При несправності, великий час на те щоб, знову запустити додаток

Продовження таблиці 1.1 – Узагальнена характеристика

Microservice	Середня зв’язність, розподіленість на самостійні сервіси для роботи додатку, різноманітність до підходу розробки індивідуального сервісу	Більше середнього часу на розробку, легке тестування, висока гнучкість при додаванні нового функціоналу до системи, або модифікації старого функціоналу, швидкість роботи	Великий час розробки, багато часу потрібно на розгортання сервісів

Підсумовуючи вище написане можна зробити висновки, що найкращий варіант для заданих вимог та поставлених задач буде – мікросервісна архітектура. Вона дозволяє розширювати функціонал додатку з максимальною легкістю [4]. Такий тип web-ресурсу можна добре протестувати. Сервіси в майбутньому можуть бути замінені. Та у випадку падіння одного із сервісів, потрібно лише щоб цей сервіс перезапустили, а не весь додаток. Саме через гнучкість та свободу у виборі реалізації сервісів було і обрано такий варіант архітектури [5].

1.2 Аналіз програмних продуктів - аналогів

Сьогодні існує обширна кількість аналогів такого типу додатків в Інтернет-просторі, але велика кількість web-ресурсів мають деякі недоліки або відсутність

потрібного функціоналу. Також у деяких web-ресурсів не вистачає важливих метрик для відстежування.

Перший аналог, який буде розглянуто – це telemart.ua. Додаток допомагає конфігурувати ПК із великою кількістю товарів по різним категоріям. Також має місце процес, який відображає процент повністю спроможного до роботи комп'ютера. Додаток має зручний дизайн для вибору комплектуючих, реєстрації та покупки товару. Також із приємних фішок web-ресурса є додаткові послуги на кожний обраний компонент. Головна сторінка web-додатку представлена на рисунку 1.1.

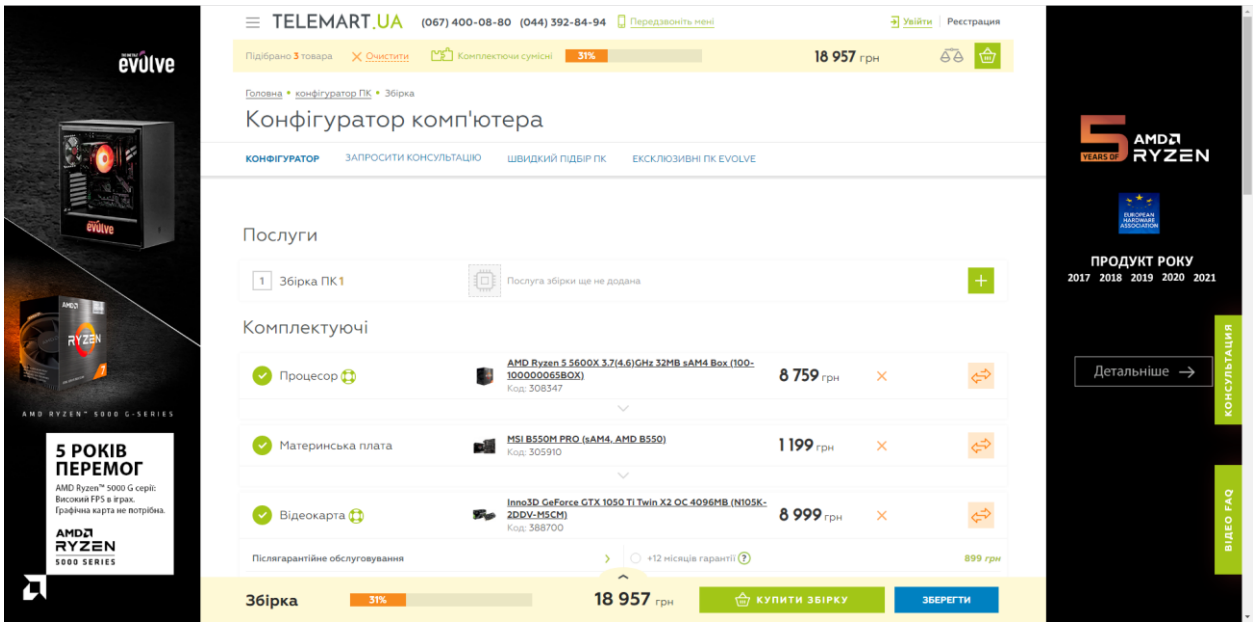


Рисунок 1.1 – Вигляд головної сторінки web-додатку telemart.ua

До мінусів даного додатку можна віднести дизайн, який не імпонує загальній структурі, кольори були обрані не правильно. Також не вистачає метрик для розуміння загальної кількості Вт потрібних для ПК, та чи буде достатньо обраного блоку живлення для підтримки працездатності всіх комплектуючих, та який є запас на випадок заміни компонентів. Сторінка для оформлення замовлення представлена на рисунку 1.2.

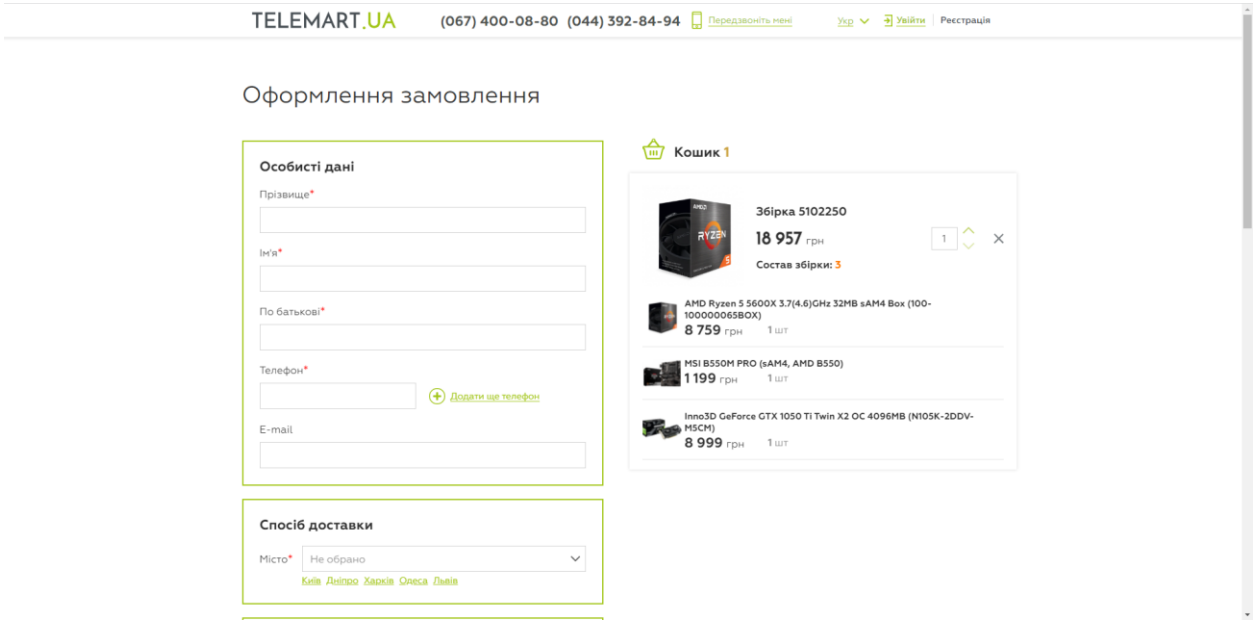


Рисунок 1.2 – Вигляд сторінки для оформлення замовлення

Сторінка із оформленням замовлення також доволі зручна і дизайн на ній вже виглядає краще.

Наступний аналог це – hyperpc.ru. Щодо цього web-додатку, можна сказати, що він доволі специфічний. При використанні ресурсу можна обрати уже готові конфігурації за допомогою основних параметрів. Але їх всього три (ціна, відеокарта та процесор), що звужує можливість для конфігурації (рис. 1.3). Дизайн є доволі різким, хоча сторінка товарів виглядає цікаво (рис. 1.4). Тому такий додаток не є зручним для гнучкої збірки комп’ютера.

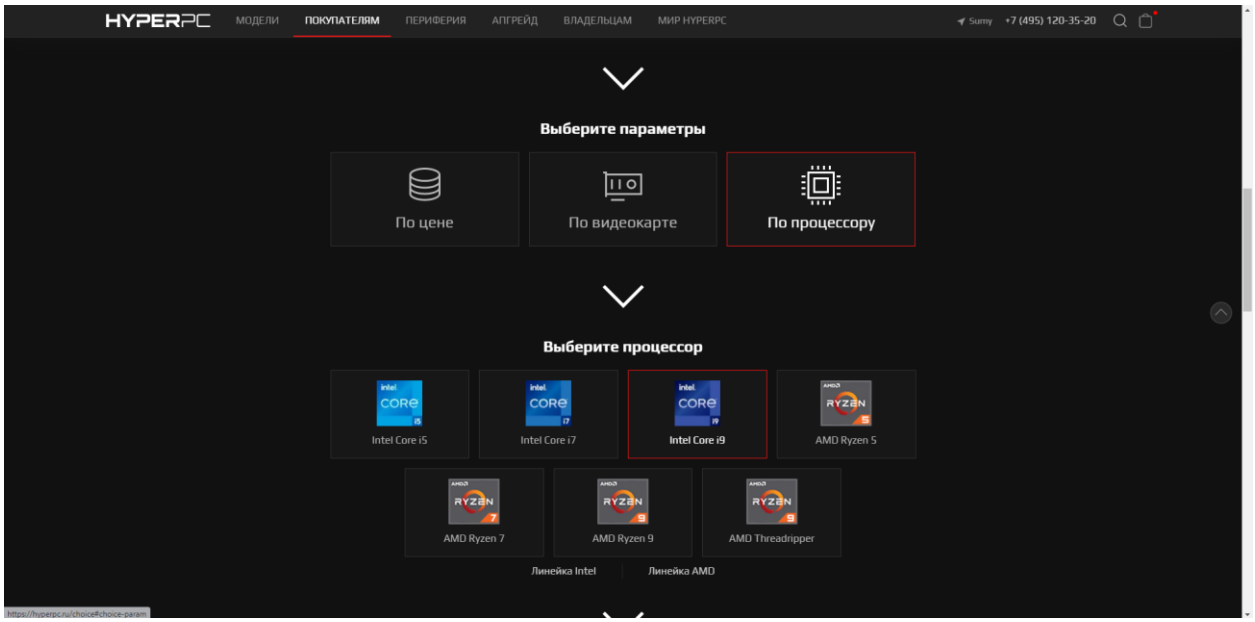


Рисунок 1.3 – Вигляд основної сторінки

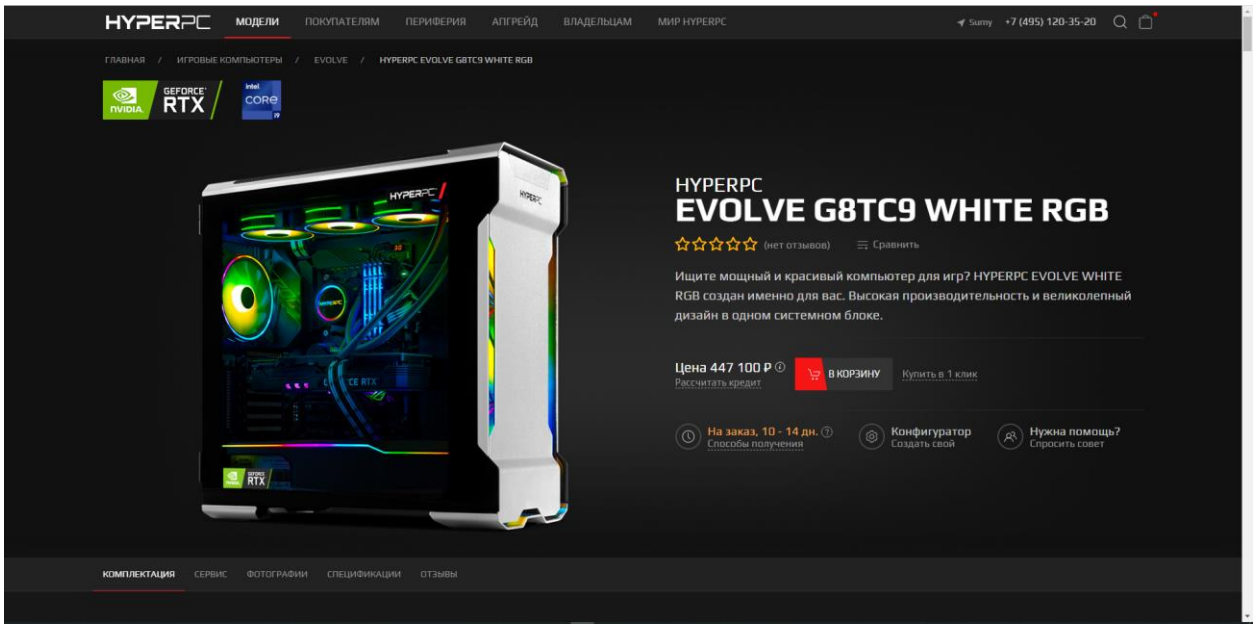


Рисунок 1.4 – Вигляд сторінки конкретного товару

Цей web-додаток має доволі не інтуїтивний дизайн, що погіршує досвід користувача (UX), який приходить на нього в перший раз. Сторінка для оформлення замовлення доволі зручна.

Останній аналог – це inteam.ua. Цей web-додаток не є простим у використанні, тому потрібен деякий час для того, щоб почати ним користуватися. Дизайн додатку також виглядає занадто просто. У шапці ресурсу обрані дивні варіанти для заднього фону кнопок. Також на ньому погано організована робота із текстом. Даний додаток майже не має стилів і тому виглядає як велика кількість тексту на якому важко концентруватися (рис. 1.5 - 1.6).

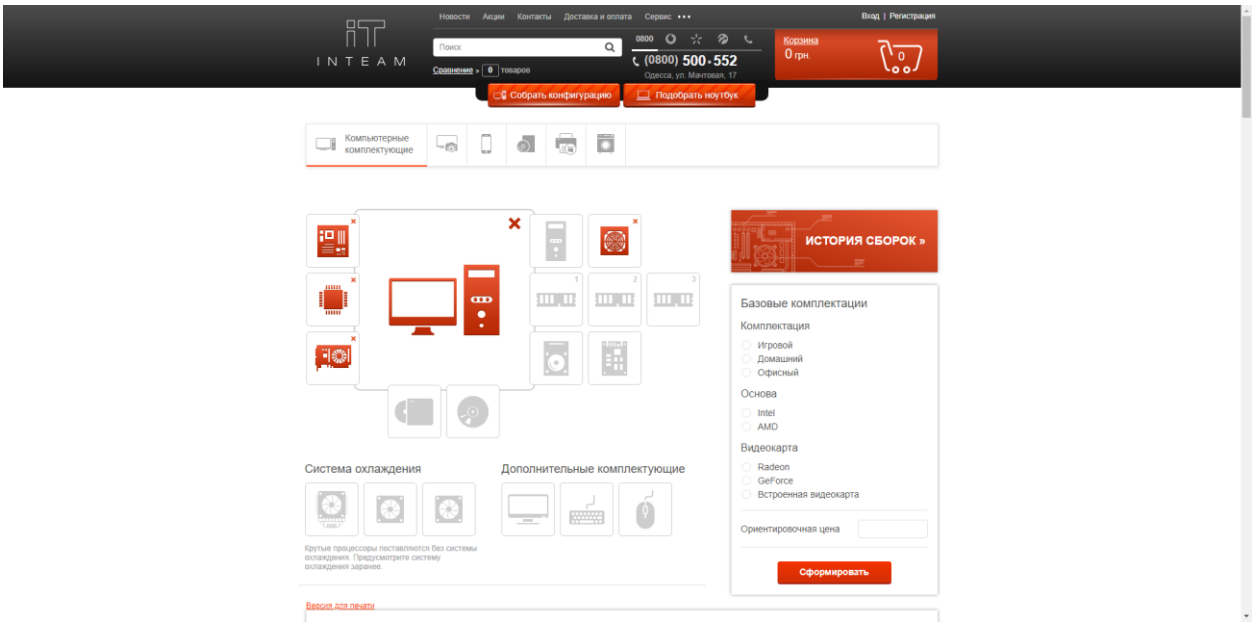


Рисунок 1.5 – Вигляд головної сторінки додатку

Рисунок 1.6 – Вигляд головної сторінки додатку

Також мінусом додатку є те, що товари в корзині зберігаються лише один день. При оформленні замовлення необхідна реєстрація, що не є зручним на випадок одноразового користування додатком (рис. 1.7).

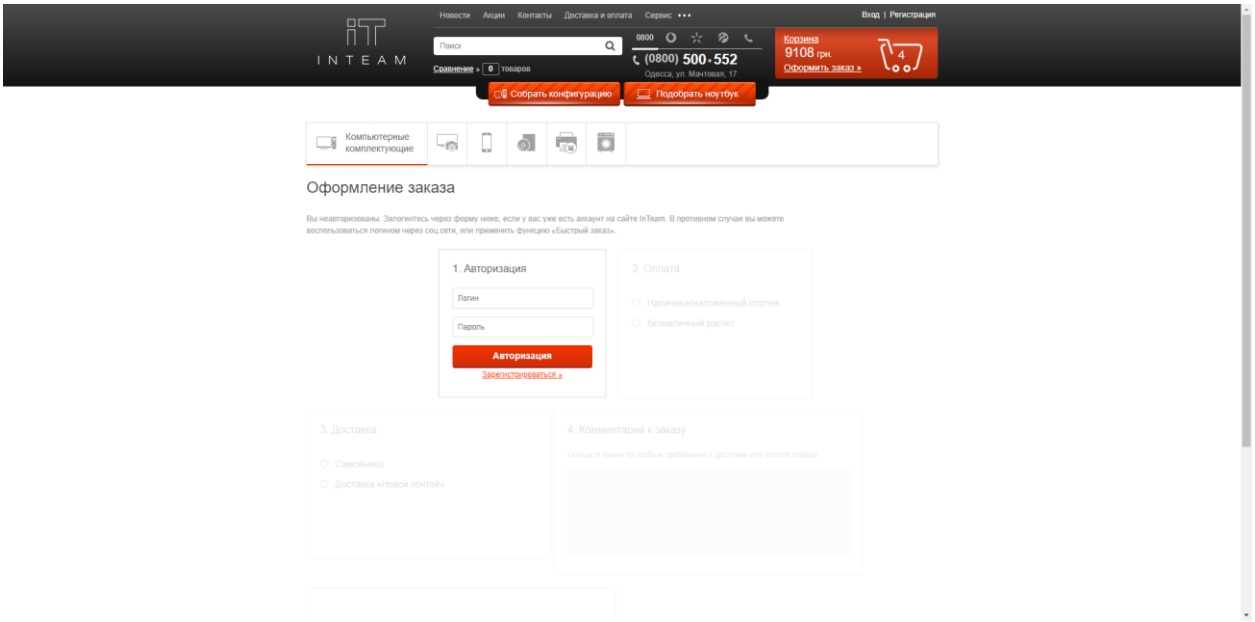


Рисунок 1.7 – Вигляд сторінки для оформлення замовлення

У результаті проведеного аналізу аналогів додатків для збору ПК можна зробити підсумки, які представлено в таблиці 1.2.

Таблиця 1.2 – Порівняльна таблиця характеристик аналогів web-додатків

Харакетристика/ Додаток	“telemart”	“hyperpc”	“inteam”	“PC Configurer”
Сучасний дизайн	+	+-	-	+
Зручний інтерфейс	+	-	+-	+
Інтерактивність	+	-	+	+
Функціональність	+	+-	+-	+
Навігація	+	-	-	+
Обов’язкова реєстрація користувача	-	-	+	-
Гнучкість конфігурування	+	-	+-	+
Швидкість	+	-	+	+

Із таблиці 1.2 зрозуміло, що не має зручного додатку для вирішення поставленої задачі, тому було вирішено розробити власний web-ресурс, щоб усунути ці недоліки, та використати сильні сторони аналогів. Розроблюваний програмний додаток повинен мати сучасний дизайн та зручну навігацію. З функціональних можливостей потрібно додати відстеження метрик.

1.3 Постановка задачі web-додатку

Метою даного дослідження є розроблення додатку для полегшення збірки комп’ютера для користувачів різного рівня підкованості та обізнаності в них за рахунок створеного web-додатку.

За результатами аналізу предметної області сформульована задача проектування: розробити додаток для систематизованого та поетапного збирання частин комп’ютера, що допомагає конфігурувати та підібрати компоненти для збірки ПК під різні задачі.

Для вирішення поставленої задачі необхідно вирішити такі під задачі проектувння:

- визначити актуальність роботи та дослідити предметну область;
- провести аналіз аналогів, виділити їх переваги та недоліки;
- спроектувати модель та структуру web-каталогу;
- обрати технології розробки;
- створити прототип web-каталогу;
- реалізувати структуру web-каталогу;
- розробити функціонал web-каталогу;
- виконати тестування;
- запропонувати шляхи покращення для подальшої роботи web-каталогу.

Використання даного додатку дозволить користувачам скоротити час на пошук апаратних частин персонального комп’ютера, процес конфігурації буде повністю поетапним, що не дозволить загубити важливих частин ПК. Додаток дозволить починаючим користувачам навчитися і зрозуміти із чого збирається комп’ютер та як обрати найкращий, збираючи його самостійно.

Після проведення аналізу та встановлення мети проекту було поставлено наступні задачі до проекту (функціональні можливості):

- конфігурування ПК, де всі важливі компоненти будуть розділенні між секціями, в яким можна обирати товар;

- робота із каталогом. В під модулях із каталогом обраного апаратного або програмного компоненту можна вибрати товари за ціною (діапазон від найнижчої до найвищої), назвою товару, брендом та специфічними характеристиками компоненту (наприклад процесор: сокет, модельний ряд, покоління, серія, кількість ядер, кількість потоків і т.д.). Також товари можна сортувати за ціною, популярністю та назвою);
- авторизація та реєстрація;
- додавання, редагування та видалення товарів та категорій (модераторська частина додатку).

Вимоги до проекту в цілому, структури web-додатку, видів забезпечення та функціонування системи описані у технічному завданні на розробку проекту (додаток А).

Web-додаток буде написано із використанням мови програмування Java та фреймворку Spring Boot, для створення мікросервісів. Для написання фронтенд частини буде використовуватися HTML для розмітки сторінок та задання струкутри, CSS для стилів, JS динамічності, та фреймворк (React.js) налаштування серверної сторони із фронтендом та додатковий функціонал. Для управління та зберігання даних в додатку буде використовуватися нереляційна база даних – mongodb. Docker та Kubernetes, будуть використані для розгортання додатку в будь-якому середовищі.

2 МОДЕЛЮВАННЯ ТА ПРОЕКТУВАННЯ WEB-ДОДАТКУ

2.1 Функціональне моделювання web-додатку в IDEF0

IDEF0 було розроблено базуючись на Structured Analysis and Design Technique (SADT), широковідомої мови графічного представлення. IDEF0 є корисним для аналізу, особливо функціонального. IDEF0 застосовує оптимізовані засоби візуалізації щоб полегшити залучення експертів для прийняття рішень. IDEF0 дозволяє визначати функції, які включають виконання необхідних дій та перевірку правильності поточного стану. Діаграми IDEF0 у вигляді "прямокутника та стрілки" відображають функцію у вигляді прямокутника, а інтерфейси - у вигляді стрілок, що входять або виходять із скриньки у функцію або з функції [6].

На рисунку 2.1 представлена діаграма A-0 верхнього рівня. До її складу входять: функціональний блок, вхідні та вихідні дані, елементи управління та механізми для виконання функції.

Функціональне моделювання web-додатку в нотації IDEF0 представлене на рисунку 2.1.

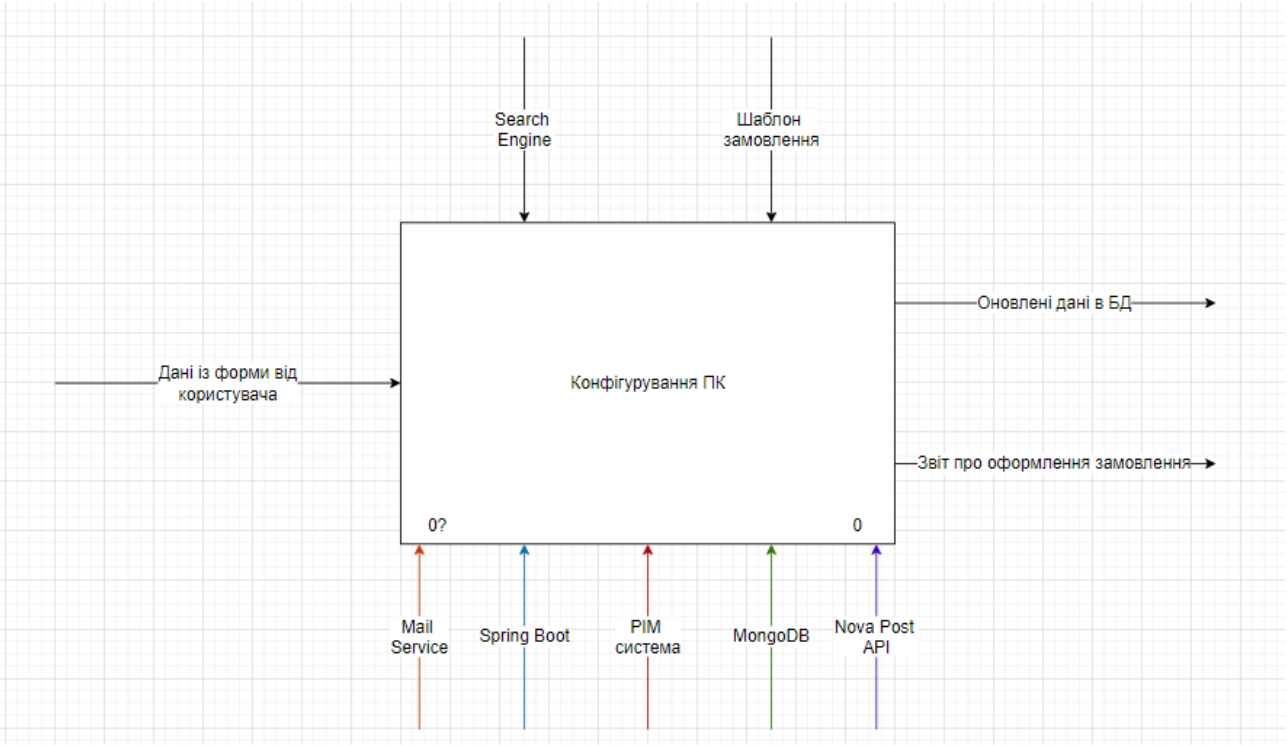


Рисунок 2.1 – IDEF0

Вхідними даними системи є наступні джерела:

- Дані із форм від користувача (всі дані що приходять із форм на користувальницькій частині додатку).

Контролюючими механізмами є:

- Search Engine (пошуковий двигун, який отримує дані із БД, та накладає мітки для швидкого пошуку);
- Шаблон замовлення, для формування запиту для менеджера, та його підтвердження.

Механізми, що використовуються для виконання роботи:

- Mail service, використовується для відправлення звітів менеджерам на пошту;
- Spring Boot, для запуску додатку на Spring системі із вбудованим сервером Tomcat;
- PIM система (Google Firebase) використовується для наповнення каталогу товарами та категоріями;

- MongoDB, використовується для управління даними та передачею її у додатку;
- Nova Post API, використовується для пошуку відділень за містом, та пошук відділень по обраному місту.

Вихідні дані системи:

- Оновлені даних в БД;
- Звіт про замовлення ПК користувачем, яке буде надіслано менеджеру.

Для поглибленого визначення основних функцій додатку, було представлено більш детальну модель його роботи. Через вхідні данні було визначено основний потік їх використання в функціональних модулях. Було побудовано поетапну схему використання основного призначення додатку. Кожен новий етап витікає із попереднього, таким чином будуючи потік для основного призначення додатку по етапам. Кожен етап опису основну функціональну одиницю, яка робить свій вклад у вирішення задачі. На вхід такий процес приймає дані на обробку, і на виході дає результуючі дані. Також в схемі декомпозиції описано залежності від механізмів для вирішення задачі [6].

Декомпозиція функціональної моделі web-додатку представлена на рисунку 2.2.

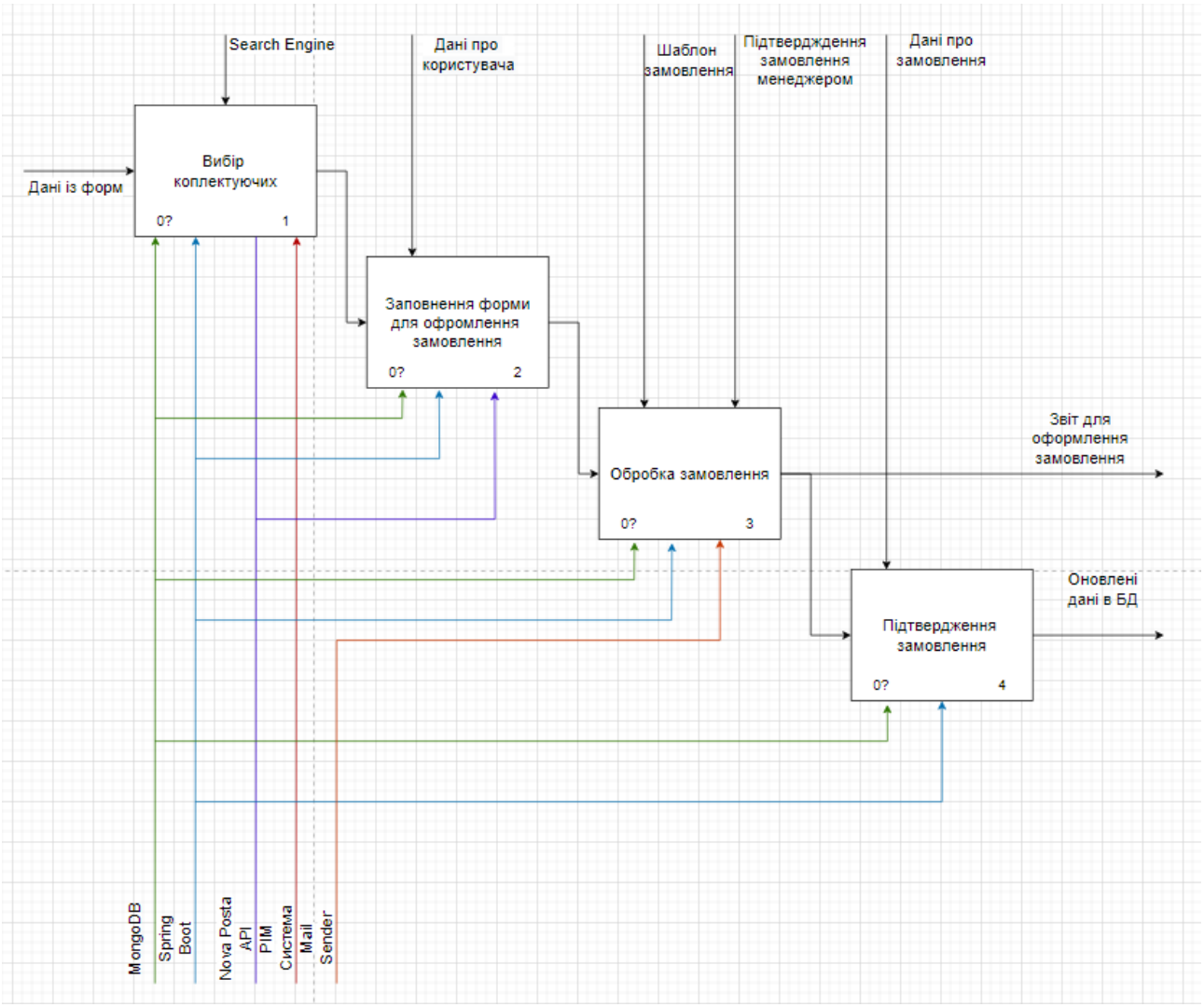


Рисунок 2.2 – Декомпозиція бізнес процесу web-додатку

2.2 Проектування інформаційної системи

2.2.1 Діаграма варіантів використання web-додатку

Для досягнення цілей функціонування спочатку будується модель у формі діаграми варіантів використання (use-case diagram), яка описує функціональне призначення системи. Діаграма варіантів використання є вихідною моделлю системи під час її проектування та розробки.

Діаграма варіантів використання UML – це основна форма вимог до системи/програмного забезпечення для нової програми, що розробляється. Варіанти використання визначають очікувану поведінку (що), а не обширний засіб його реалізації (як). Варіанти використання можуть бути у текстовому та візуальному варіантах (наприклад, діаграма варіантів використання) [7]. Ключовою концепцією моделювання у вигляді діаграми варіантів використання є проектування системи з погляду кінцевого користувача. Це ефективна техніка візуалізації передачі поведінки системи в термінах користувача за допомогою вказівки зовнішньої видимої поведінки системи [8].

Діаграма сценарію використання є простою і не показує деталізації варіантів використання:

- Вона узагальнює певні зв’язки між варіантами використання, системами та дійовими особами;
- Вона вказує на правильну послідовність кроків для досягнення цілей кожного з компонентів варіанта використання.

Діаграма варіантів використання в нотації UML представлена на рисунку 2.3.

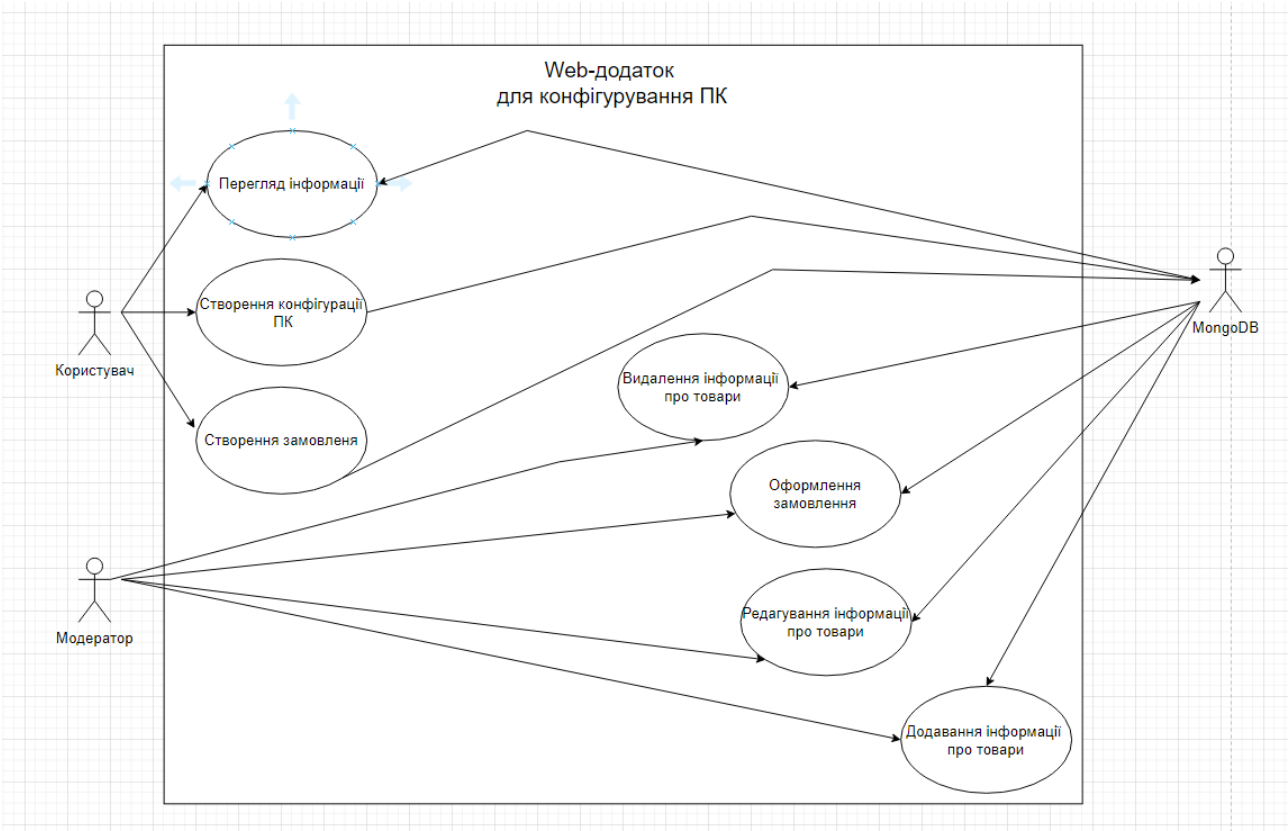


Рисунок 2.3 – Діаграма варіантів використання

У системі присутні 4 актори:

- Користувач додатку, який взаємодіє із системою;
- MongoDB, яка надає можливість зберігати, модифікувати, читати та видаляти дані;
- Модератор, контролює продукцію.

2.2.2 Діаграма структурного моделювання

Структурне моделювання допомагає представити потік та взаємодію між базою даних та додатком. Вона описує основні операції, які можуть бути проведені у web-ресурсі за допомогою БД.

Діаграма структурного моделювання web-додатку представлена на рисунку 2.4.

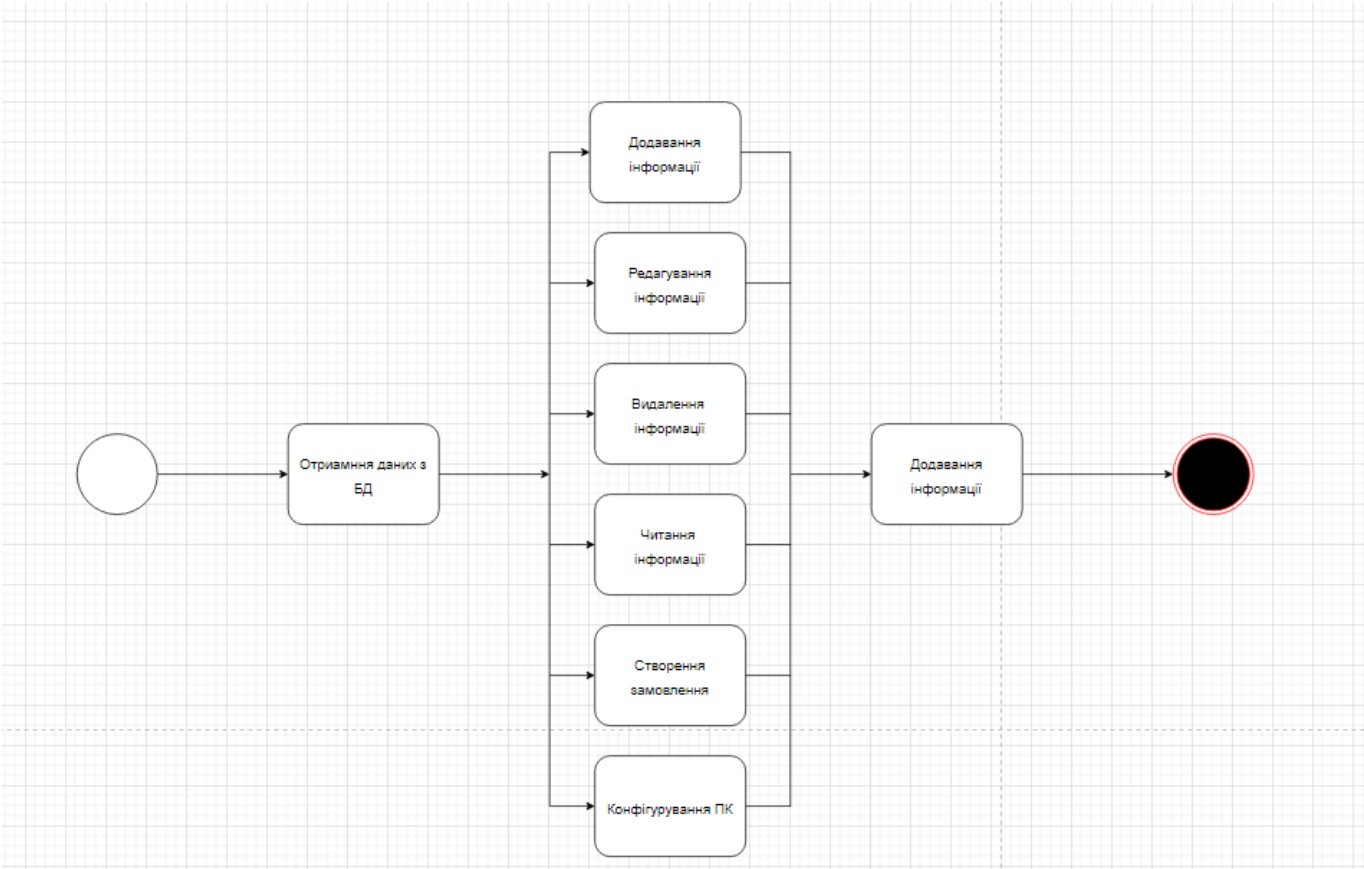


Рисунок 2.4 – Діаграма роботи web-додатку як скінченного web-додатку

Складовими схеми є потік даних та операції, які проводяться над ним. Для початку отримується доступ до даних із БД. Із даними можна проводити наступні прості операції:

- Додавання інформації, наприклад додавання товарів, або користувачів до системи;
- Редагування інформації, про товари та користувачів;
- Видалення інформації із БД, про товари та користувачів;
- Читання інформації із БД, про товари та користувачів;

До складних операцій можна віднести наступні операції:

- Створення замовлення на покупку товарів;
- Конфігурування ПК (зберігається лише для зареєстрованих користувачів).

2.2.3 Діаграма класів аналізу

Основна відмінність моделі варіантів використання від моделі аналізу полягає в тому, що при побудові першої основна увага приділяється визначенню функціональних можливостей (вимог) системи, а при побудові другої – їх уточненню з урахуванням внутрішньої організації (архітектури) проектованої системи [9].

Побудова моделі аналізу необхідна:

- для виявлення внутрішньої архітектури (визначення підсистем та основних класів);
- для пошуку альтернативних варіантів реалізації системи (підсистем) та вибору основного;
- для уточнення всіх вимог (функціональних та нефункціональних).

Діаграма класів web-додатку представлена на рисунку 3.5.

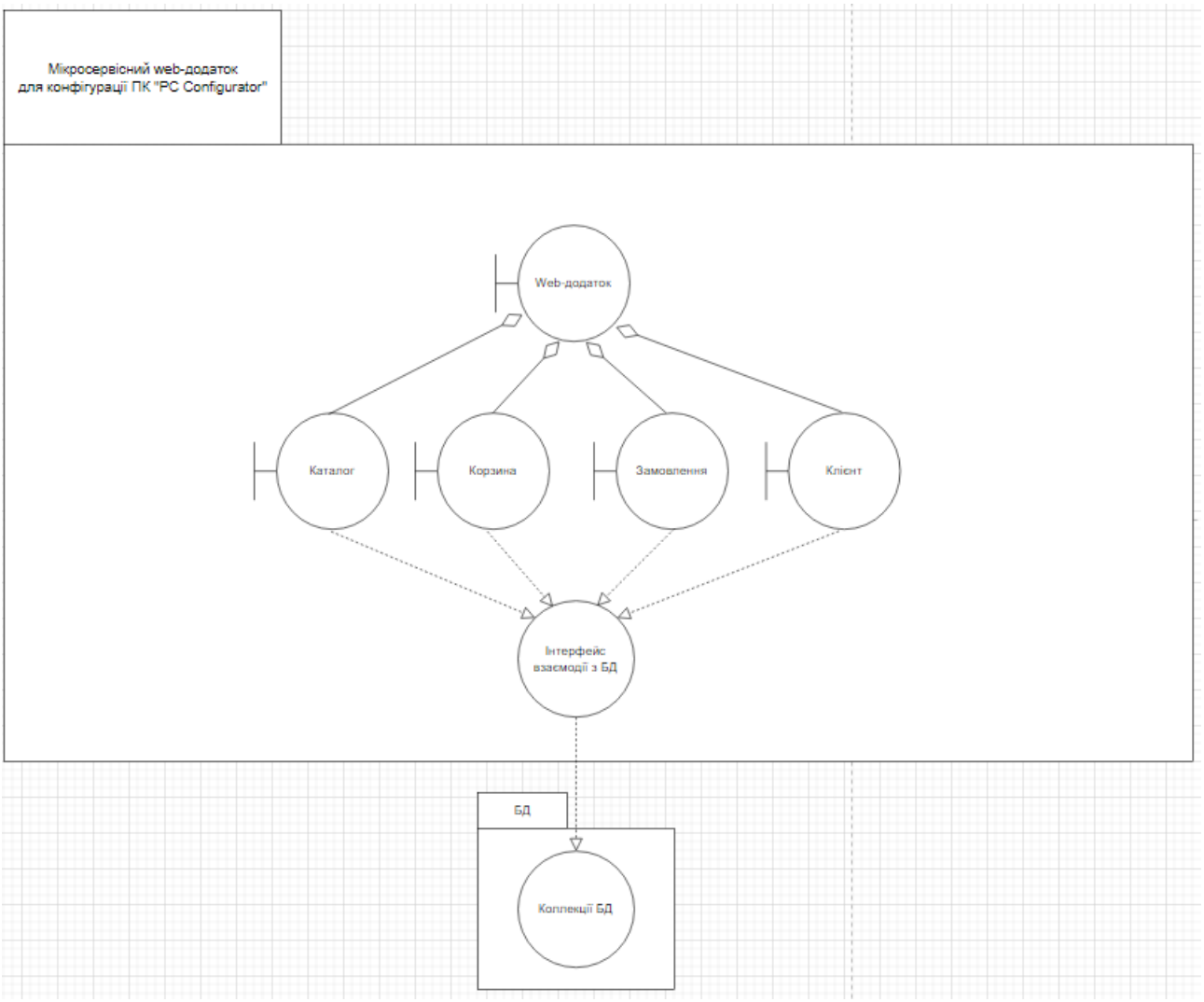


Рисунок 2.5 – Діаграма класів аналізу

На діаграмі зображено головний елемент це додаток, від нього залежать основні компоненти додатку:

- Компонент каталог додатку, надає можливість переглядати компоненти та конфігурувати ПК;
- Компонент кошику, дозволяє конфігурувати кількість товарів для створення замовлення;
- Компонент замовлення, надає можливість заповнити деталі замовлення;
- Компонент користувача, надає можливість, зайти, зареєструватися, вийти та отримати дані про користувача із аккаунту.

Всі описані вище компоненти реалізують функціонал інтерфейсу БД, у випадку додаток це шар репозиторію, який реалізує зв'язок для написання запитів до БД.

База даних у даному випадку представляє собою відокремлений від додатку самостійний компонент.

2.2.4 Діаграми послідовності web-додатку

Діаграма послідовності – є різновидом діаграми взаємодії, адже вона визначає порядок групи об'єктів, які працюють разом. Їх застосовують розробники ПЗ та бізнес-аналітиками для визначення вимог до нової системи або документування існуючого процесу. Діаграми послідовностей іноді називають діаграмами подій чи сценаріями подій [10].

Діаграми послідовності можуть бути корисними довідниками підприємств та інших організацій. Вони використовуються щоб:

- Подати деталі сценарію використання UML;
- Змоделювати логіку складної процедури, функції чи операції;
- Побачити, як об'єкти та компоненти взаємодіють один з одним для завершення процесу;
- Планувати та розуміти детальну функціональність існуючого чи майбутнього сценарію.

На рисунку 2.6 представлена діаграма послідовності web-додатку.

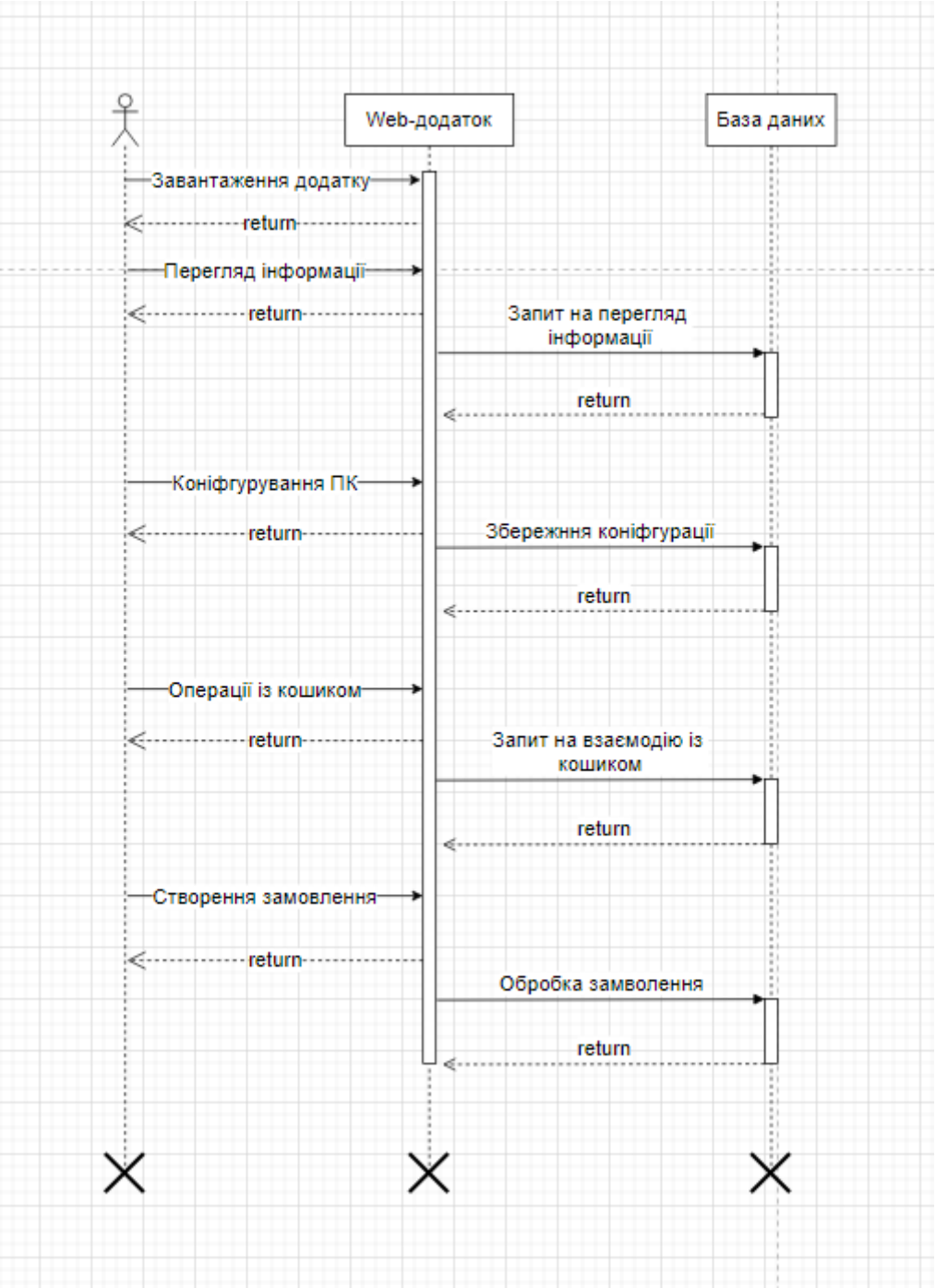


Рисунок 2.6 – Діаграма послідовності web-додатку

Для початку користувач завантажує додаток, а саме користувальницьку частину із серверу. Після чого він має можливість переглядати інформацію. Для перегляду інформації потрібно взаємодіяти із базою даних, після отриманих даних, можна їх переглядати на сайті. Коли буде завантажено товари, то можна почати конфігурацію ПК. Конфігурацію можна зберегти в БД для

зареєстрованого користувача. Коли комп’ютер буде сконфігуровано частково або повністю, його компоненти можна додати до кошику. Кошик буде також взаємодіяти із БД. Після чого можна створити замовлення, яке буде переглянуто менеджером. Коли замовлення буде підтверджено товар буде відправлено користувачу.

2.2.5 Діаграма компонентів web-додатку

Для створення конкретної фізичної системи необхідно реалізувати всі елементи логічного опису в конкретних матеріальних елементах. Для опису таких реальних елементів призначене фізичне подання моделі. У мові UML це означає сукупність зв'язаних елементів, включаючи програмне і апаратне забезпечення, а також персонал, для виконання спеціальних завдань. Фізичне подання моделей систем відбувається за допомогою діаграм реалізації. Використовуючи діаграму компонентів можна визначити архітектуру системи, яка розробляється. До базових графічних елементів діаграми компонентів входять її компоненти, інтерфейси і залежності між ними.

На рисунку 2.7 представлена діаграма компонентів web-додатку .

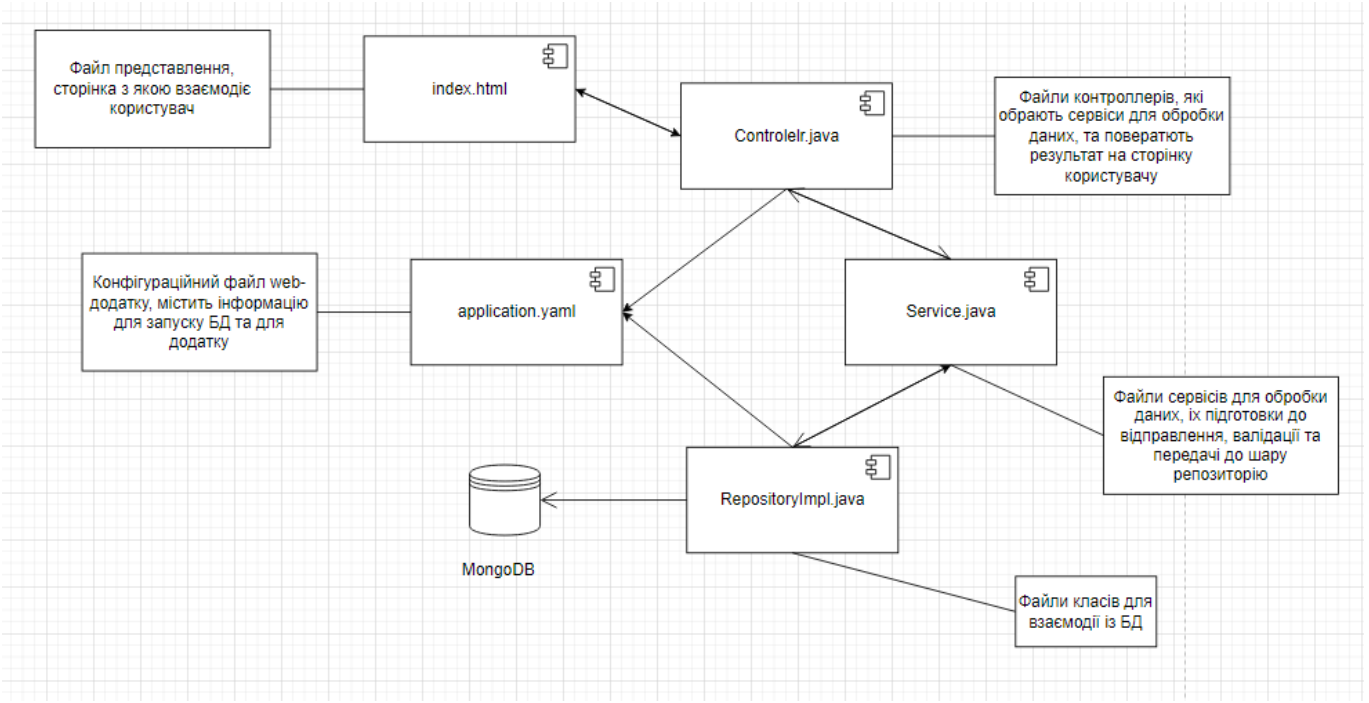


Рисунок 2.7 – Діаграма компонентів

2.2.6 Діаграма розгортання

У контексті уніфікованої мови моделювання (UML) діаграма розгортання відноситься до сімейства структурних діаграм, оскільки описує аспект самої системи. У разі діаграма розгортання визначає фізичне розгортання інформації, генерованої програмним забезпеченням, на апаратних компонентах. Інформація, яку створює програма, називається артефактом. Це не слід плутати з використанням цього терміну в інших підходах до моделювання, таких як BPMN.

Діаграми розгортання складаються з кількох фігур UML. Тривимірні коробки, відомі як вузли, являють собою основні програмні або апаратні елементи, або вузли, в системі. Лінії від вузла до вузла вказують на взаємозв'язки, а дрібніші фігури, що містяться всередині коробок, представляють програмні артефакти, що розгортаються [11].

На рисунку 2.8 представлена діаграма розгортання web-додатку.

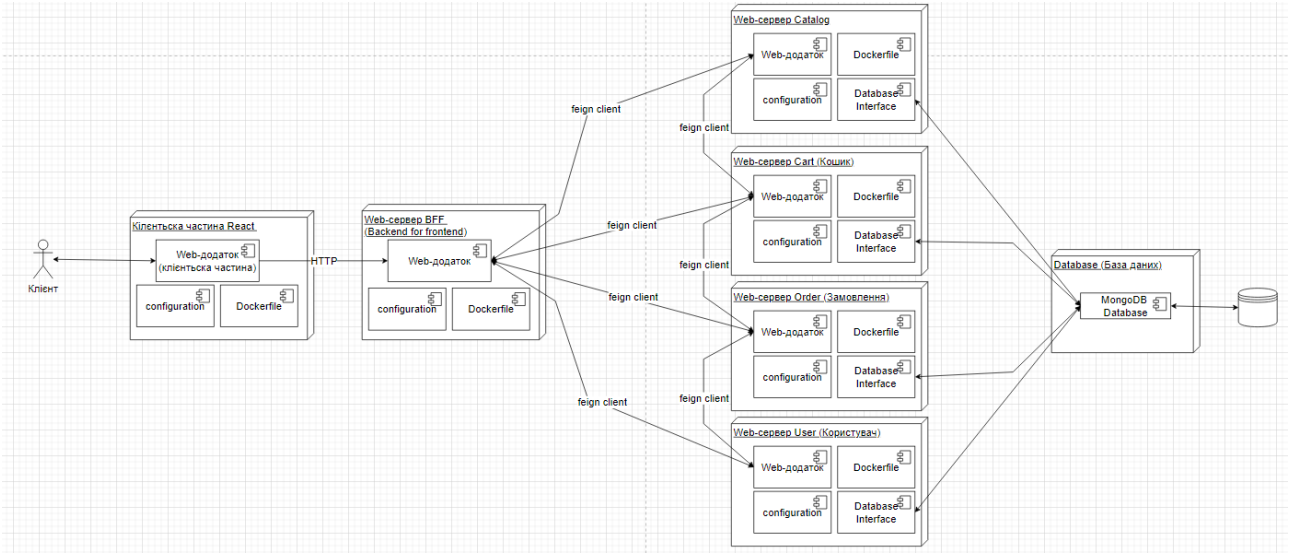


Рисунок 2.8 – Діаграма розгортання

В основі діаграми розгортання лежить 5 компонентів.

Перший компонент це клієнтська частина React комп’ютер, яка напряму працює із користувачами додатку.

Наступний компонент – це Backend for Frontend (BFF), він в собі містить скомпільований додаток, файл для збирання його в контейнер. Метод комунікації із користувачем HTTP/HTTPS. В свою чергу цей компонент взаємодіє із Catalog, Cart, Order, User сервісами, використовуючи технологію передачі даних – Feign Client.

Сервіси Catalog та Cart, Order, User містять в собі відповідні програми сервісів, файли для збирання в контейнер та сервіс, файл конфігурації та інтерфейс комунікації з БД.

Сервіс каталогу та сервіс кошику передають дані між собою, через технологію Feign Client. Сервіс кошику збирає інформацію про товари перед тим, як товари додаються до кошику, а сервіс каталогу, перевіряє компоненти на сумісність на основі товарів у кошику для відображення сумісних товарів в каталозі.

Сервіс замовлення пов'язаний із наступними сервісами:

- Кошик, використовується для формування замовлення на основі товарів у кошику;
- Користувач, використовується для пошуку першого вільного менеджера, котрий може прийняти замовлення.

Останній компонент діаграми – це База даних. Вона взаємодіє із сервісами, які потребують її для роботи додатку.

2.2.7 Діаграма потоків даних DFD

Діаграма потоку даних (DFD) відображає потік інформації будь-якого процесу чи системи. Вона застосовує набір символів: прямокутники, круги, стрілки, а також текстові позначки, щоб показати входи, виходи, точки зберігання даних та шляхи між пунктами призначення. Схеми потоків даних можуть бути простими, навіть намальованими від руки оглядів процесів до глибоких, багаторівневих DFD, які поступово заглиблюються в те, як обробляються дані. Їх можна використовувати для аналізу чи моделювання системи. Як і всі найкращі діаграми та графіки, DFD часто можуть візуально "сказати" те, що важко пояснити словами, і вони працюють як для технічної, так і для нетехнічної аудиторії, від розробника до генерального директора. Саме тому DFD залишаються такими популярними через стільки років. Хоча вони добре працюють для програмного забезпечення та систем з потоком даних, в даний час вони менш застосовні для візуалізації інтерактивного, орієнтованого на реальний час або бази даних програмного забезпечення або систем [12].

На рисунку 2.9 показано діаграму потоку даних нульового рівня, а на рисунку 2.10 показано DFD першого рівня.

Рисунок 2.9 – Дігамма потоку даних нульового рівня

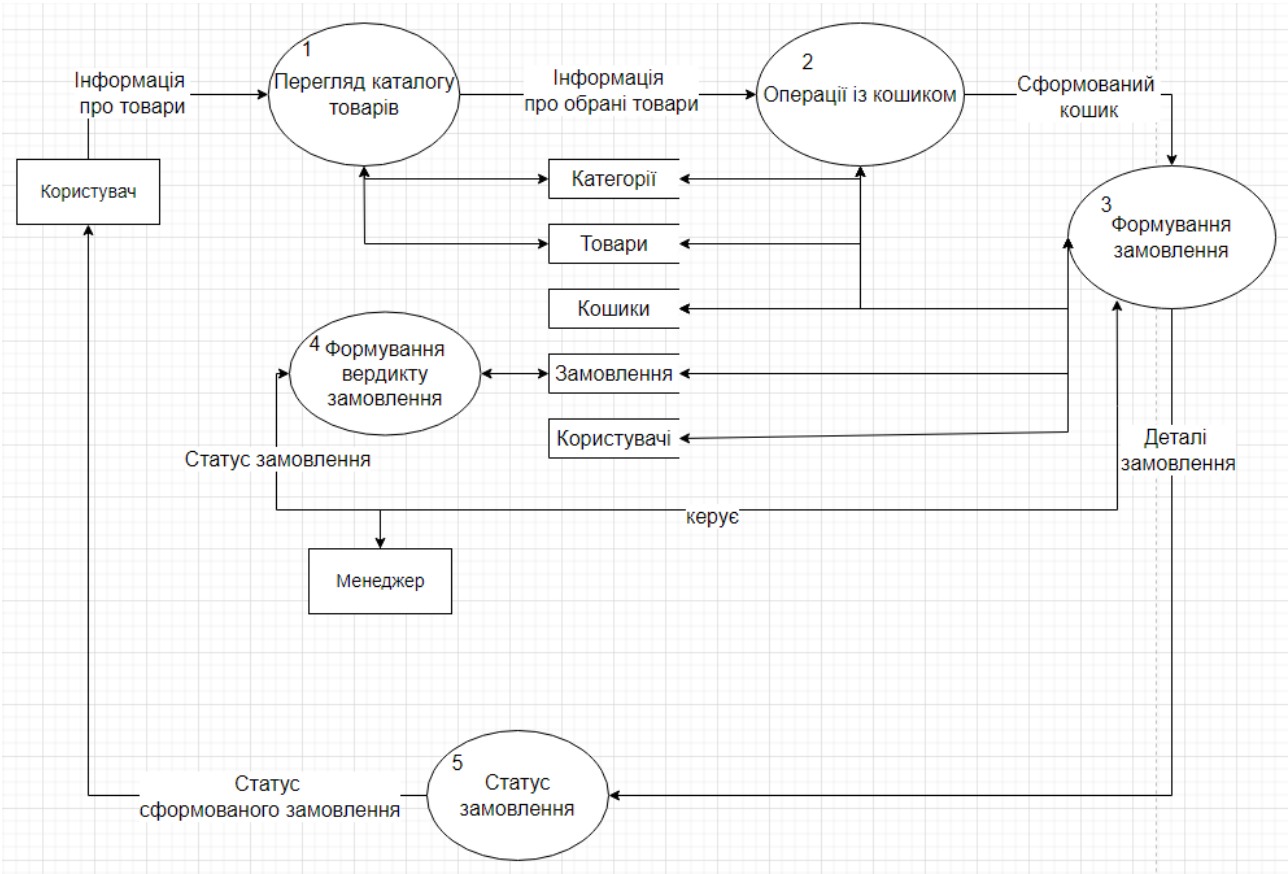


Рисунок 2.10 – Діаграма потоків даних web-додатку

Дана схема показує потік даних. Користувач конфігурує ПК, де дані про товари беруться з БД. Після чого деталі замовлення ідуть в компонент формування замовлення, де замовлення керується модератором всі записи ідуть

до БД. Де дані потім зберігаються. Модератор в цей час може управляти потоком даних пов’язаних із товарами.

2.3 Проектування моделі бази даних

Бази даних NoSQL (вони ж "як не лише SQL") є нетабличними базами даних і зберігають дані по-іншому, ніж реляційні БД. Бази даних NoSQL поділяються на різні типи в залежності від моделі даних. Основними типами є документ, граф, широкий стовпець та ключ-значення. Ці типи забезпечують гнучкість схеми і легке масштабування при великих обсягах даних і високому навантаженні користувача [13].

У випадку використання MongoDB дані представляються у форматі JSON. JSON (JavaScript Object Notation) – є простим форматом обміну даних. Його легко читати, формувати та структурувати дані. Машини легко розбирають та генерують його. JSON – це текстовий формат, який не залежить від мови програмування, але використовує знайомі стандарти для програмістів сімейства мов C. Ці характеристики, притаманні JSON роблять його ідеальним стандартом для обміну даними [14].

NoSQL, на відміну SQL, у якому є ER- і діаграми класів, немає імен, ні обмежень для діаграм моделювання даних. Очевидна причина цього – пом'якшені правила NoSQL щодо відносин, які спрямовані на те, щоб розробник розпочав роботу з мінімальними вимогами [15].

Засіб реалізації та відсутність чітких визначеної схеми не дозволяє показати модель даних. Даний засіб реалізації був обраний через те що дані мають різну структуру, і тому для полегшення процесу роботи із нечіткими даним, які важко описати схемою, використовується документо-орієнтована база даних MongoDB. Приклади документів можна побачити на рисунку 2.11.

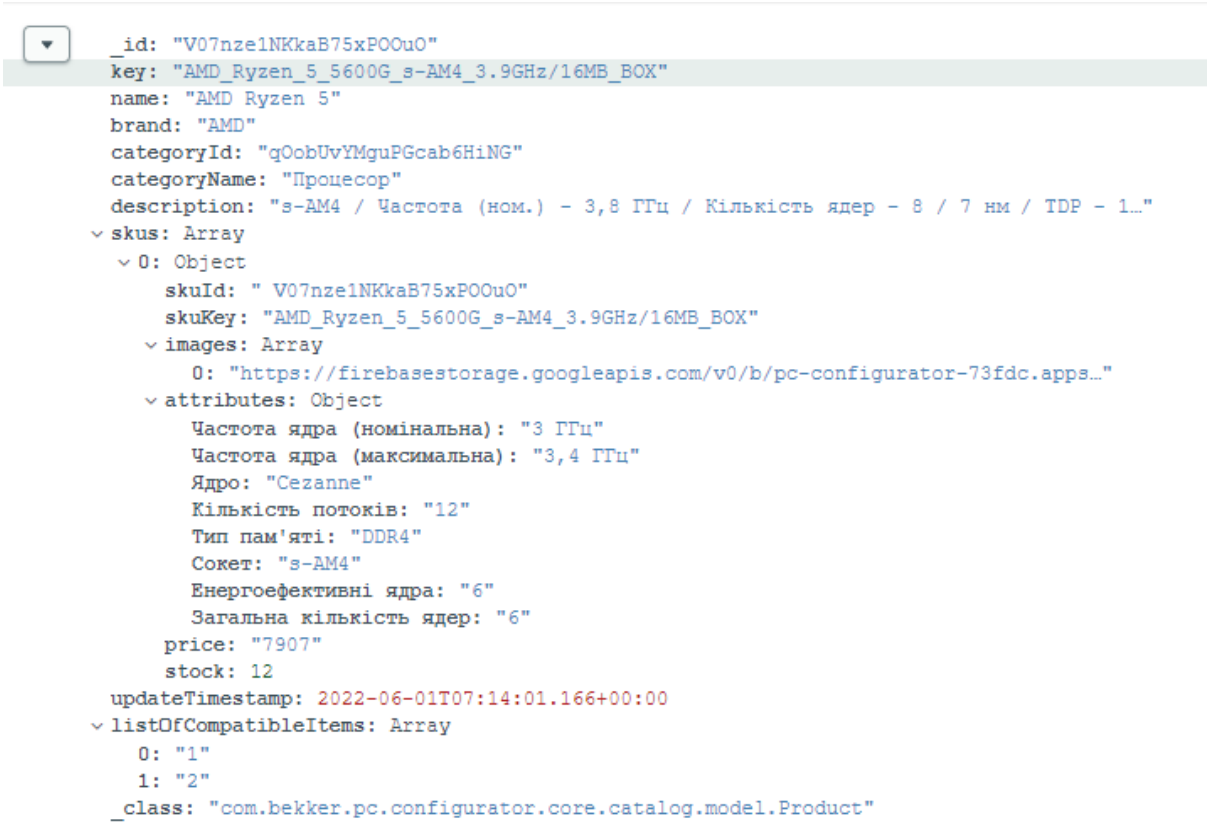


Рисунок 2.11 – Приклад документу для нереляційної бази даних

3 РОЗРОБКА WEB-ДОДАТКУ

3.1 Архітектура web-додатку

3.1.1 Загальна архітектура web-додатку

Для розробки web-додатку було обрано мікросервісну архітектуру. Суть якої полягає у розділенні додатку на сервіси, які будуть виконувати функціональні можливості додатку, але будуть розподілені, та будуть вести комунікацію між собою. Даний підхід дозволяє, покращити стабільність роботи додатку, у випадках коли один із сервісів може зламатися через пошкодження на сервері тощо. Також такий підхід дозволяє розширювати функціональну базу ресурсу, замість його модифікації. Це зроблено з метою отримання користувачами додатку незміненої поведінки відповідей від сервера за тими ж самими вхідними параметрами.

3.1.2 Архітектурний стиль взаємодії із серверною стороною

Також під час розробки було обрано архітектурний стиль взаємодії клієнтської частини із серверною під назвою - REST API. Даний стиль надає гнучкі функціональні можливості для вирішення задач на сервері.

API (Application Programming Interface) – це сукупність визначень та протоколів для створення та інтеграції прикладного програмного забезпечення. Інша його назва – контракт між користувачем та постачальником інформації, що встановлює контекст, згідно вимоги користувача (запит), і змісту постачальника (відповідь) [16].

REST являє собою набір архітектурних обмежень, а не протокол чи стандарт. REST може бути реалізованим у різний спосіб. Коли клієнтський запит виконується за допомогою RESTful API, він передає уявлення про стан ресурсу запитуючому або кінцевій точці. Ця інформація або представлення передається

в одному з найбільш популярних форматів за протоколом HTTP: JSON (Javascript Object Notation) [16].

Всі запити та відповіді на сервері побудовані відносно REST Maturity Model (модель зрілості). API у даному проекті побудований на основі трьох принципів із чотирьох, які описані у REST Maturity Model.

Для розуміння моделі зрілості для REST, ключові рівні будуть порівнюватися із використаною у проекті моделлю.

API REST не визначає фіксовані імена ієрархії чи ресурсів (звичайний зв'язок сервера з клієнтом). На сервері повинна бути можливість контролю простору власних імен. Натомість, необхідно дозволити серверам надавати інструкцію клієнтам щодо побудови відповідних URI (Unifor Resource Identifier, єдиний ідентифікатор ресурсів), як це роблять при побудові форм HTML та шаблонів URI, таким чином визначаючи ці інструкції у типах медіа та відносних посиланнях [17].

Вхід у REST API повинен здійснюватися без будь-яких попередніх знань, крім початкового URI (закладки) та набору стандартизованих типів носіїв, які підходять для цільової аудиторії (тобто очікується, що вони будуть зрозумілі будь-якому клієнту, який може використовувати API). З цього моменту всі переходи стану програми повинні бути обумовлені вибором клієнтом наданих сервером варіантів, які є в отриманих уявленнях або маються на увазі в результаті маніпуляцій користувача з цими уявленнями [17].

Модель зрілості REST Річардсон описує чотири різних рівня REST (починаючи з рівня 0). API REST, що підтримує елементи управління гіпермедіа, класифікується як рівень 3 цієї моделі зрілості [17].

Level Zero (нульовий рівень):

Нульовий рівень зрілості не використовує жодну з можливостей URI, HTTP-методів та HATEOAS. Служби на нульовому рівні зрілості мають єдиний URI та використовують єдиний метод HTTP (зазвичай POST). Наприклад, більшість веб-служб SOAP використовують один URI для ідентифікації кінцевої

точки та HTTP POST для передачі корисного навантаження на основі SOAP, фактично ігноруючи інші дієслова HTTP. Аналогічно, послуги на основі XML-RPC передають дані у форматі Plain Old XML (POX). Це найпримітивніші способи побудови SOA-додатків з єдиною кінцевою точкою методу POST та використанням XML для зв'язку між клієнтом та сервером [17].

Коли це зручно, у шляхах URI краще використовувати малі літери, оскільки великі літери іноді можуть викликати проблеми. RFC 3986 визначає URI як чутливі до регістру, крім компонентів схеми і хоста [18].

API REST не повинен містити штучні розширення файлів в URI для вказівки формату тіла сутності повідомлення. Натомість вони повинні покладатися на тип медіа, що передається через заголовок Content-Type, щоб визначити, як обробляти вміст тіла [18].

У випадку додатку всі URI для запиту відповідають правилам нульового рівня моделі зрілості. Ось приклади деяких із запитів:

http://localhost:8081/users/{id}
http://localhost:8081/users
http://localhost:8081/users/accounts/details
http://localhost:8081/users/login

Level One (перший рівень):

На першому рівні зрілості використовуються URI, але не використовуються методи HTTP та HATEOAS. Сервіси першого рівня зрілості використовують безліч URI, але тільки один HTTP дієслово - зазвичай HTTP POST. Ці послуги надають кожному ресурсу, доступному в системі, унікальний URI. Унікальний URI окремо ідентифікує один унікальний ресурс – і це робить ці сервіси кращими, ніж сервіси нульового рівня [17].

Це одне з найважливіших правил, яке необхідно дотримуватись, оскільки останній символ у шляху URI, слеш (/), не додає семантичної цінності і може

викликати плутанину. API REST не повинні очікувати наявності прямої косої межі і не повинні включати її в посилання, які вони надають клієнтам [18].

Кожен символ URI має значення для унікальної ідентифікації ресурсу. Два різні URI вказують на два різні ресурси [18].

Для позначення ієрархічних відносин слід використовувати слеш характеристика (/). Символ слешу (/) використовується у частині шляху URI для вказівки ієрархічних відносин між ресурсами [18].

У випадку додатку всі URI для запиту відповідають правилам першого рівня моделі зрілості. Ось приклади деяких із запитів:

http://localhost:8081/catalog/products
http://localhost:8081/catalog/categories
http://localhost:8081/catalog/categories/compatible/{category id}
http://localhost:8081/catalog/products/search
http://localhost:8081/catalog/products/filter

Level Two (другий рівень):

На другому рівні зрілості використовуються URI та методи HTTP, але не використовується HATEOAS. Послуги другого рівня зазвичай містять багато URI, тобто. адресованих ресурсів. Такі послуги підтримують кілька дієслів HTTP кожному відкритому ресурсі – послуги Create, Read, Update і Delete (CRUD). Тут можна маніпулювати станом ресурсів, які зазвичай представляють бізнес-сутності, через мережу. Розробники сервісів другого рівня очікують, що люди докладуть деяких зусиль для освоєння API – як правило, читання документації, що додається. Рівень зрілості 2 є найпопулярнішим прикладом використання принципів REST, які передбачають використання різних дієслів залежно від методів запиту HTTP, тоді як система може мати кілька ресурсів [17]. Нижче перераховані методи/дії, що часто використовуються (табл. 3.1):

Таблиця 3.1 – Методи HTTP та їх опис

Метод	Область застосування	Семантика
GET	колекція	Отримання всіх ресурсів у колекції
GET	ресурс	Вилучення одного ресурсу
HEAD	колекція	Отримання всіх ресурсів у колекції (лише заголовок)
HEAD	ресурс	Вилучення одного ресурсу (тільки заголовок)
POST	колекція	Створити новий ресурс у колекції
PUT	ресурс	Оновлення ресурсу
PATCH	ресурс	Оновлення ресурсу
DELETE	ресурс	Видалити ресурс
OPTIONS	будь-який	Повернення доступних методів HTTP та інших опцій

У випадку web-додатку всі URI для запиту відповідають правилам другого рівня моделі зрілості. Ось приклади деяких із запитів:

POST - http://localhost:8081/orders (Створення нового замовлення).

GET - http://localhost:8081/orders (Отримання замовлення).

DELETE - http://localhost:8081/orders (Видалення замовлення).

PUT - http://localhost:8081/users (Оновлення даних про користувача).

GET - http://localhost:8081/catalog/products (Отримати колекцію всіх продуктів).

Level Three (третій рівень):

Третій рівень зрілості використовує всі три, тобто. URI, HTTP та HATEOAS. Третій рівень – це найбільш зрілий рівень моделі Річардсона, який заохочує легке виявлення. На цьому рівні відповіді легко стають самоописовими завдяки використанню HATEOAS. Послуги третього рівня ведуть споживачів послуг за слідом ресурсів, викликаючи в результаті переходи стану програми [17].

В REST API, який було спроектовано та розроблено не має третього рівня зрілості. Це було зроблено в рамках того, що при розробці сервісів велася документація всіх вхідних точок, та їх використання. Але в майбутній версії API буде додано цей рівень, для покращеного користування інтерфейсом.

3.1.3 Архітектура та структура окремого сервісу

Тепер коли було обрано архітектуру всієї інтеграції та було обрано архітектурний підхід для взаємодії із серверною стороною, настав час спроектувати архітектуру кожного із сервісів. Додаток має 5 сервісів, що використовують REST API (серверна сторона додатку) та 1 сервіс, який на пряму взаємодіє із клієнтом.

Кожен сервіс спроектовано відповідно до DDD (Domain Driven Design) та MVC (Model-View-Control). DDD – це патерн, який зазначає, що модуль повинен бути розділений за бізнес процесами із якими він працює. Це означає, що сервіси розділені між собою за бізнес потребами, які виконуються. Для прикладу розглянемо сервіс замовлення. В ньому описано лише ті функціональні можливості, які відносяться до замовлення (створення замовлення, його видалення, та отримання). MVC – використовується наступним чином. Всі

сервіси, мають структуру модель-представлення-контролер. Де контролер відповідає за прийняття запиту та відправлення відповіді клієнту. Моделі – це бізнес об’єкти, які зберігають дані, та сервісний рівень який обробляє ці дані. Та рівень представлення, туди входять моделі запитів та моделі відповідей. Розділення шарів за патерном MVC можна побачити на рисунку 3.1:

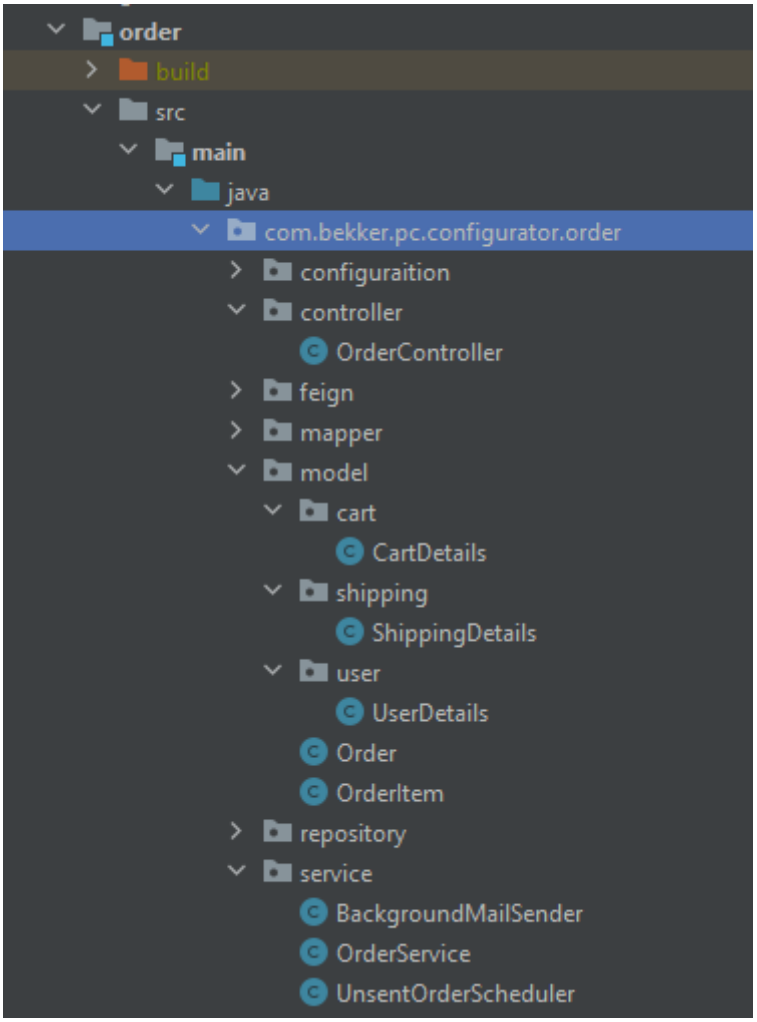


Рисунок 3.1 – Структура проекту відповідно то патерну MVC

Серверні сервіси зв’язані декларативним клієнтом web-сервісу Feign Client. За його допомогою сервіси можуть передавати інформацію між собою. Основний сервіс який зв’язує всі – це BFF (Backend For Frontend). Цей сервіс слідує патерну проектування Front Controller. Тобто всі запити, які приходять на сервер, спочатку проходять через нього. Після чого BFF викликає потрібний сервіс. Той сервіс в свою чергу приймає запит (модифікований або ні) та

обробляє його, на виході цей сервіс надсилає відповідь до BFF, після чого BFF надсилає відповідь клієнту. Приклад взаємодії BFF із сервісом та подальшу комунікацію із API можна побачити на рисунку 3.2.

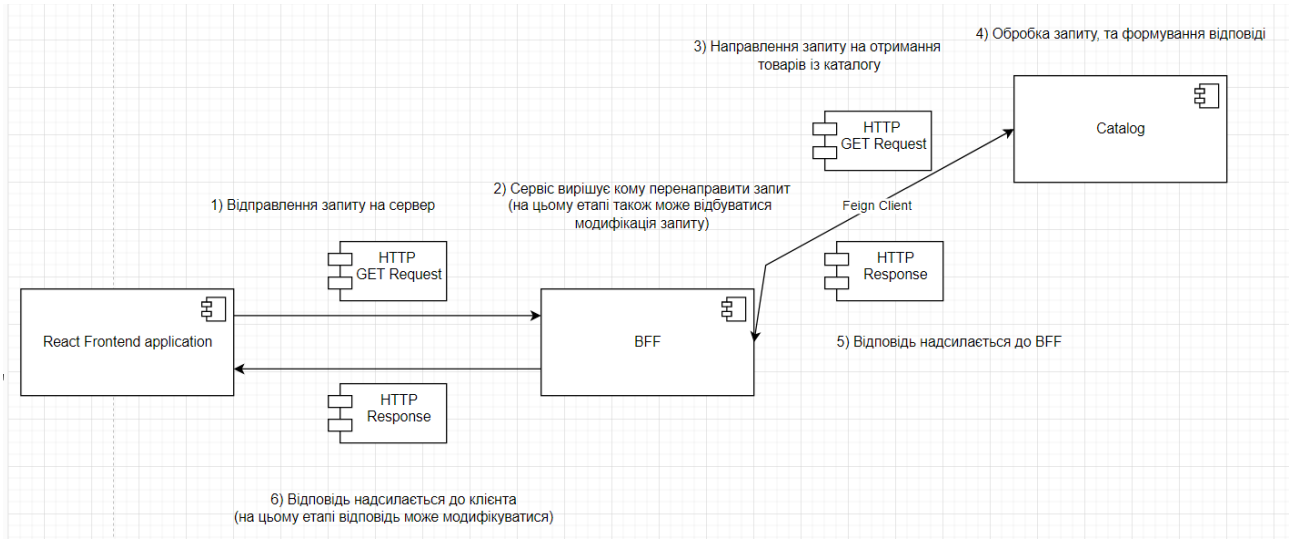


Рисунок 3.2 – Взаємодія сервісу BFF із Catalog. Опис потоку подій під час звернення до серверу

3.2 Програмна реалізація

3.2.1 Програмна реалізація серверної сторони web-додатку

Для розробки серверної частини web-додатку було обрано мову програмування Java, і додатково буде використовуватися фреймворк Spring Boot. Цей фреймворк полегшить роботу написання сервісів, і дозволить зробити максимальний акцент на написання бізнес рішень. Сервіс формується із декількох обов’язкових складових:

- Конфігурація додатку;
- Шар контролерів;
- Сервісний шар;
- Шар репозиторію.

- Шар бізнес моделей.
- Шар ремапінгу (видозміна даних, для відділення бізнес моделей та DTO).

Конфігурація додатку містить інформацію, яка може змінюватися в залежності від параметрів, які туди будуть передані. Таким чином можна налагоджувати роботу web-додатку в залежності від різних ситуацій. До прикладу в сервісі BFF використовуються змінні середовища, які контролюють назву заголовку JWT токена, який використовується для аутентифікації та авторизації. Ще один приклад сервіс замовлення, він містить змінну, яка контролює через який час не оброблені замовлення будуть відправлятися повторно. Таким чином можна доволі гнучко конфігурувати сервіс додатку.

Шар контролерів відповідає за прийняття та подальшу передачу на обробку даних запиту. Від клієнта ми отримуємо HTTP запит, який містить:

- URI (endpoint) - точка входу;
- Headers (заголовки);
- Parameters (параметри);
- Body (тіло запиту).

Контролер за допомогою mappings(зіставлення точки входу із вказаним шаблоном) визначає, який метод використати для обробки запиту. Далі цей запит передається до сервісного шару.

Сервісний шар відповідає за основну логіку роботи із даними. Тут відбуваються наступні операції:

- Витягування параметрів, та їх валідація;
- Запити до репозиторію та інших сервісів для отримання додаткової інформації;
- Бізнес правила, які обробляють дані;
- Обробка інформації (фільтрація, трансформація тощо);
- Виклики перетворення даних для відправки клієнту;
- Формування помилок;
- Ведення аудиту web-додатку.

Після обробки даних, та їх формування для відправлення клієнту. Дані знову повертаються до контролеру, а звідти повертаються до клієнта, який надіслав запит.

Шар репозиторію, працює із даними. Передані параметри від сервісного шару, потрапляють сюди та надсилаються до бази даних, або PIM системи, або до пошукового двигуна. Репозиторій це патерн проектування, який описує основні методи роботи із джерелом даних. Таким чином ця абстракція дозволяє використовувати будь-яке джерело без зміни сервісного рівня. Такий шаблон – інтерфейс описує правила вхідних та вихідних параметрів за допомогою сигнатури. Таким чином сервіс, має слабку залежність відносно до рівня репозиторію, тому його можна легко підмінити на іншу реалізацію. Як приклад в проекті використовується три різновиди репозиторіїв:

- Репозиторій, який працює із базою даних. До речі цей підвид, також може урізноманітнюватися різними базами даних;
- Репозиторій, який працює із PIM системою. Це 3d-party залежність, яка працює із наповнення каталогу товарами та категоріями Google Firebase;
- Репозиторій, який працює із Search Engine (пошуковий двигун) Algolia Search.

Шар бізнес моделей представляє собою, об’єкти, що зберігають в собі дані із джерел інформації та можуть піддаватися обробці.

Шар ремапінгу відповідає за відділення логіки бізнес моделей та представлень, які будуть надсилатися до клієнту. Під час надсилання запиту до серверу дані формуються у так звані DTO (Data Transfer Object), такі моделі несуть в собі сирі не оброблені дані (якщо говорити про запит). Такі дані потребують валідації на серверній стороні. Шар ремапінгу дістає після валідації переносить дані із об’єкта представлення DTO на бізнес модель. Та у випадку відправки даних до клієнта також використовуються DTO, це потрібно для того, щоб додати нові поля, які не потрібні бізнес моделям, та видалити поля, які краще не показувати назовні. Наприклад бізнес модель списку продукції не

повинна містити інформації про кількість сторінок, номер сторінки, варіант сортування, параметр сортування і таке інше. Це дані які потрібні для клієнта, але аж ніяк не повинні фігурувати в бізнес моделі.

Вище описані шари використовуються в кожному із сервісів, деякі із сервісів додають шари для комунікації із іншими сервісами та шари для утильних модулів.

Також важливим компонентом у сервісі додатків є автоматизована збірка. Для цього використовується популярний build tool – Gradle. Його суть полягає у описі декларативних команд, які будуть виконувати наступні операції:

- підтягування залежностей та бібліотек із інтернет репозиторіїв;
- запуск стадії тестування;
- запуск стадії компіляції та збирання проекту для подальшого його запуску;
- запуск сформованого додатку.

Процес тестування додатку є важливою частиною перевірки його роботоспроможності та виявлення можливих проблем та недоліків у майбутньому. Тестування проводиться за допомогою фреймворку Junit. У випадку тестування сервісів важливо зробити максимальний акцент на сервісний шар. Адже саме там відбуваються операції по обробці даних. Модульні тести включають в себе перевірку лише одної мінімальної функціональної одиниці. Таким чином при хорошому проектуванні, можна сказати, що перевірка одного методу сервісного класу і відповідає критеріям модульного тестування. Процес тестування включає в себе перевірки правильності роботи методів, та можливі виключні дані. Це означає, що важливо перевірити ці методи у нормальних умовах, та у виключних умовах. Наприклад, дані від клієнта були передані не вірно і таке інше.

3.2.2 Програмна реалізація клієнтської сторони web-додатку

Клієнтська частина додатку написана за допомогою фреймворку React.js. Суть якої полягає у створені компонентів, які можуть використовуватися повторно. Механізми, які впроваджені у фреймворк, надають можливості для розробки гнучких клієнтських додатків. Таким чином важливою частиною при проектуванні клієнтської частини додатку є розбиття на компоненти. Web-додаток клієнтської частини представлено наступною структурою:

- конфігурація;
- стилі;
- картинки;
- компоненти;
- контекст;
- утильні модулі;
- бізнес об’єкти;
- головний компонент.

Конфігурація включає в себе файл із підтягуванням всіх потрібних залежностей (бібліотек) та етапи запуск проекту.

Стилі в собі містять стилі оформлення основних компонентів. Також там описуються анімації та адаптації для коректного відображення на моніторах.

Компоненти поділяються на папки в яких кожна папка відповідає за екран, або логічно пов’язана із іншими компонентами. Наприклад компоненти кошику знаходяться в одному пакеті, але використовуються на різних екранах. А от каталог містить лише ті компоненти, які відносяться лише до нього.

Контекст описує область в якій будуть передаватися дані. Тобто всі компоненти, які знаходяться в області контексту можуть ділитися даними із усіма іншими компонентами у цьому контексті. Таким чином можна передавати дані між компонентами, які не мають прямої залежності один із одним.

Утильні модулі містять в собі скрипти, які можуть повторно використовуватися в компонентах. Це зроблена для того, щоб знизити дуплікацію коду.

Бізнес об’єкти в собі містять поля, які будуть зберігати дані, та передаватися між компонентами.

Головний компонент, він працює по аналогії головної сторінки на сайті. Тобто він включає в себе всі компоненти, які будуть використовуватися на екранах. Також там описується модель переходу між сторінками та їх послідовність у відображені.

3.2.3 PIM система Google Firebase

В якості PIM системи було вирішено використовувати документо-орієнтовану базу даних.

Інструмент PIM – це програмне рішення, яке допомагає роздрібним торговцям, оптовикам та виробникам контролювати свої продукти у централізованому місці. Він забезпечує єдину точку істини продукту (і всіх його даних). Це місце для збору, оптимізації та розповсюдження даних про вміст продукту. У певному сенсі це центральний інструмент для керування даними про продукт, хоча він пропонує набагато більше [19].

Завдяки такій системі наповнення контенту у додатку стає простою роботою для менеджерів. Все ж менеджеру потрібні знання JSON, але це не складно вивчити, або просто робити по прикладу.

Консоль додатку для додавання товарів виглядає наступним чином (рис. 3.3).

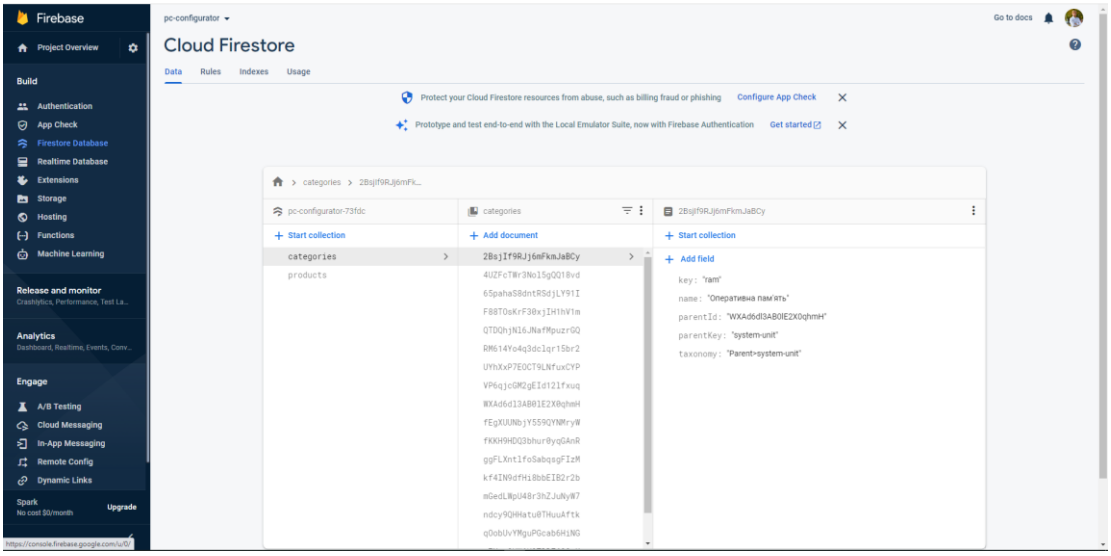
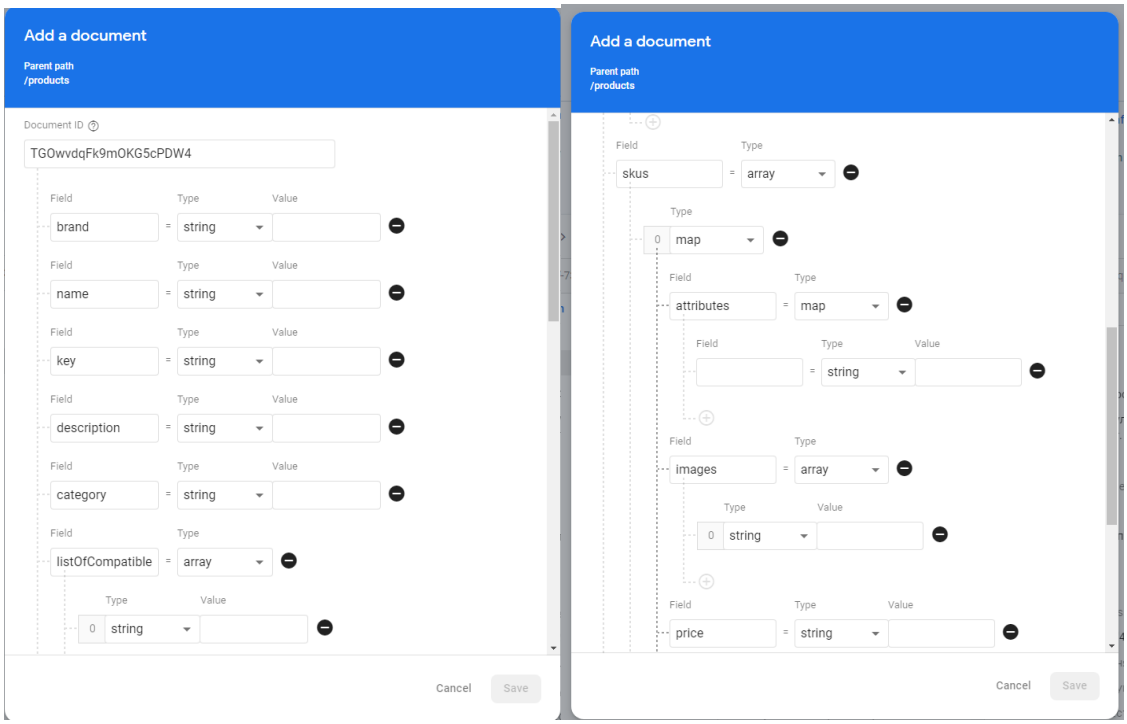


Рисунок 3.3 Вигляд консолі для додавання контенту для web-додатку

Для створення документу достатньо просто клікнути на add document та створити поля товару чи категорії, або взагалі нового бізнес об’єкту (рис 3.4- 3.5).



Рисунки 3.4-3.5 Модальне вікно для створення бізнес об’єкту в Cloud Firestore

Цей інструмент дозволяє проводити всі CRUD операції відносно даних. Таким чином можна додавати, видаляти, змінювати та читати дані із цієї PIM системи.

Також важливим для web-додатку було питання зберігання картинок товарів. Для вирішення цієї задачі оптимальним було вирішення використовувати хмарне сховище на цій же платформа Google Firebase. Приклад організації картинок можна побачити на рисунках 3.6-3.7.

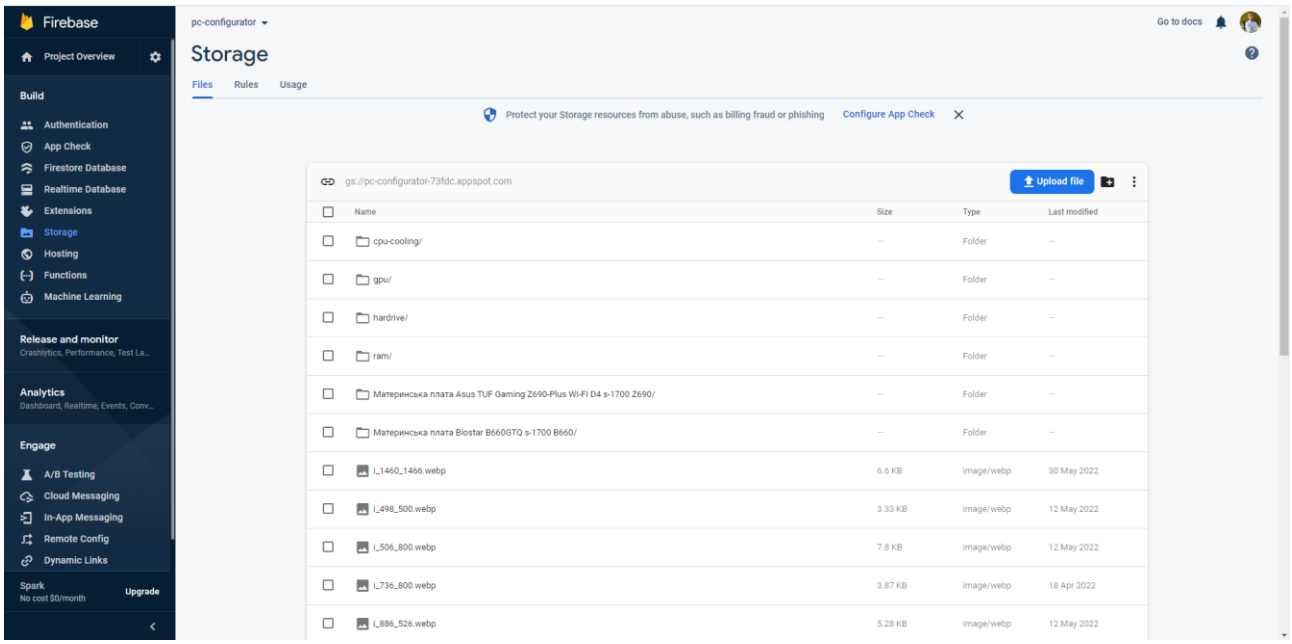


Рисунок 3.6 – Вигляд панелі консолі для зберігання картинок товарів

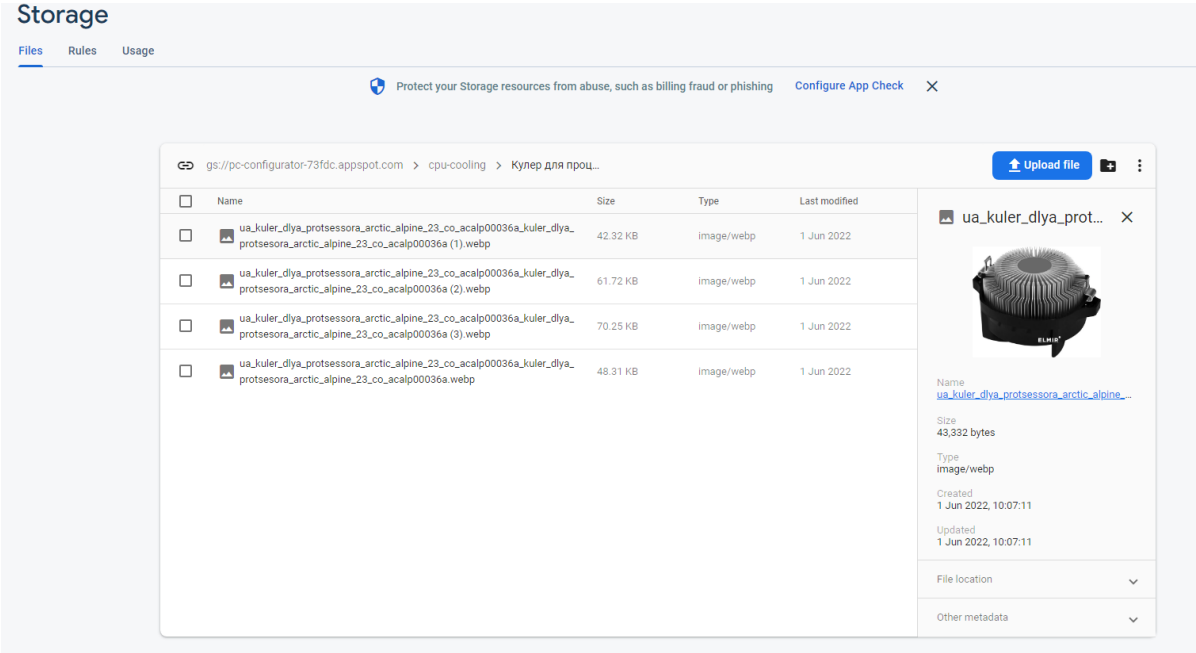


Рисунок 3.7 Вигляд медіа файлів завантажених до хмарного сховища

Після завантаження медіа файлу на хмару, до цього файлу можна отримати доступ за посиланням. Таким чином можна налаштувати зберігання фото для товарів наступним чином (рис. 3.8).

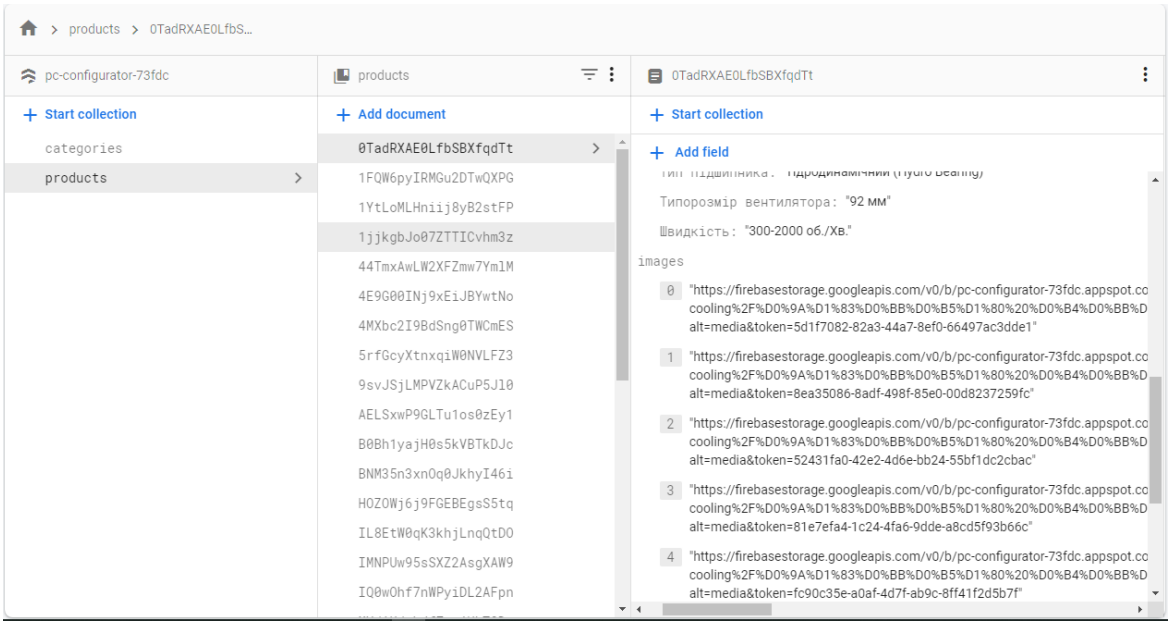


Рисунок 3.8 – Додавання картинок до товару, які зберігаються на хмарі

3.2.4 Пошуковий двигун Algolia Search Engine

Важливою частиною при роботі із товарами є оптимізація та швидкість. Для того щоб надати швидкості для таких операцій як пошук за ключовими словами, фільтрація, сортування та інші операції над даними, в проекті використовується Algolia Search API. Для конфігурування пошукових полів, та фільтрів було написано наступний метод:

```
@Override
public void setIndexSettings(List<String> facetsAttributes) {
    productSearchIndex.setSettings(
            new IndexSettings()
                    .setSearchableAttributes(Arrays.asList(
                            "objectID",
                            "name",
                            "brand",
                            "description",
                            "categoryId",
                            "categoryName",
                            "skus.attributes"
                    )).setAttributesForFaceting(facetsAttributes),
            SET_FORWARD_TO_REPLICAS
    );
}
```

Цей метод індексує всі поля, які повинні бути пошуковими. Також цей метод приймає поля, які будуть використовуватися для того щоб проводити фільтрацію. В цьому ж методі ми оновлюємо всі ці налаштування на репліки колекцій даних.

Стандартні репліки пов'язані зі своїм первинним індексом: стандартна репліка копіює вміст первинного індексу, але може мати інші налаштування. Хоча стандартна репліка пов'язана зі своїм первинним індексом, вона є окремим індексом. Це означає, що стандартні репліки збільшують кількість записів у вашому додатку. Стандартні репліки мають такі самі дані, як і їх первинний індекс, і Algolia виконує синхронізацію даних за вас. Коли ви додаєте, оновлюєте або видаляєте записи в первинному індексі, ваша репліка автоматично змінюється. Ви не можете змінити автоматичну синхронізацію даних. Наприклад, ви не можете безпосередньо додавати, оновлювати або видаляти

записи в індексах репліки. Стандартні репліки запускаються з тими ж налаштуваннями, що їх первинний індекс. Однак можна змінити налаштування репліки. Найбільш поширеними змінами налаштувань репліки є зміни стратегії вичерпного сортування [20].

Панель Algolia Search Engine можна побачити на рисунку 3.9.

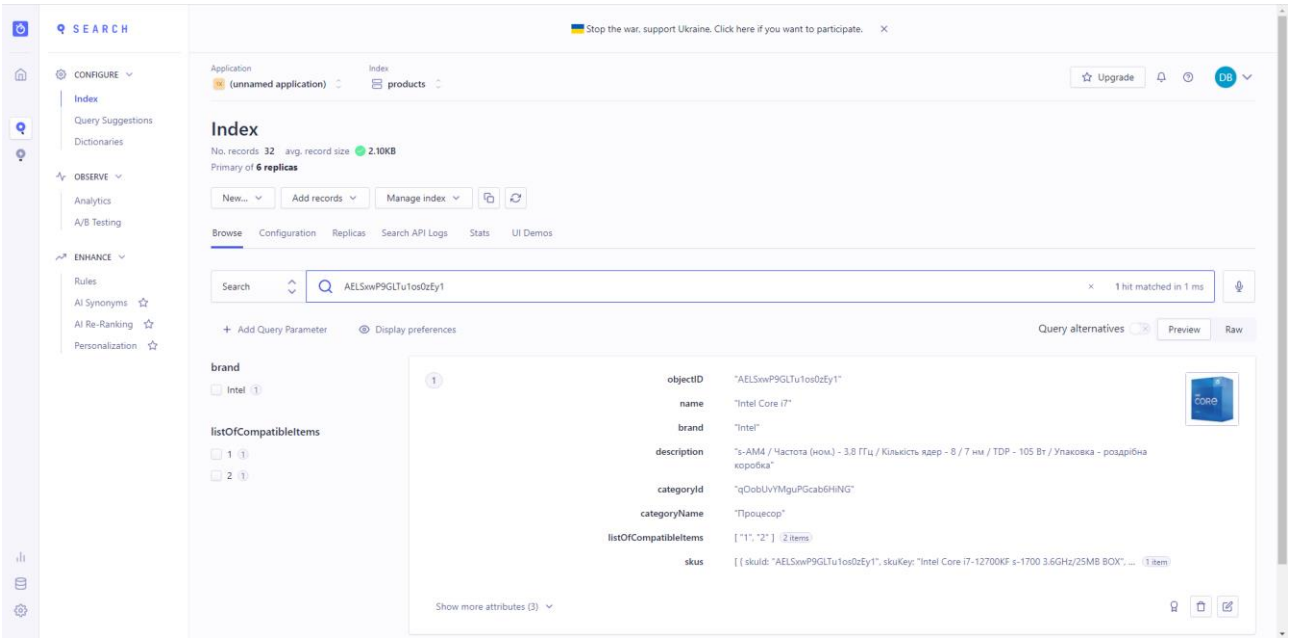


Рисунок 3.9 – Панель консолі для конфігурування пошукового двигуна

Для конфігурування реплік також використовується консоль пошукового двигуна. Репліки для сортування полегшують навантаження на пошуковий двигун. Заздалегідь сформовані репліки вже відсортовані по параметрам та відповідно до параметру послідовності (від більшого до меншого, та навпаки). Репліки для основного індексу товарів можна побачити на рисунку 3.10.

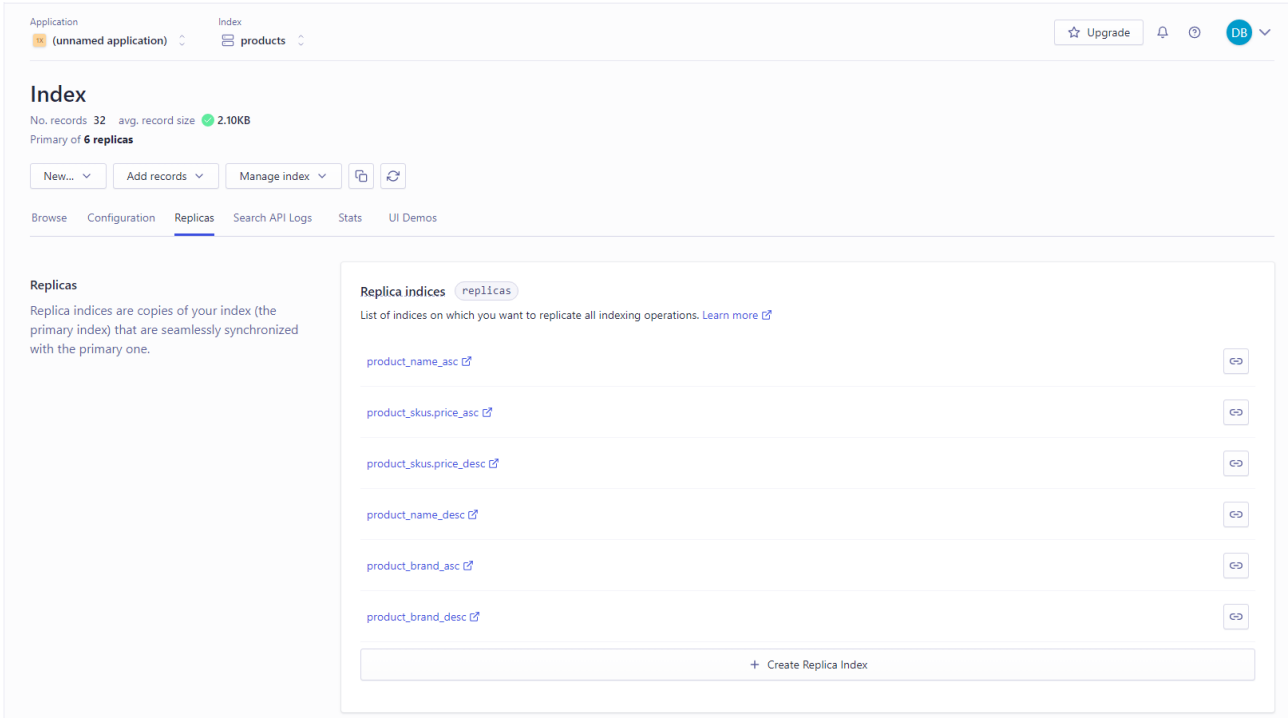


Рисунок 3.10 – Репліки для сортування налаштовані відносно основного індексу

Налаштовані репліки виглядають наступним чином: атрибути фасетів для фільтрації товарів можна побачити на рисунку 3.11

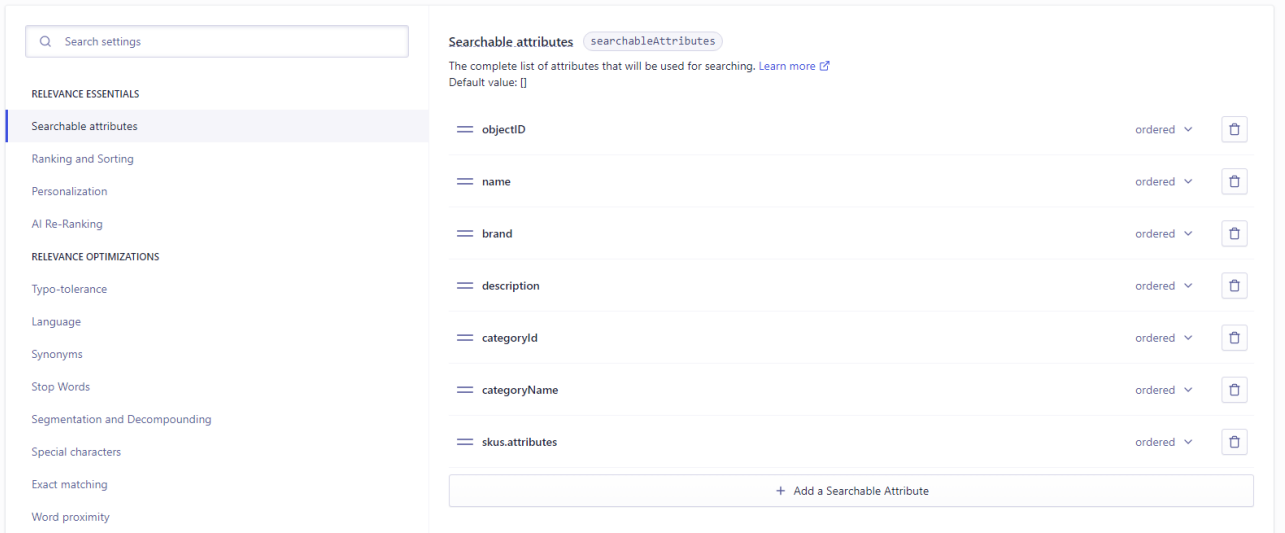


Рисунок 3.11 – Налаштовані атрибути фасетів для фільтрації товарів для реплік

3.3 Використання web-додатку

При відкритті сайту, перше, що можна побачити на екрані – це основні категорії, для конфігурування комп’ютера (рис 3.12).

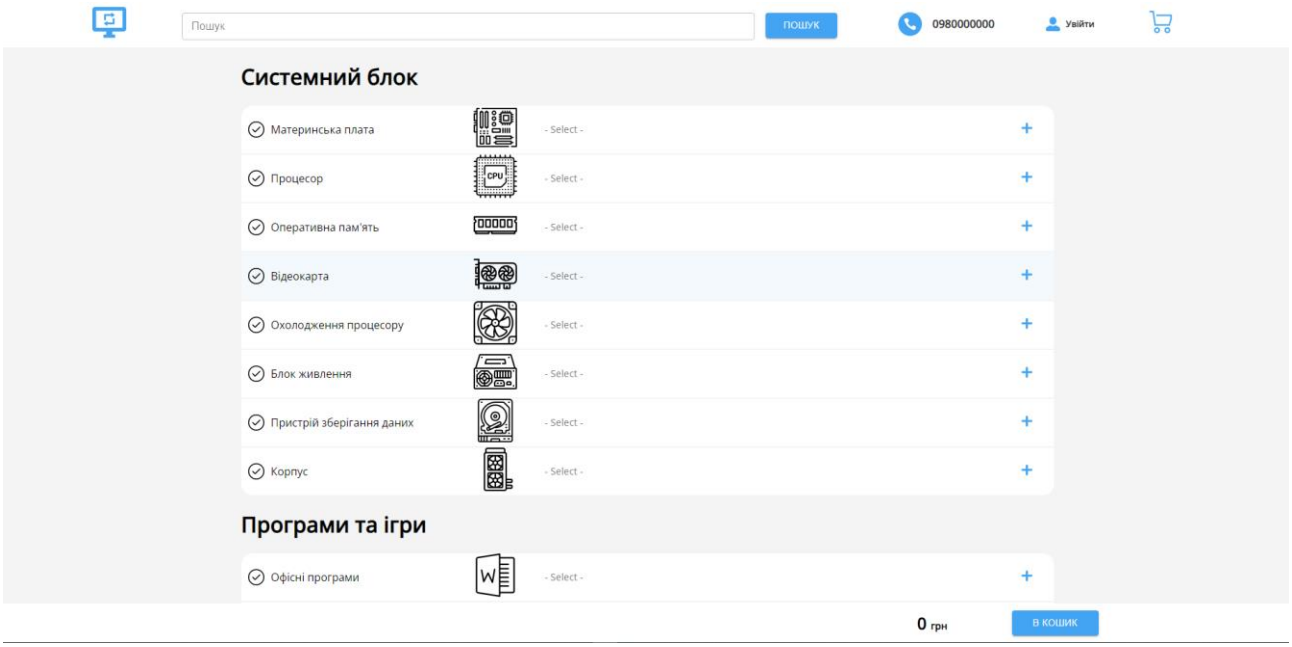


Рисунок 3.12 – Вигляд web-додатку після його запуску

Далі користувач може обрати будь-яку з категорій у довільному порядку, простим кліком на цю категорію. Після чого з’явиться список товарів, та меню фільтрації (рис 3.13).

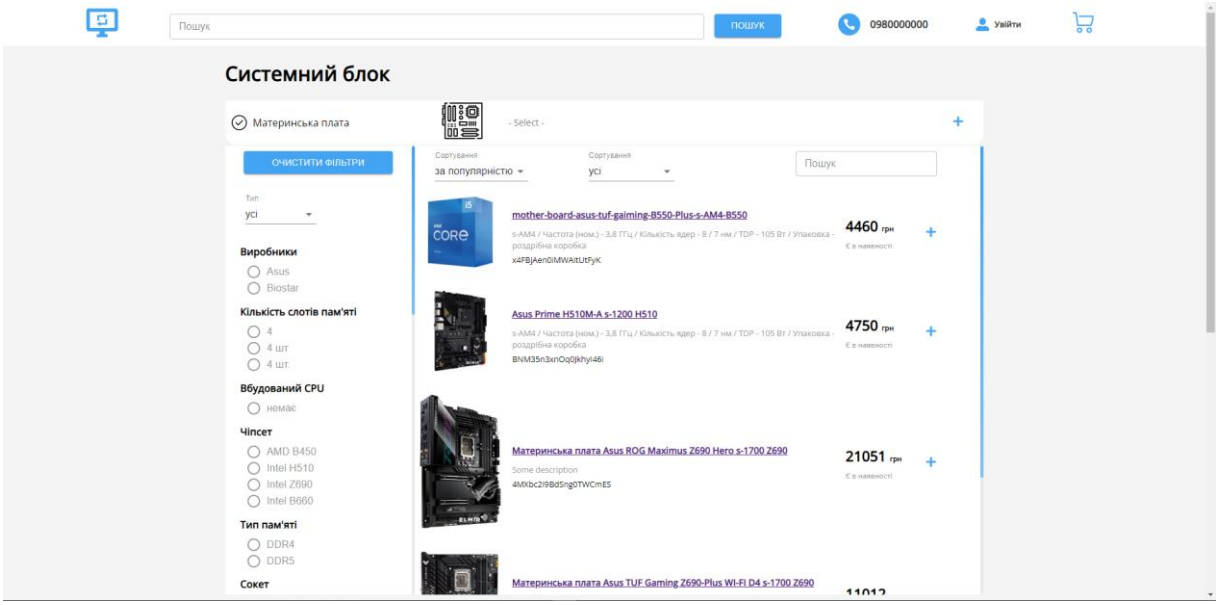


Рисунок 3.13 – Вигляд екрану після обраної категорії

Список товарів можна фільтрувати та сортувати. Фільтрування працює по принципу AND. Тобто обраний варіант звужує кількість фільтрів для того, щоб із вибраним фільтром шукати товари, які будуть підходити і під старий параметр фільтру, і під новий. Фільтрування та сортування товарів можна побачити на рисунках 3.14-3.18.

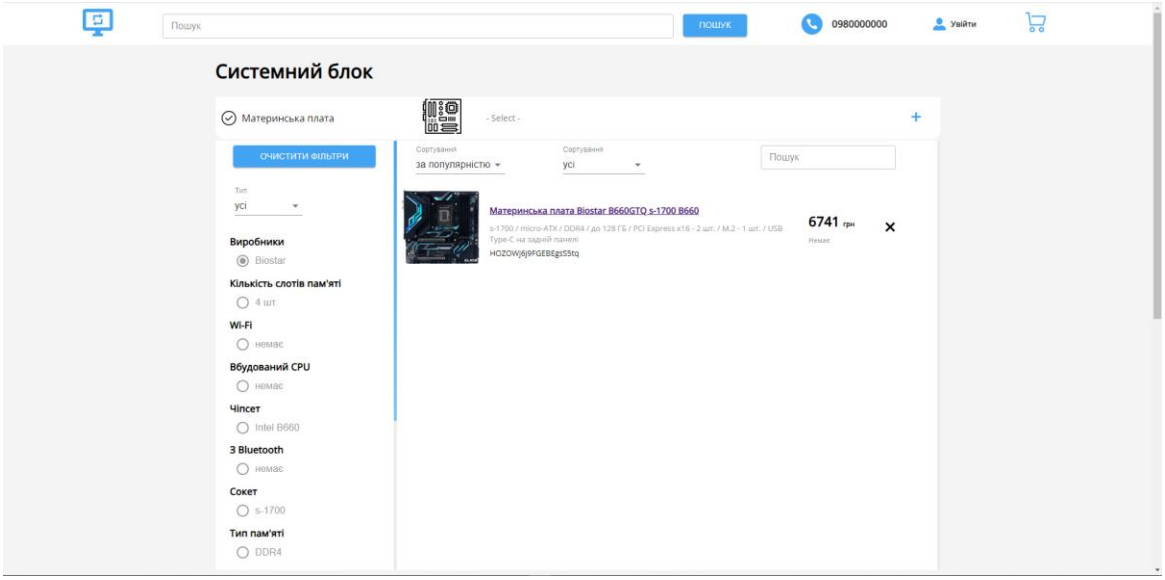


Рисунок 3.14 – Приклад фільтрації товару за брендом

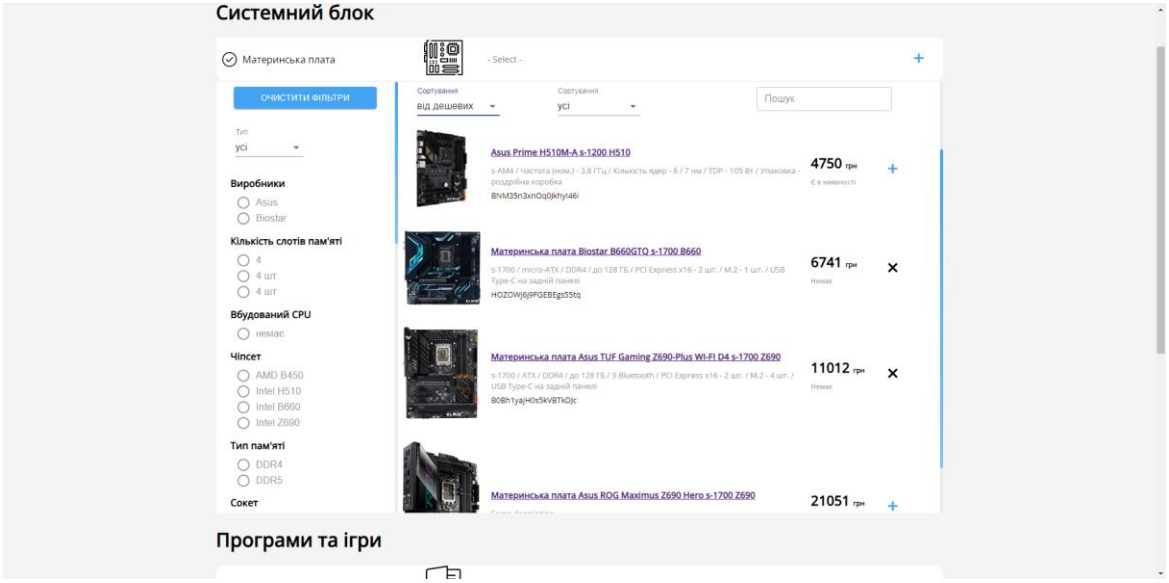


Рисунок 3.15 – Приклад сортування товарів за критерієм (від дешевих до дорогих)

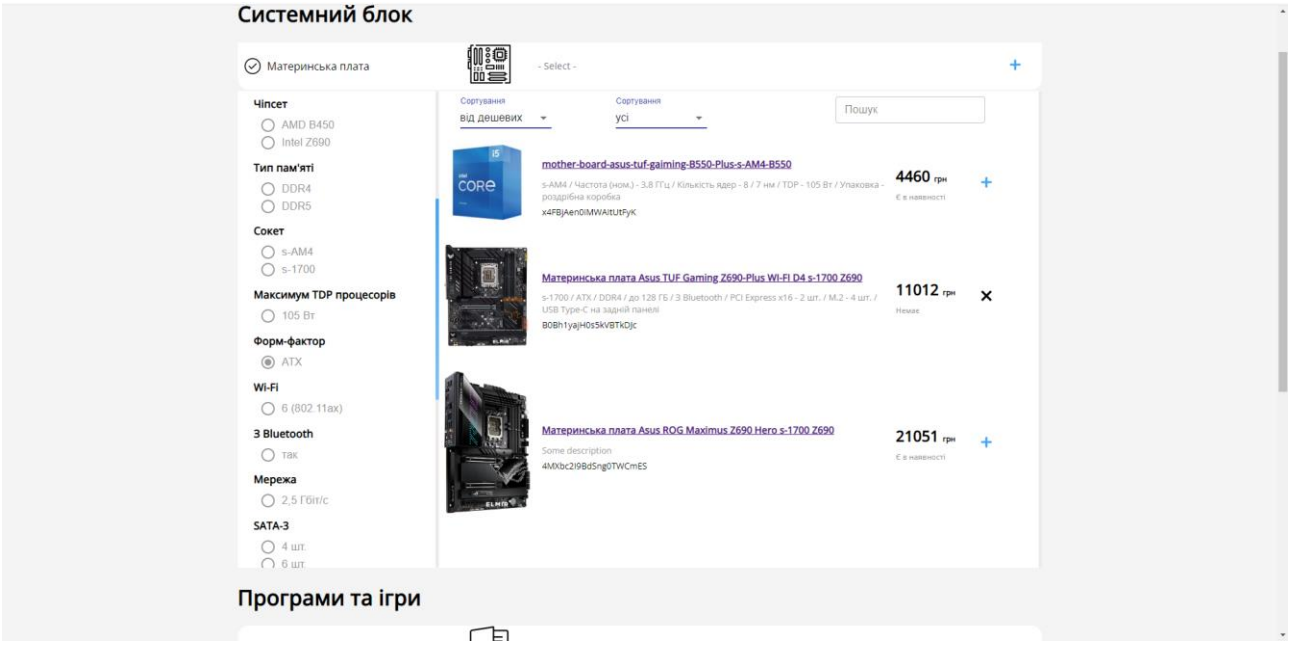


Рисунок 3.16 – Приклад сортування за критерієм та фільтрації за форм фактором

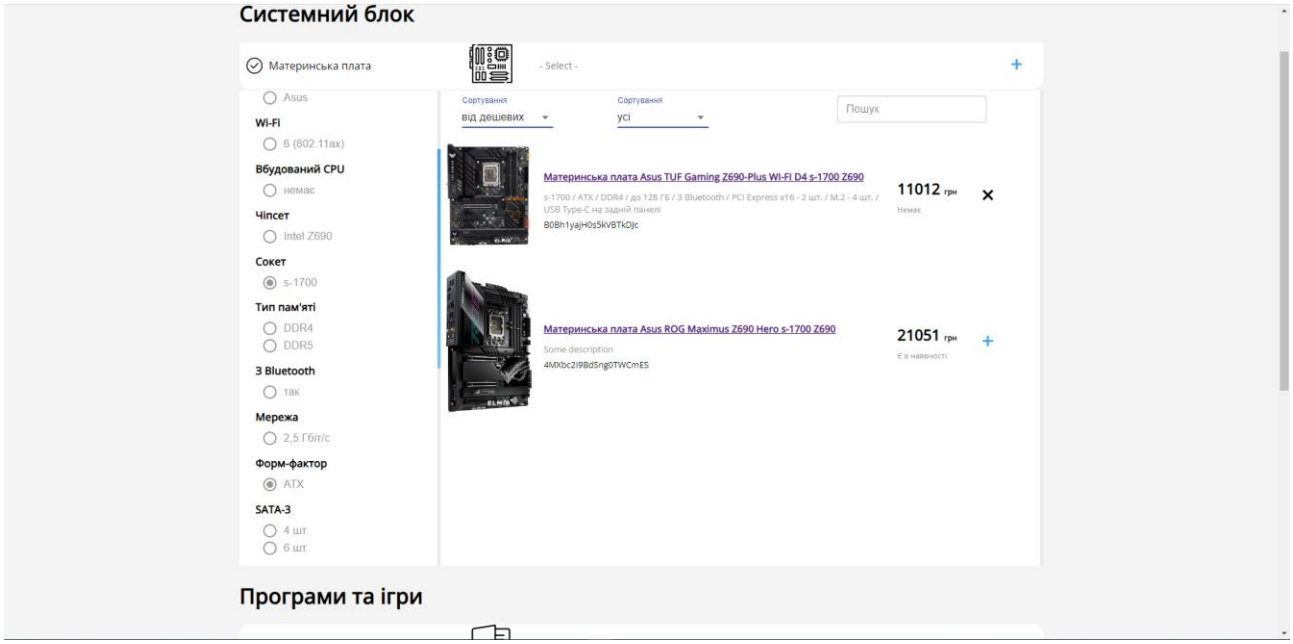


Рисунок 3.17 – Приклад сортування за критерієм та двома атрибутами товару (форм фактор та сокет)

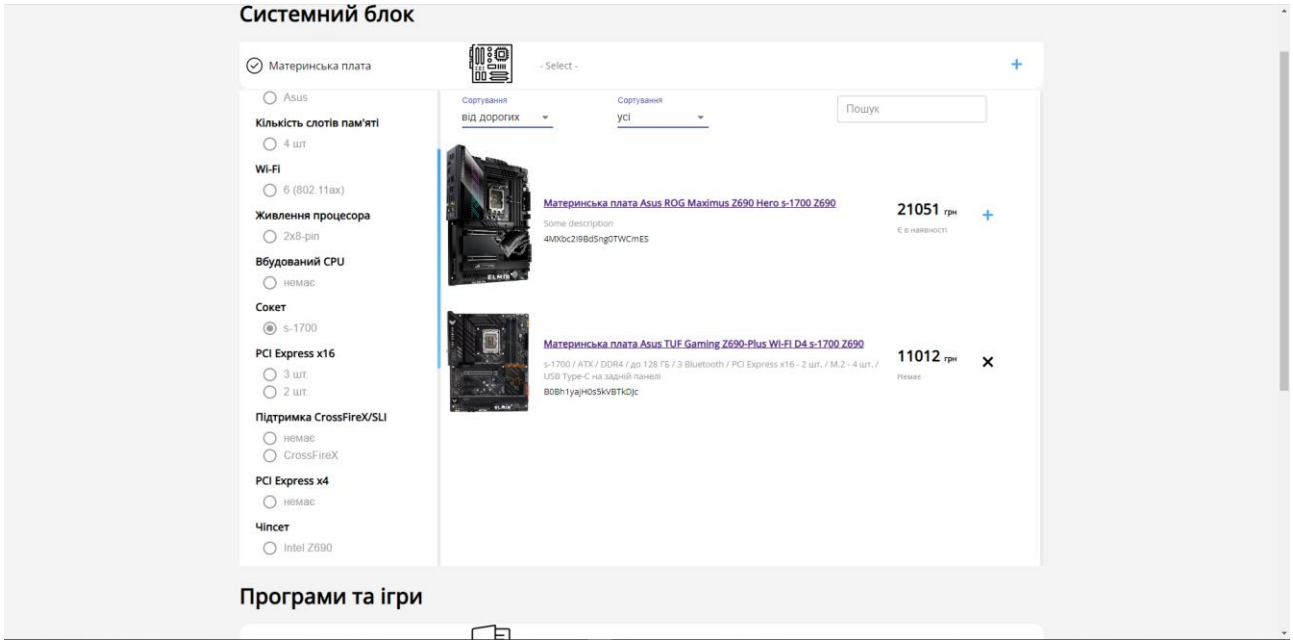


Рисунок 3.18 – Приклад сортування за іншим критерієм та відфільтрований за двома критеріями

В списку товарів або вже при обраному товарі за категорією, можна клацнути на назву товару та перейти на детальний огляд товару. Тут можна побачити назву товару, ціну, гарантію, характеристики апаратного компоненту та додати товар до кошику. У випадку, якщо товар уже у кошику чи користувач щойно купив товар, це буде відображено на сторінці. Сторінка детального оглядку товару показана на рисунках 3.19-3.21.

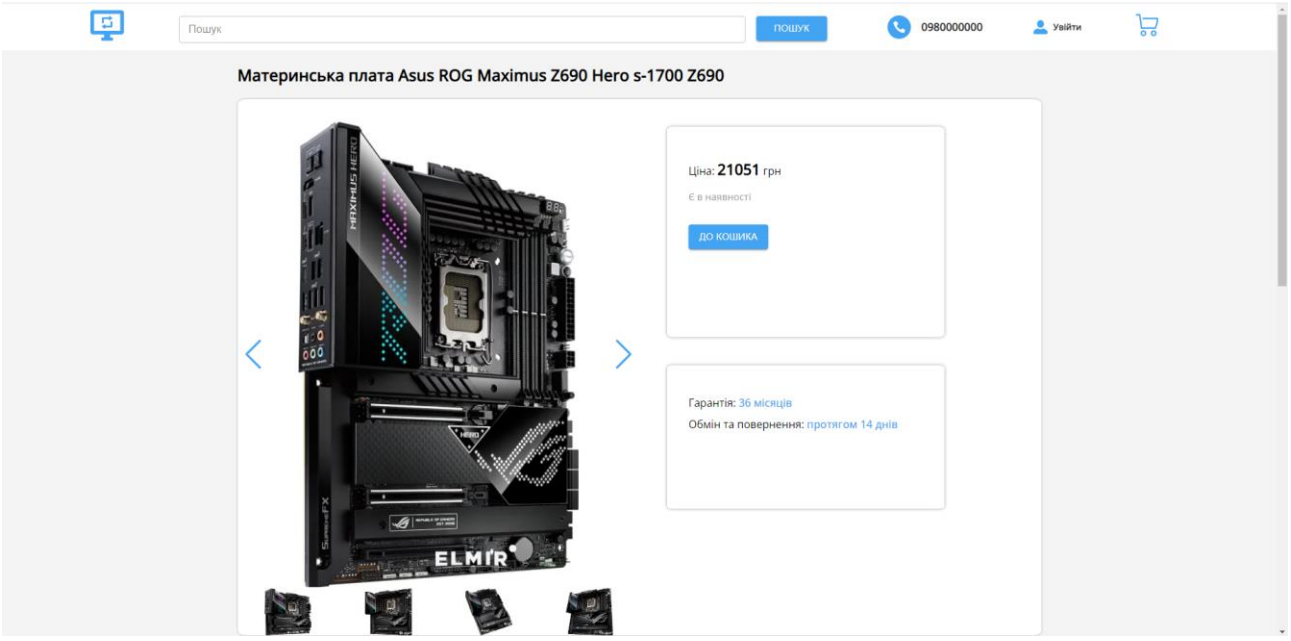


Рисунок 3.19 – Перша частина сторінки про товар

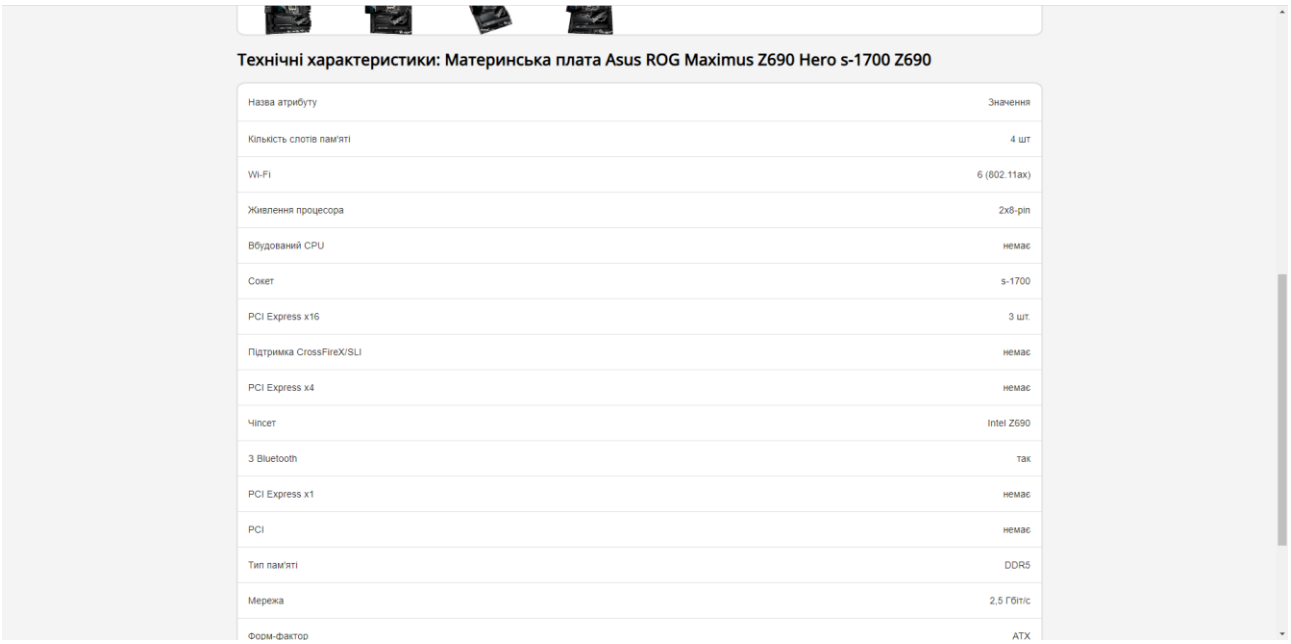


Рисунок 3.20 – Друга частина сторінки про товар

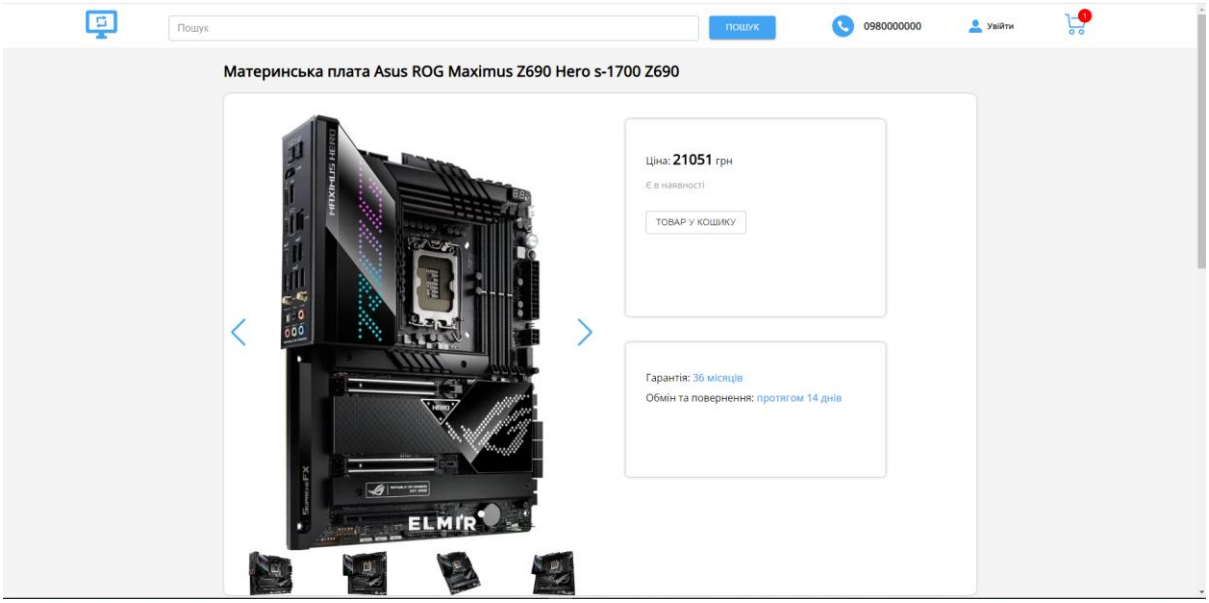


Рисунок 3.21 – Сторінка із товаром у випадку, якщо товар уже в кошику

Якщо повернутися до головного меню, обраний товар буде відображено на панелі категорії, до якої він відноситься та буде виведено всю важливу інформацію про товар. Також внизу буде відображатися сума товарів у кошику.

Ідентифікатор справа вверху над корзиною буде показувати поточну кількість товарів у кошику (рис. 3.22).

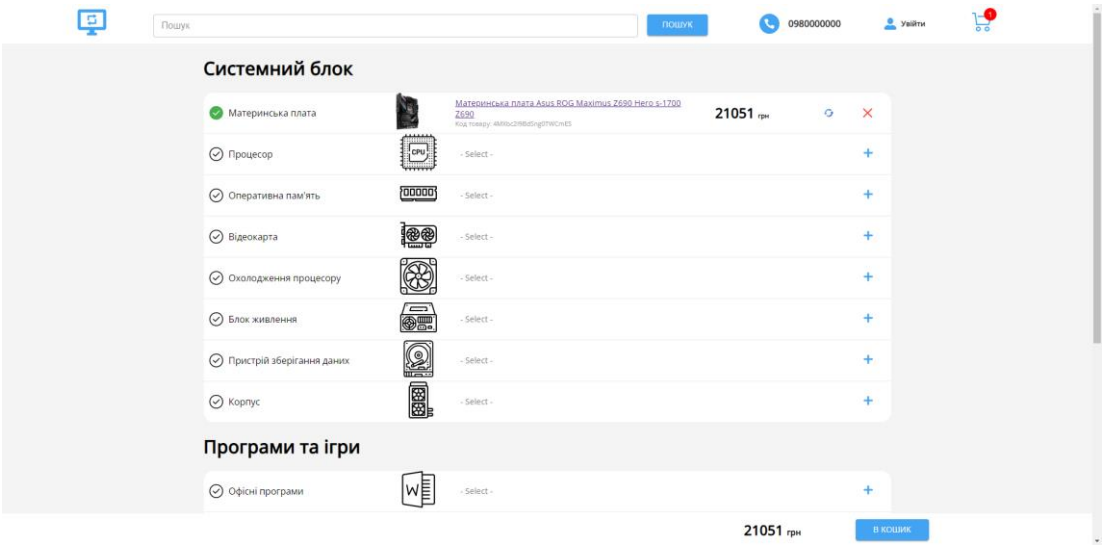


Рисунок 3.22 – Вигляд головного меню після доданого товару по категорії

Уже обраний товар можна як переглянути, так і замінити або видалити з корзини. Відповідні кнопки ілюструють операції, які можна провести з товаром. При заміні товару буде відображатися обраний товар та можливість додати новий (рис. 3.23-3.24).

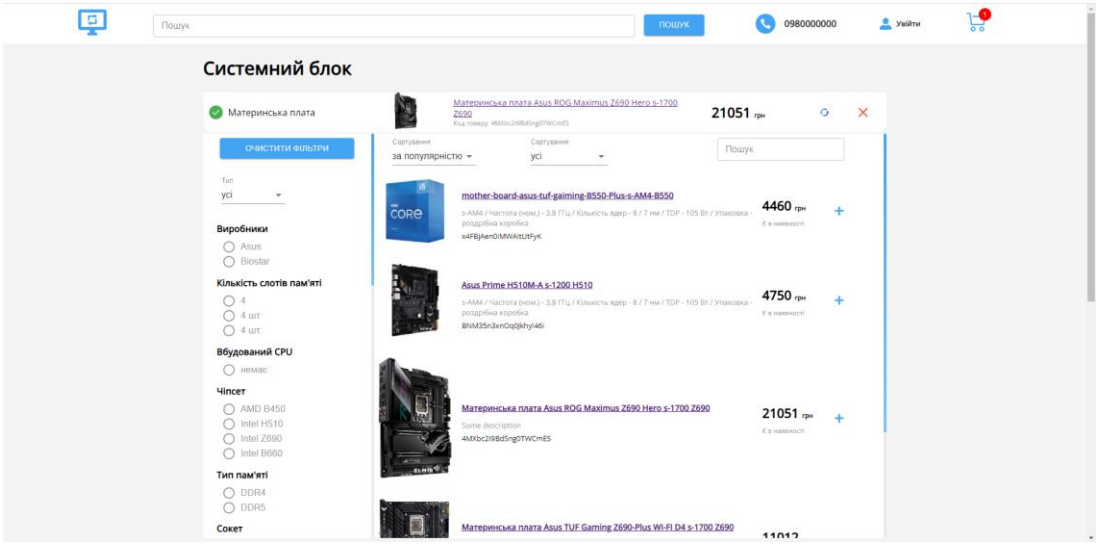


Рисунок 3.22 – Вигляд головного меню при заміні товару по категорії

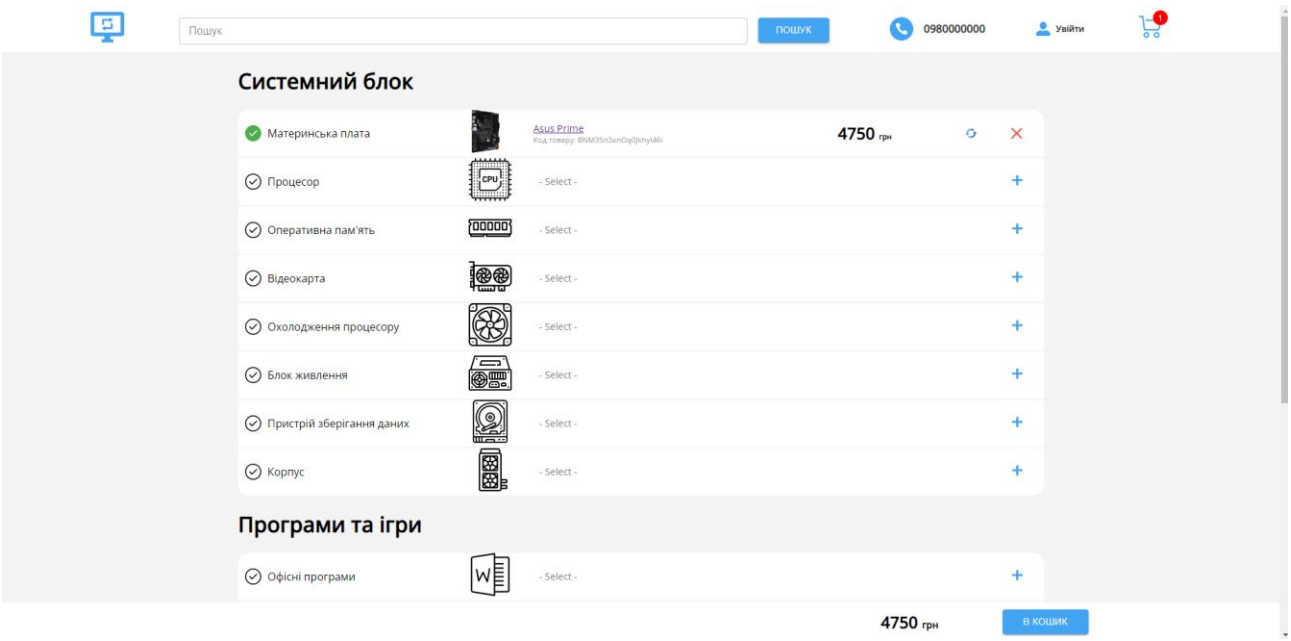


Рисунок 3.23 – Вигляд головного меню після заміни товару по категорії

Вигляди головного меню при додаванні товарів до кошику. Ціна внизу додатку змінюється відносно загальній сумі всіх товарів при додаванні та видаленні товарів у кошику (рис. 3.24-3.25).

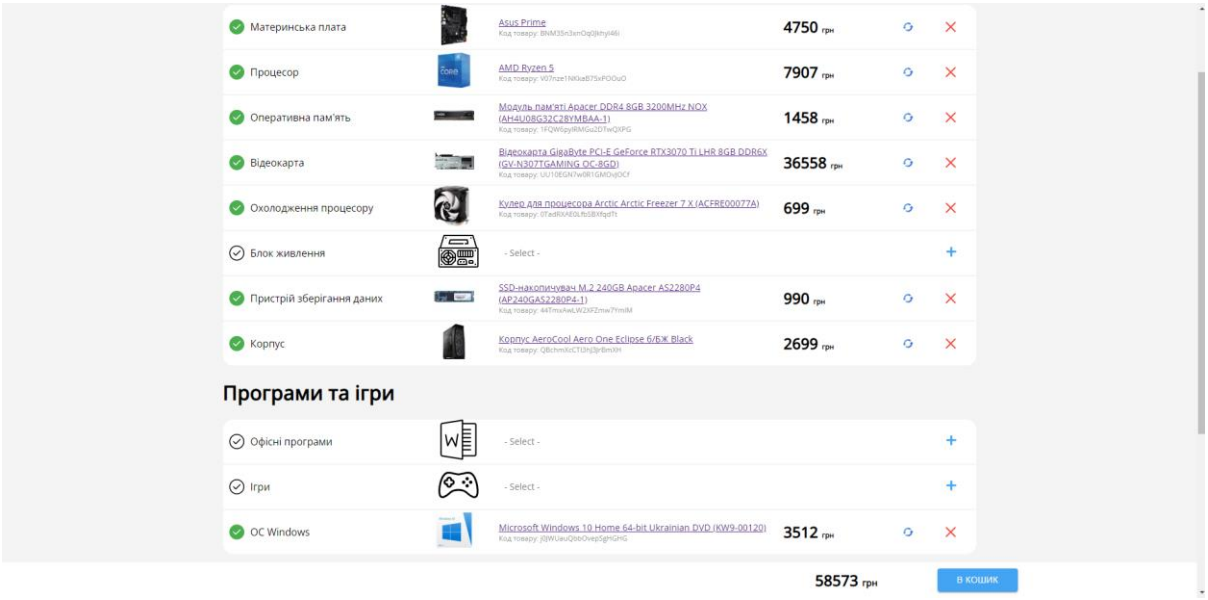


Рисунок 3.24 – Вигляд головного меню після додавання декількох товарів

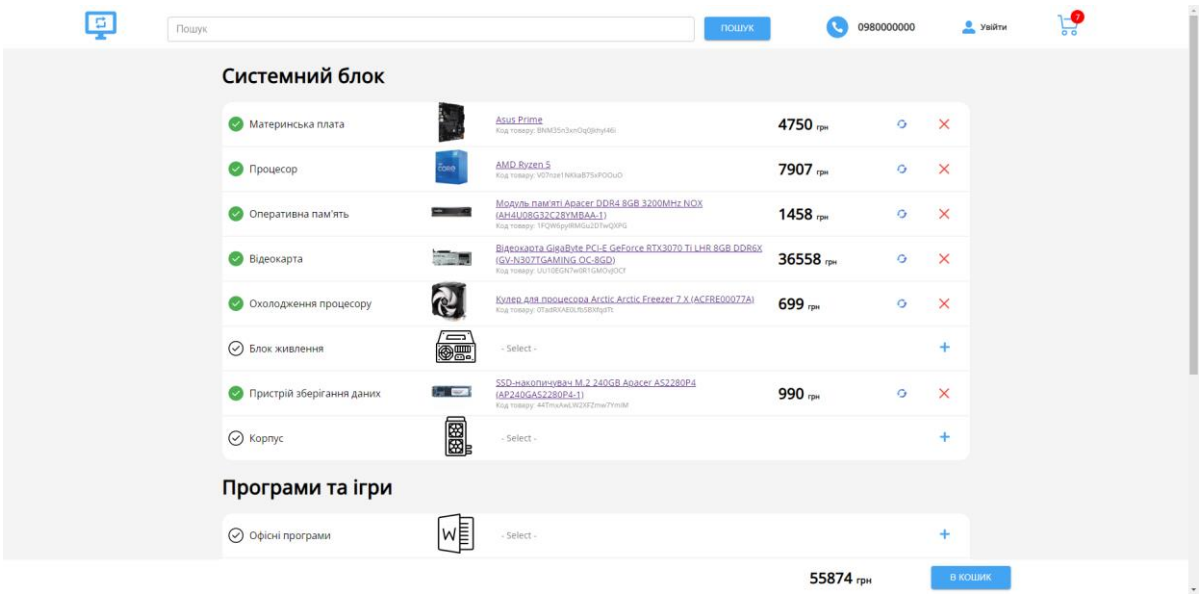


Рисунок 3.25 – Вигляд головного меню після видалення одного із товарів

Сторінка кошику містить в собі всі товари, які були туди додані з попереднього меню. В цій частині додатку можна змінити кількість товару або видалити конкретний товар. На полях кількості є валідація, тобто не може бути обрано менше одиниці товару або бути більше ніж кількість товару на складі. Вигляд кошику можна побачити на рисунках 3.26-3.27.

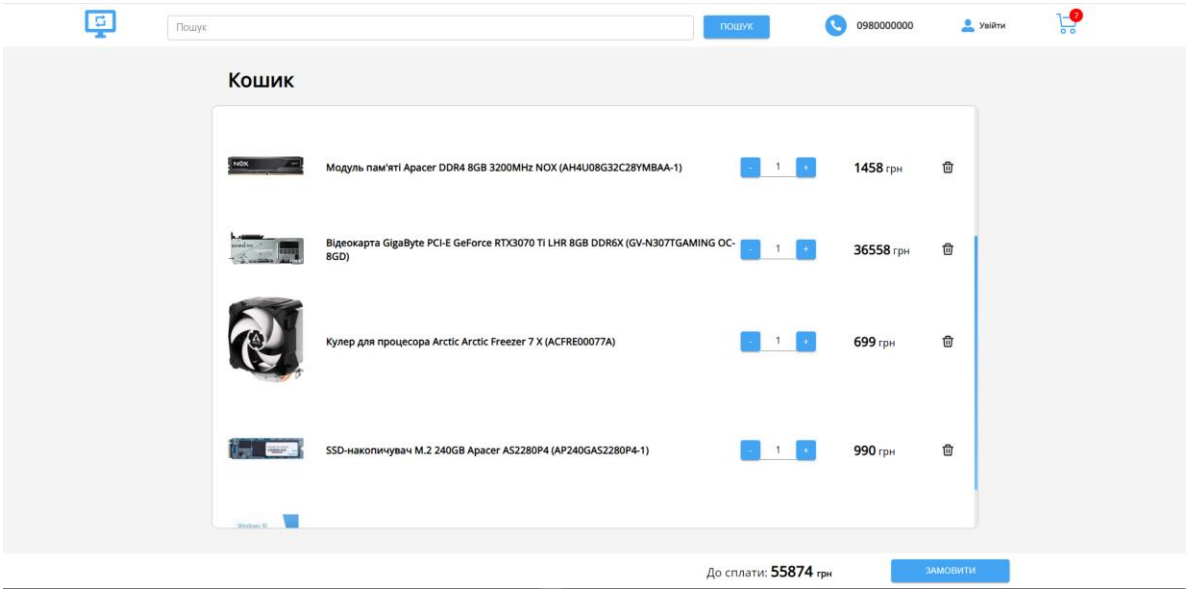


Рисунок 3.26 – Перша частина сторінки кошику

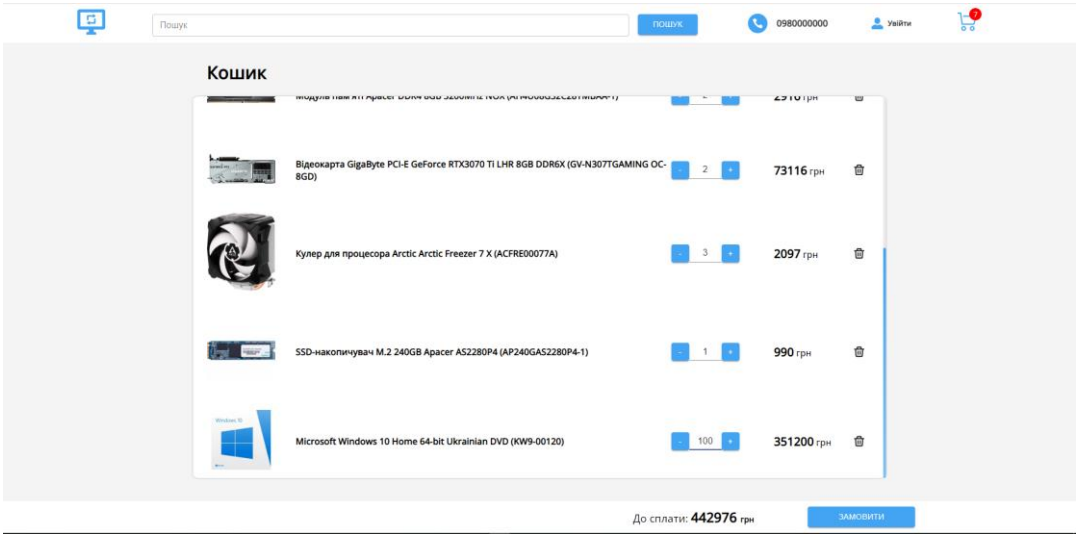


Рисунок 3.26 – Друга частина сторінки кошику

Після сформованого кошику, користувач може перейти до частини замовлення. Спочатку у користувача запитуються особисті дані. У випадку, якщо користувач зайшов у додаток, дані буде взято з аккаунту. На всіх полях накладені обмеження на дані, які туди можна записувати. Також у цій частині додатку використовується Nova Post API. Тобто, дані про місто та відділення, куди потрібно доставити замовлення беруть звідти, і актуально оновлюються згідно із офіційними змінами. Вигляд заповнення інформації для замовлення показано на рисунках 3.27-3.32.

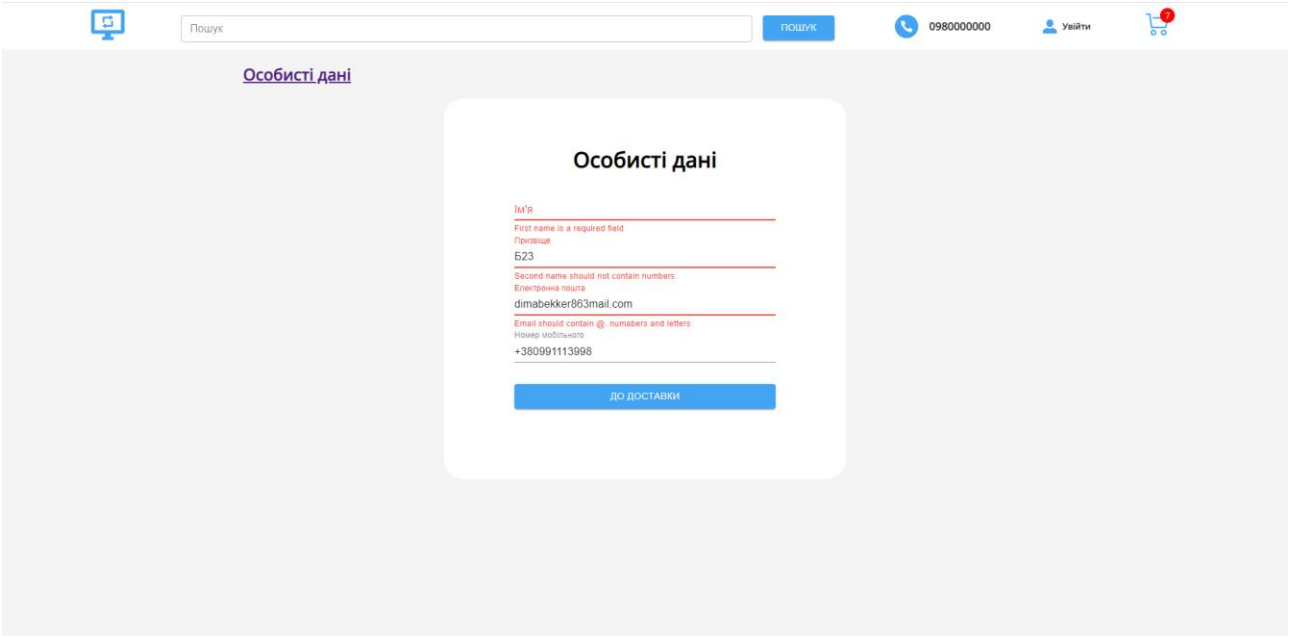
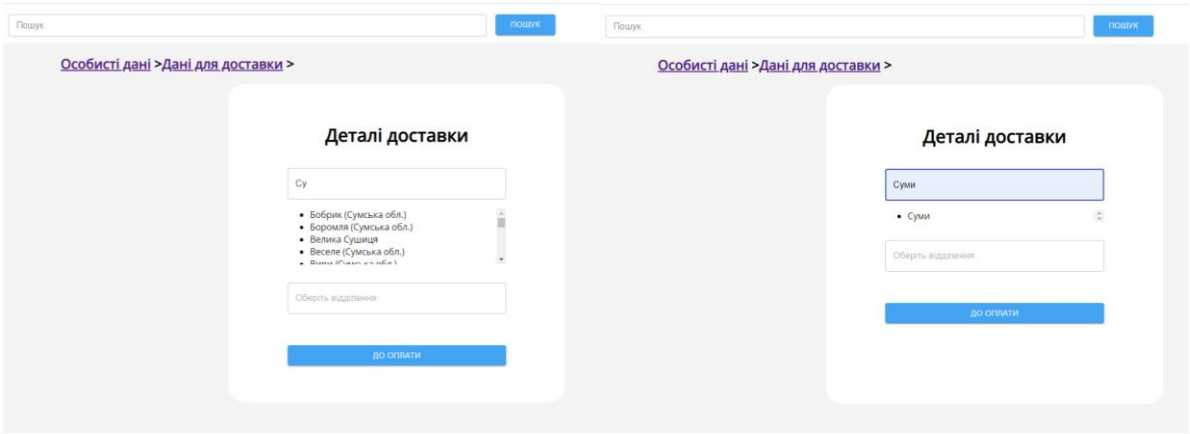
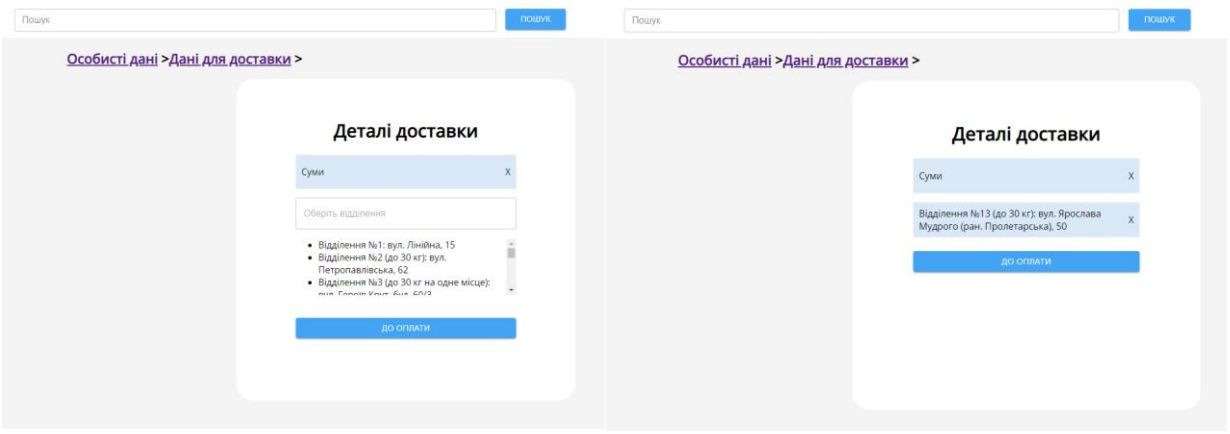


Рисунок 3.27 – Перша частина формування замовлення (особисті дані)



а б

Рисунок 3.28 – Друга частина формування замовлення (деталі доставки), вибір міста зі списку (а), вибір за повною назвою міста (б)



а б

Рисунок 3.30 – Друга частина формування замовлення (деталі доставки), вибір відділення зі списку (а), обране відділення (б)

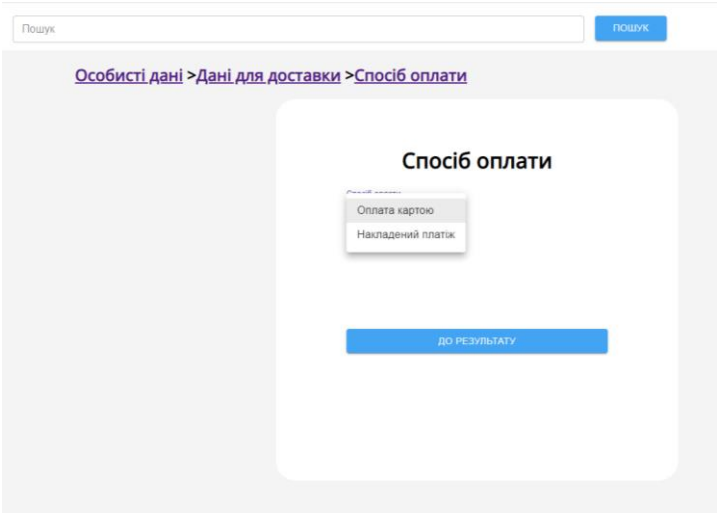


Рисунок 3.32 – Третя частина формування замовлення (спосіб оплати)

Після збору всіх даних, виводиться результуюча сторінка де можна переглянути всі дані замовлення (рис. 3.33).

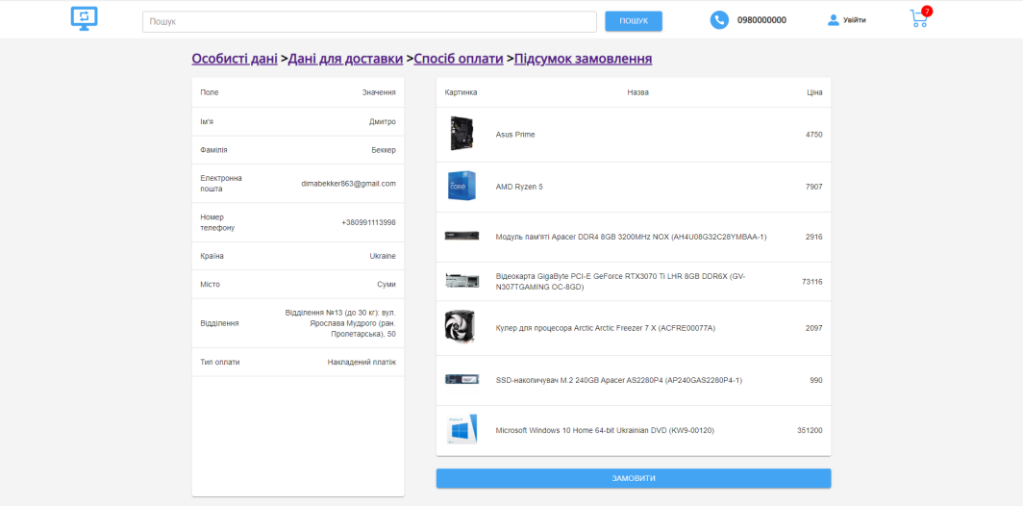


Рисунок 3.33 – Вигляд результуючої сторінки замовлення

Після кліку на замовити, дані відправляються на сервер, та замовлення надсилається клієнту та менеджеру (рис. 3.34).

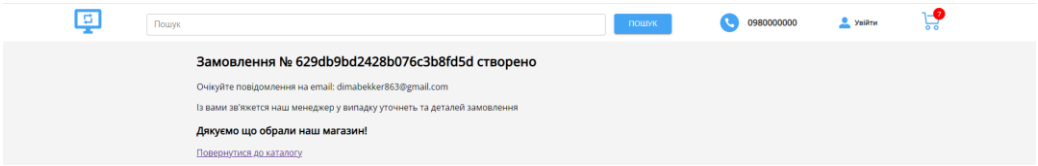


Рисунок 3.34 – Сторінка вже відправленого замовлення

Приклад сформованого звіту про замовлення можна побачити на рисунках 3.36-3.37.

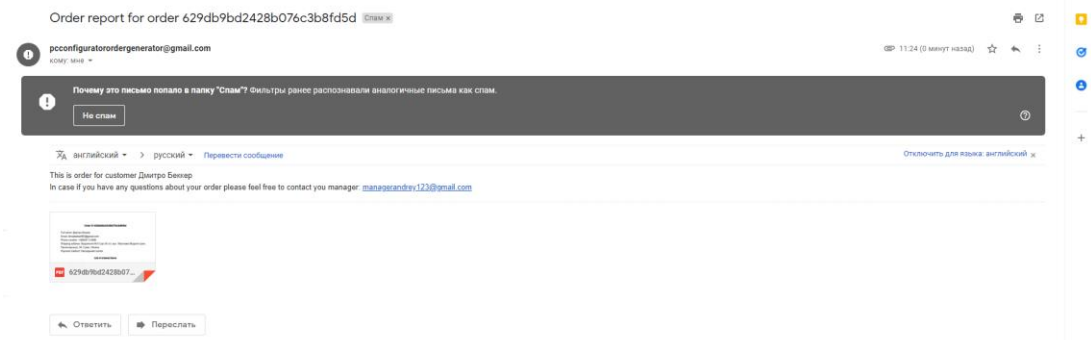


Рисунок 3.36 – Google пошта із надісланим звітом про замовлення.

Order ID 629dbde8c1811e44719c3510

Full name: Dmitry Bekker
Email: dimabekker863@gmail.com
Phone number: +380991113998
Shipping address: Відділення №13 (до 30 кг): вул. Ярослава Мудрого (ран. Пролетарська), 50, Суми, Ukraine
Payment method: Накладений платіж

List of ordered items

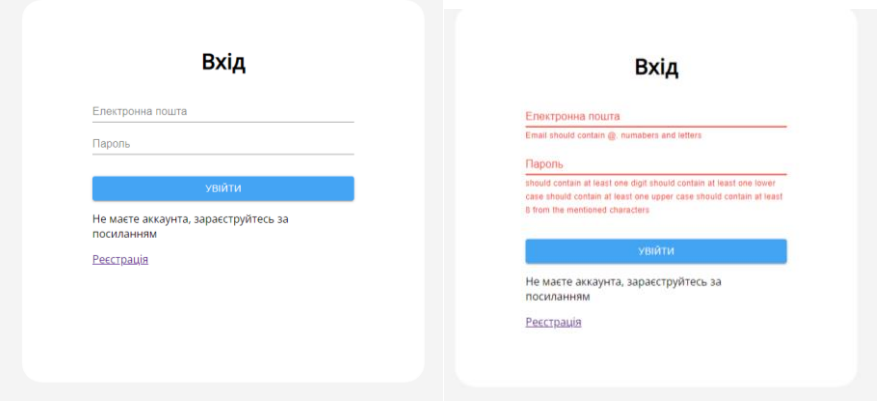
Product ID	Sku ID	Назва	Категорія	Кількість	Ціна за одиницю	Загальна ціна
4MXbc2I9BdSng0TWCmES	4MXbc2I9BdSng0TWCmES	Asus ROG Maximus Z690 Heros-1700 Z690	Материнська плата	1	21051	21051
V07nze1NKkaB75xPOOuO	V07nze1NKkaB75xPOOuO	AMD Ryzen 5	Процесор	1	7907	7907
1jjkgbJo07ZTTICvhm3z	1jjkgbJo07ZTTICvhm3z	' Kingston Fury DDR4 32GB 2x16GB 3200MHz Beast Black (KF432C16BB1K2/32)	Оперативна пам'ять	1	4639	4639
4E9G00INj9xEiJBYwtNo	4E9G00INj9xEiJBYwtNo	HP PCI-E Nvidia RTX A6000 48GB DDR6 (2S6U3AA)	Відеокарта	1	257400	257400
IMNPUw95sSXZ2AsgXAW9	IMNPUw95sSXZ2AsgXAW9	Cooler Master MasterLiquid ML280 Mirror (MLX-D28M-A14PK-R1)	Охолодження процесору	1	4021	4021
TrXXAq7XOPil4LWLawUD	TrXXAq7XOPil4LWLawUD	SSD 2.5" [illegible] 1TB AMD Радеон	Пристрій збереігання даних	1	3131	3131
QBchmXcCTI3hJ3JrBmXH	QBchmXcCTI3hJ3JrBmXH	AeroCool Aero One Eclipse / Black	Корпус	1	2699	2699
j0JWUauQbbOvepSgHGHG	j0JWUauQbbOvepSgHGHG	Microsoft Windows 10 Home 64-bit Ukrainian DVD (KW9-00120)	ОС Windows	1	3512	3512

Страница 1 из 2

Total quantity: 8
Total price: 304360

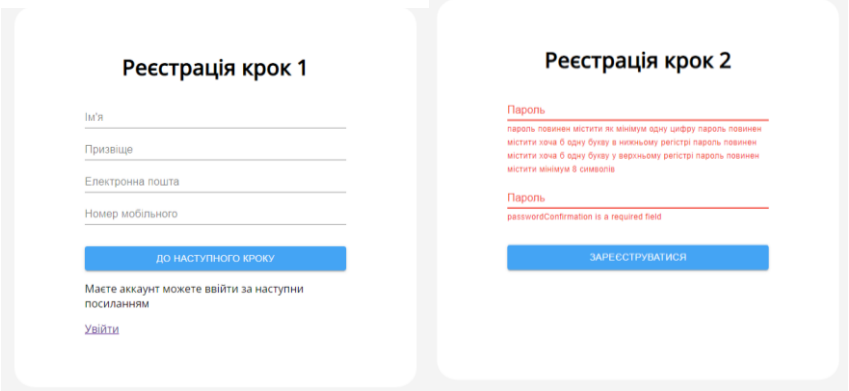
Рисунок 3.37 – PDF звіт про замовлення

У додатку також є можливість увійти та зареєструватися. Зареєстрований користувач може не заносити дані до особистих даних при формуванні замовлення. Етапи реєстрації та авторизації показано на рисунку 3.38-3.39.



а б

Рисунок 3.38 – Форма авторизації у додатку (а), при невірно заданих даних у полях (б)



а б

Рисунок 3.39 – Форма реєстрації у додатку крок 1 (а), крок 2 (б)

ВИСНОВКИ

Кваліфікаційну роботу бакалавра присвячено розробці web-додатку для конфігурування ПК інтернет магазину «PC Configurer».

У ході виконання дипломної роботи було виконано огляд досліджень та публікацій відносно архітектурних рішень для web-додатку. В результаті огляду продуктів аналогів було сформовано вимоги до функціоналу серверної частини та інтерфейсу користувача клієнтської частини додатку.

Метою даного дослідження є розробка системи для полегшення збірки комп'ютера для користувачів різного рівня підкованості та обізнаності в них за рахунок створеного web-додатку. Було сформовано мету роботи, обрано інструменти та засоби для реалізації додатку. Додаток реалізовано із використанням мови програмування Java та фреймворку Spring Boot, для створення мікросервісів та React фреймворку для написання клієнтської частини додатку.

Результати моделювання додатку представлені у вигляді:

- функціонального моделювання web-додатку в idef0;
- діаграми варіантів використання web-додатку;
- діаграми структурного моделювання;
- діаграми класів аналізу;
- діаграми компонентів web-додатку;
- діаграми розгортання;
- діаграми потоків даних dfd.

Для ведення даних про товари магазину була використана нереляційна система управління базою даних – MongoDB. Це дозволило гнучко та документо-орієнтовно створювати товари та категорії. Ця ж СУБД використовується для кошику, замовлень та користувачів.

Результатом роботи є web-додаток, який є надійним стійким та здатним на масштабування. Додаток має зручний та простий інтерфейс, функціонал достатній для користування додатком у реальних умовах.

Результати досліджень було представлено на науковій конференції викладачів, аспірантів – "IMA-2022" – ElIT Faculty's Conference.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Developing Web API’s — Microservices Architecture, SOLID, DDD, Onion Architecture, Clean Architecture, CQRS [Електронний ресурс] – Доступ до ресурсу: https://medium.com/geekculture/developing-web-apis-microservices-architecture-solid-ddd-onion-architecture-clean-55f46052c41b (Дата звернення: 25.12.21)

2. Understanding Onion Architecture [Електронний ресурс] – Доступ до ресурсу: https://www.codeguru.com/csharp/understanding-onion-architecture/ (Дата звернення: 26.12.21)

3. DDD, Hexagonal, Onion, Clean, CQRS, … How I put it all together [Електронний ресурс] – Доступ до ресурсу: https://herbertograca.com/2017/11/16/explicit-architecture-01-ddd-hexagonal-onion-clean-cqrs-how-i-put-it-all-together/ (Дата звернення: 28.12.21)

4. Onion Architecture and Domain-Driven-Design - an architect's perspective on tackling "Application Integration Hell" [Електронний ресурс] – Доступ до ресурсу: https://www.linkedin.com/pulse/onion-architecture-architects-perspective-tackling-hell-santikary/ (Дата звернення: 25.12.21)

5. Why using Microservices or Monolith can be just a detail [Електронний ресурс] – Доступ до ресурсу: https://threedots.tech/post/microservices-or-monolith-its-detail/ (Дата звернення: 24.12.21)

6. The complete Guide To Understand IDEF Diagram [Електронний ресурс] – Доступ до ресурсу: https://www.edrawmax.com/article/the-complete-guide-to-understand-idef-diagram.html (Дата звернення: 14.12.21)

7. What is Use Case Diagram [Електронний ресурс] – Доступ до ресурсу: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/ (Дата звернення: 14.12.21)

8. Модель аналізу [Електронний ресурс] – Доступ до ресурсу: https://ppt-online.org/751856 (Дата звернення: 15.12.21)

9. UML – Class Diagram [Електронний ресурс] – Доступ до ресурсу: https://www.tutorialspoint.com/uml/uml_class_diagram.htm (Дата звернення: 19.12.21)

10. UML Sequence Diagram tutorial [Електронний ресурс] – Доступ до ресурсу: https://www.lucidchart.com/pages/uml-sequence-diagram (Дата звернення: 17.12.21)

11. Deployment diagram tutorial [Електронний ресурс] – Доступ до ресурсу: https://www.lucidchart.com/pages/uml-deployment-diagram (Дата звернення: 21.12.21)

12. What is a Data Flow Diagram [Електронний ресурс] – Доступ до ресурсу: https://www.lucidchart.com/pages/data-flow-diagram (Дата звернення: 22.12.21)

13. What is NoSQL [Електронний ресурс] – Доступ до ресурсу: https://www.mongodb.com/nosql-explained (Дата звернення: 08.06.2022)

14. Introducing JSON [Електронний ресурс] – Доступ до ресурсу: https://www.json.org/json-en.html (Дата звернення: 08.06.2022)

15. How to draw NoSQL Data Model Diagram [Електронний ресурс] – Доступ до ресурсу: https://www.techighness.com/post/how-to-draw-no-sql-data-model-diagram/ (Дата звернення: 08.06.22)

16. What is REST API [Електронний ресурс] – Доступ до ресурсу: https://www.redhat.com/en/topics/api/what-is-a-rest-api (Дата звернення: 05.06.2022)

17. Richardson Maturity Model [Електронний ресурс] – Доступ до ресурсу: https://restfulapi.net/richardson-maturity-model/ (Дата звернення: 05.06.2022)

18. 4 Maturity Levels of REST API Design [Електронний ресурс] – Доступ до ресурсу: https://blog.restcase.com/4-maturity-levels-of-rest-api-design/ (Дата звернення: 05.06.2022)

19. What is a Product Information Management System (PIM) System [Електронний ресурс] – Доступ до ресурсу: https://www.pimvendors.com/articles/what-is-pim (Дата звернення: 05.06.2022)

20. Understanding Replicas Algolia [Електронний ресурс] – Доступ до ресурсу: https://www.algolia.com/doc/guides/managing-results/refine-results/sorting/in-depth/replicas/ Дата звернення: 05.06.2022)

ТЕХНІЧНЕ ЗАВДАННЯ

на розробку інформаційної системи
«Web-додаток для конфігурування ПК інтернет магазину «PC Configurer»»

ПОГОДЖЕНО:

Доцент кафедри комп’ютерних наук

_____ Марченко А.В.

Студент групи ІТ-81

_____ Беккер Д.О.

Суми 2022

1 ПРИЗНАЧЕННЯ Й МЕТА СТВОРЕННЯ WEB ДОДАТКУ

Призначення web додатку

Web додаток призначений для створення конфігурації ПК згідно до уподобань користувача web додатку та різних характеристик (ціна, актуальність, характеристики апаратного забезпечення).

Мета створення web додатку

Метою створення додатку є забезпечення систематизованого, гнучкого та поетапного підходу до зборки комп’ютера, що значно покращую досвід користувача при конфігуруванню ПК.

Цільова аудиторія

До цільової аудиторії можна віднести людей, зацікавлених в зборці своїх власних ПК, самостійно. Статистично, це доволі велика кількість людей, так як зараз дуже популярні комп’ютерні ігри, через що люди самостійно збираюсь ігрові комп’ютери.

2 ВИМОГИ ДО WEB ДОДАТКУ

2.1 Вимоги до web додатку в цілому

2.1.1 Вимоги до структури й функціонування web додатку

Web додаток повинен функціонувати для всіх типів пристроїв таких як: смартфон, комп’ютер.

2.1.2 Вимоги до модераторів

Ніяких особливих знань для наповнювачів контенту не потрібно, в системі буде доступний спеціальний модераторський розділ де адмін, або модератор

зможе додавати, редагувати, видаляти інформацію пов’язану із товарами та категоріями.

2.1.3 Вимоги до збереження інформації

Інформація буде зберігатися в базі даних, звідки буде діставатися на сервер, а далі за рахунок запитів на сервер будуть діставатися на клієнтську частину (браузер клієнта).

2.1.4 Вимоги до розмежування доступу

Розроблюваний додаток буде містити два рівня розділення доступу: користувачі та модератори. Модератор матиме доступ до всієї інформації, в той час як звичайний користувач не матиме доступу до внутрішньої будови товарів та категорії на сайті.

2.2 Структура web додатку

2.2.1 Загальна інформація про структуру web додатку

Web додаток буде складатися із основної сторінки, яка буде модульною. Перший модуль буде представляти собою компоненти для комплектації системного блоку, де можна буде обирати із обширної кількості в каталозі, там же можна буде вибирати товари за різними характеристиками, також там буде доступне сортування за різним параметрами (ціна, популярність та іменування).

Другий модуль буде містити в собі периферійні частини до комп’ютера, там також буде доступний пошук в каталозі по обраному компоненту, відбір (за категоріями, брендами та ціною та специфічними характеристиками певного компоненту).

Третій модуль буде в собі містити програмне забезпечення та ігри. В цьому модулі також будуть доступні відбір та сортування за ціною, назвою, брендом та іншим специфічними для компонента характеристиками.

Відокремленою сторінкою буде кошик де можна буде переглянути товари та у випадку чого видалити або змінити кількість обраних товарів.

Також буде сторінка для оформлення замовлення, яка буде складатися із анкети (для неавторизованих користувачів), доставки, оплати та фінального перегляду важливої для відправлення інформації.

Відокремленою також буде сторінка для авторизації та реєстрації.

2.2.2 Навігація

Навігація буде здійснюватися за допомогою шапки сайту, там буде можливість перейти до кошику, особистого кабінету, та на сторінку із конфігуруванням ПК (головна сторінка). Внизу сайту також буде можливість перейти до кошику.

Із кошика буде можливість перейти на сторінку формування замовлення.

В модераторскій частині сайту буде навігація по товарам та категоріям, де конкретні товари будуть мати окремі сторінки для наповнення контентом.

2.2.3 Дизайн та структура додатку

Дизайн web додатку буде мінімалістичним, буде максимально user friendly. Всі важливі моменти будуть виділенні за допомогою блоків, щоб користувачі не загубилися і могли швидко і легко фокусувати увагу на важливих речах. Будуть використовуватися м’які анімації для спокійного користування сайтом.

2.3 Вимоги до функціонування системи

Відповідно до вимог, розроблений додаток має задовольняти таким функціональним вимогам:

- конфігурування ПК, де всі важливі компоненти будуть розділенні між секціями, в яким можна обирати товар;
- В під модулях із каталогом обраного апаратного або програмного компоненту можна вибрати товари за ціною (діапазон від найнижчої до найвищої), назвою товару, брендом та специфічними характеристиками компоненту (наприклад процесор: сокет, модельний ряд, покоління, серія, кількість ядер, кількість потоків і т.д.). Також товари можна сортувати за ціною, популярністю та назвою);
- Авторизація та реєстрація;
- Додавання, редагування та видалення товарів та категорій (модерераторська частина додатку);
- Формування PDF звіту про покупку;

2.4 Вимоги до видів забезпечення

2.4.1 Вимоги до інформаційного забезпечення

Реалізація web додатку відбувається із використанням наступних технологій та інструментів:

- Java 11
- Spring, Spring Boot, Spring Security
- MongoDB/MySQL
- Algolia API
- Gradle
- IntelliJ IDEA
- Feign client
- Junit, Spook, Mockito
- Lombok
- JSON
- React
- HTML, CSS, SASS, JS

- Nova Post API
- iText
- GIT/ GitHub

2.4.2 Вимоги до лінгвістичного забезпечення

Web-додаток буде містити один лінгвістичний пакет для локалізації основного інтерфейсу – українська мова.

2.4.3 Вимоги до програмного забезпечення

Програмне забезпечення клієнтської частини повинне задовольняти наступним вимогам:

- Web браузер – Google Chrome, Opera, Firefox, Safari, Edge бажано нових версій

3 СКЛАД І ЗМІСТ РОБІТ ЗІ СТВОРЕННЯ WEB ДОДАТКУ

Докладний опис етапів роботи зі створення web додатку наведено в таблиці А.3.

Таблиця А.3 – Етапи створення web додатку

№	Склад і зміст робіт	Строк розробки (у робочих днях)
1	Постановка задачі проекту	7 день
2	Складання технічного завдання	3 дні
3	Підготовка прототипу	10 днів
4	Створення макету дизайну web додатку	7 дні
5	Розробка	35 днів
6	Робота над адаптивністю дизайну	7 дні
7	Тестування	10 день
8	Підготовка до випуску	5 день
9	Завершення роботи	3 день
10	Загальна тривалість робіт	92 дні

4 ВИМОГИ ДО СКЛАДУ Й ЗМІСТУ РОБІТ ІЗ ВВЕДЕННЯ WEB ДОДАТКУ В ЕКСПЛУАТАЦІЮ

Для введення web додатку в експлуатацію потрібно локально розгорнути його на локальному хості робочої станції.

ДОДАТОК Б
ПЛАНУВАННЯ РОБІТ

Мета проекту – забезпечення систематизованого, гнучкого та поетапного підходу до зборки комп'ютера, за допомогою web додатку, що допомагає конфігурувати та підібрати частини для зборки комп'ютера під різні задач.

Користувач web додатку може зайти на сайт і підібрати та зібрати повністю робочий та вже сконфігурований за власним бажанням та іншим параметрами (ціна, характеристики та ін.) комп'ютер. Також можна створити декілька комп'ютерів та порівняти їх між собою для того, щоб обрати оптимальну конфігурацію.

Використання даного додатку дозволить користувачам скоротити час на пошук апаратних частин ПК, також процес конфігурації буде повністю поетапним, що не дозволить загубити важливих часин ПК. Додаток дозволить починаючим користувачам навчитися і зрозуміти із чого збирається ПК та як обрати найкращий, збираючи його самостійно.

Розроблення web додатку має бути завершено до того часу який буде встановлено графіком та доступних ресурсів. Даний продукт буде актуальним на території України, оскільки доставка ПК буде не із легких задач, то доставляти в інші країни буде доволі проблематично та дорого.

Результат деталізації методом SMART розміщено у таблиці Б.1.

Таблиця Б.1 – Деталізація мети проекту методом SMART

Specific (конкретна)	Спрощення процесу збирання ПК, систематизований та поетапний підхід до конфігурації комп'ютера
Measurable (вимірювана)	Повністю сконфігурований користувачем ПК, який буде готовий для доставки

Продовження таблиці Б.1 – Деталізація мети проекту методом SMART

Achievable (досяжна, узгоджена)	**Вересень – Листопад**, активне планування архітектури(ТЗ, планування спринтів, вибір стеку технологій, опис моделей даних, розробка БД). **Грудень – Березень**, розробка сервісів та повноцінну функціональної системи. **Квітень – Травень**, полірування проекту (рефакторинг, усунення дефектів)
Relevant (актуальна)	Для підвищення рейтингу серед магазинів по продажу електротехніки та підвищення ефективності продажу комп'ютерного обладнання
Time-framed (обмежена в часі)	Є конкретний термін – до кінця 4 курсу (05 червня 2022р.).

Планування змісту робіт. Для представлення всіх робіт пов'язаних із проектом використовується ієрархічна діаграма декомпозиції WBS. На ній представлена загальна робота, яку потрібно виконати. В даному випадку – це створення мікро сервісного додатку «PC Configurer». Роботу розділено на шість основних етапів, які відбуваються послідовно, хоча етап unit тестування можна проводити під час розробки. Так як в більшості випадків праця послідовна, вона розділена у оптимальному порядку для усунення затримок. Кожний наступний крок у її виконанні потребує попереднього.

Робота починаються із постановки задачі, де описується предметна область та формалізація ідеї проекту до більш чітких задач.

Відразу після планування настає фаза проектування. Усі завдання починають набувати чіткості, в залежності від функціоналу, інструментів реалізації, плану, ризиків та критеріїв виконаної роботи готової до здачі. Для початку описуються всі основні функціональні можливості додатку. Після чого

потрібно підігнати їх під інструменти реалізації. На цій стадії можливі легкі зміни функціональності та визначення альтернатив для інструментів. Розробка структури роботи під собою має на увазі планування робіт, та векторів для руху, розробки, тестування та деплойменту. Календарний план потрібен для відстеження прогресу роботи та загального розуміння до якого часу, які етапи повинні бути виконанні. Ідентифікація ризиків допомагає зрозуміти, можливість виникнення необхідності перенесення робіт або затримки під час розробки та інших етапів. Опис критеріїв якості – це ті пункти, які допоможуть зрозуміти чи цілі проекту були досягнуті.

Наступний етап – реалізація проекту на основі тих задач, які поставленні та інструментів, що обрані. Почати роботу краще всього саме із архітектури мікро сервісного web-додатку. Загальне розуміння сервісів та їх комунікації між один одним потрібне для правильного початку розробки та побудови зв’язків між ними. Наступний крок – це написання скелетів сервісів. Потрібно написати MVC (module-view-control) модель, для початку робіт. Далі наступає розробка проекту.

Тестування є невід’ємним компонентом для написання працездатного додатку. У даному випадку доволі простий порядок виконання робіт. Для початку тестування потрібно провести злагоджену роботу сервісного шару кожного мікро сервісу. Для цього використовують unit тестування, яке перевіряє одну функціональну одиницю проекту. Далі проводять інтеграційне тестування, для працездатності різних сервісів між собою, мікро сервісного компоненту. Після чого слідує мануальне тестування для перевірки того, як буде використовуватися API, та чи все відображається і працює, як потрібно для frontend частини.

Розгортання проекту потрібне для розділення тестування, розробки, автоматизації процесу збирання та розгортання проекту в потрібному середовищі.

При фінальній частині проекту потрібно підключити проект до Sonar системи, що гарантує перевірку тестів та коду на написання чистого та

відлагодженого продукту. Плюсом іде перевірка на можливі вразливості системи. Документація у вигляді Swagger, дасть повний опис сервісів та можливостей для їх використання. Діаграма WBS із усіма вище описаними етапами зображені на рисунку Б.1.

Для розділення робіт між учасниками роботи використовується дігамма OBS. В даному випадку всі активності пов’язані із ідентифікацією ідеї та описом робіт призначений керівнику дипломного проекту. Усі інші етапи повинні бути виконані студентом. Схему OBS представлено на рисунку Б.2.

Матрицю відповідальності за роботи по проекту представлено в таблиці Б.2.

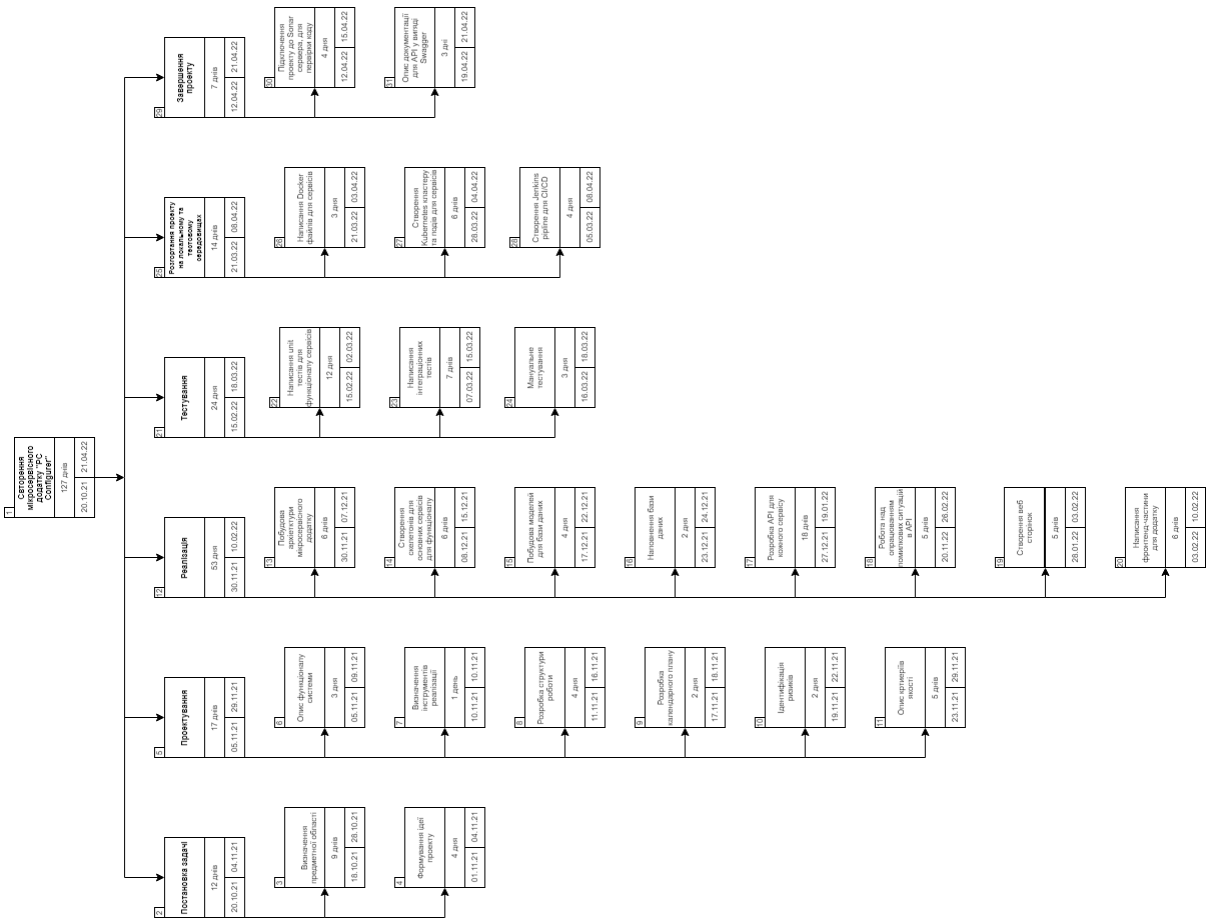


Рисунок Б.1 – Діаграма декомпозиції проекту засобами WBS

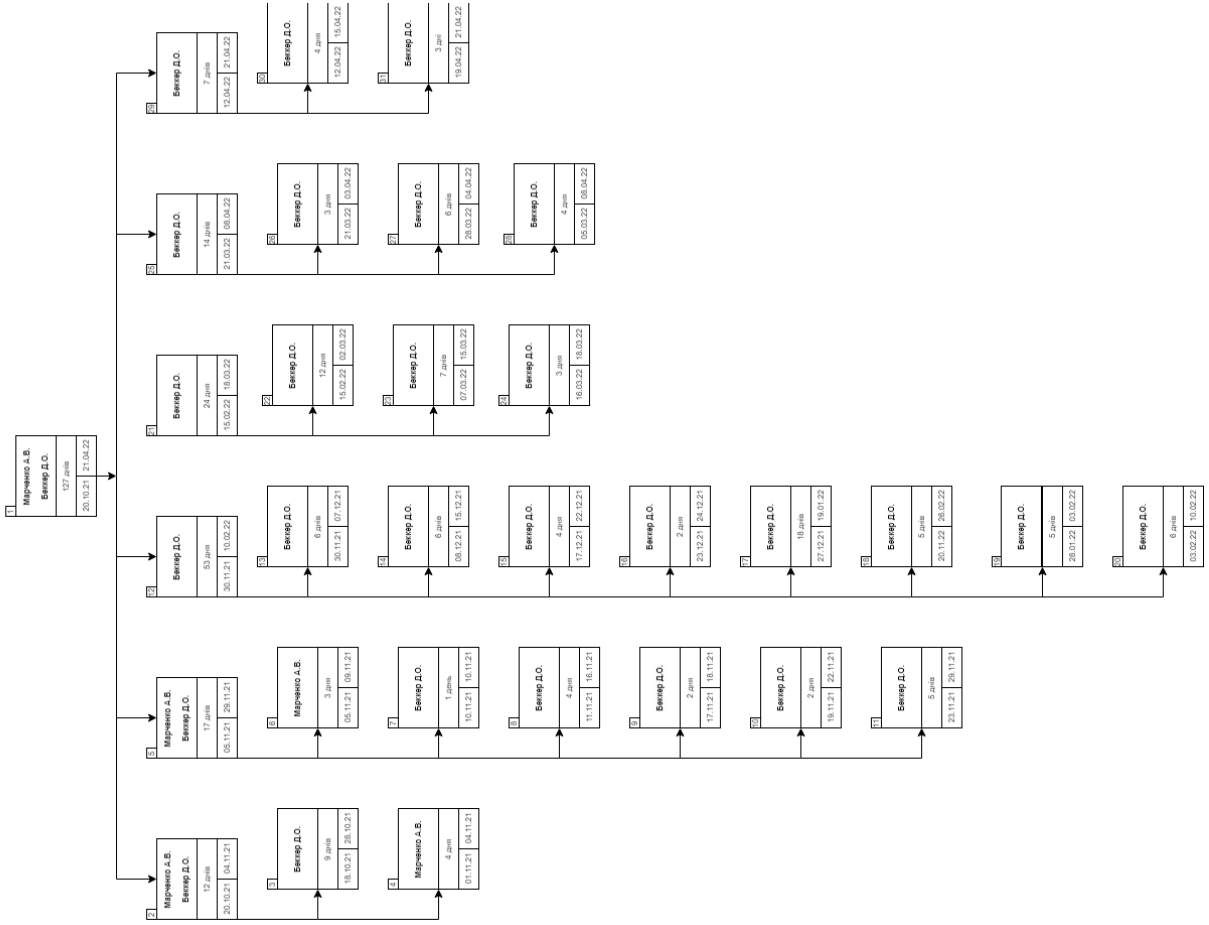


Рисунок Б.2 – Діаграма розділення робіт зроблена засобами OBS

Діаграма Ґанта, була побудована для того щоб відстежувати основні метрики, такі як: завдання та під-задачі, строки їх виконання, послідовність виконання задач та те, хто відповідальний за виконання цих цілей представлена на рисунках Б.2 - Б.6.

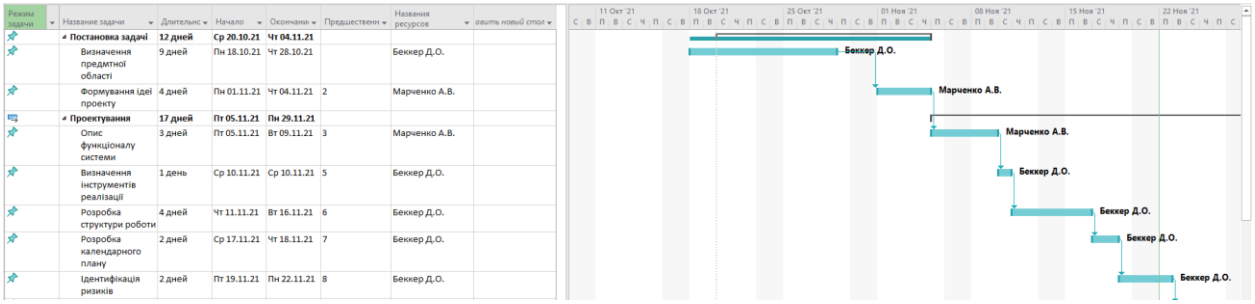


Рисунок Б.2 – Загальний вигляд діаграми всього проекту, частина 1

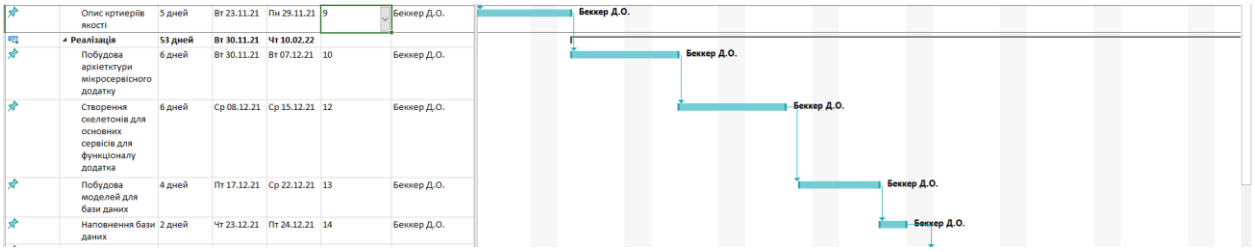


Рисунок Б.3 – Загальний вигляд діаграми всього проекту, частина 2

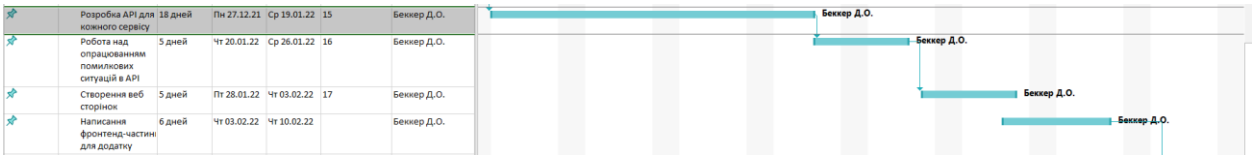


Рисунок Б.4 – Загальний вигляд діаграми всього проекту, частина 3

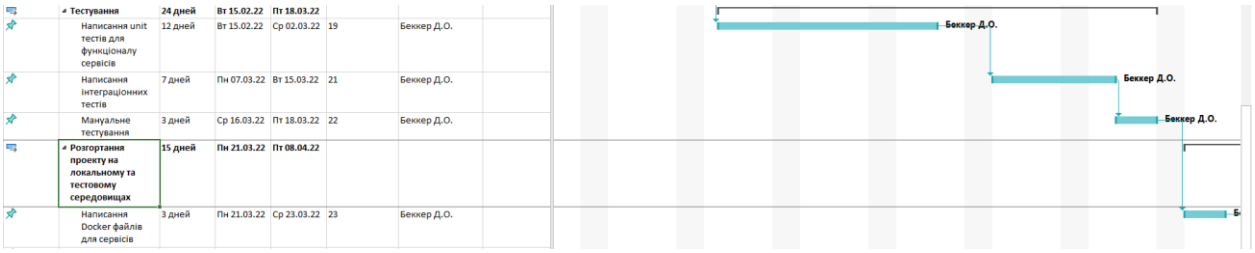


Рисунок Б.5 – Загальний вигляд діаграми всього проекту, частина 4

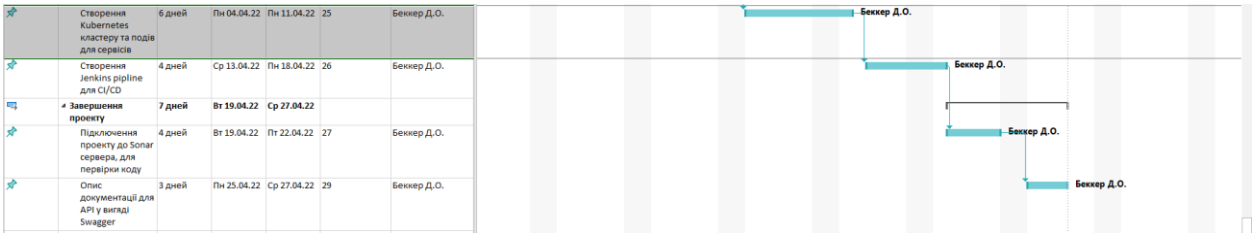


Рисунок Б.6 – Загальний вигляд діаграми всього проекту, частина 5

Управління ризиками проекту одна із важливих його части. Визначення потенційних проблем із розробкою, тестування, деплойментом, або проектування робіт, може врятувати від лишніх активностей. Також завдяки ідентифікованим несправностям можна пришвидшити процес розробки, та знайти альтернаивні рішення ще на етапі планування робіт. Інформація про ідентифіковані ризики представлена у таблиці Б.3.

Таблиця Б.7 – Ідентифікація ризиків

№	Назва (опис) ризику	Ймовірність (0,1-0,9)	Вплив (0,05-0,8)	Ранг
1	Неправильно визначене ТЗ проекту	0,3	0,6	0,18

Продовження таблиці Б.7 – Ідентифікація ризиків

2	Неправильно описані критерії завершеності проекту	0,1	0,3	0,03
3	Виявлення критичних помилок при unit тестуванні проекту	0,5	0,4	0,2
4	Виявлення критичних помилок при інтеграційному тестуванні проекту	0,3	0,7	0,21
5	Нераціональне розподілення часу	0,3	0,4	0,12
6	Часте внесення правок у ТЗ	0,3	0,4	0,12
7	Проблеми із 3d party libraries, наприклад проблеми із пошуковим двигуном	0,5	0,2	0,1
8	Неправильна конфігурація для мікро сервісного додатку під час розгортання проекту на локальному та тестовому середовищах	0,4	0,4	0,16
9	Технічні поломки техніки під час розробки	0.1	0.4	0.04
10	Розробка непотрібного функціоналу	0.5	0.3	0.15

Для надання, оцінки проблемам буде використано матрицю імовірності та ризику. Всі перешкоди, що знаходяться в зеленій зоні підпорядковуються прийнятній оцінці, жовта – виправдані, та червона – недопустимі. Зелена оцінка, має на увазі те, що проблема може лише трішки погубно вплинути на загальні роботи. Жовта оцінка означає, що ризик є, але при додаткових стратегіях можна уникнути проблем. А ось червоні, потрібно як можна скоріше відкинути, або значно зменшити можливість нанесення шкоди проекту. Розподілення ризиків за їх впливом та імовірністю показано у талиці Б.4. Шкала оцінювання за ризиками представлена у таблиці Б.5. Заходи реагування на виявлені ризики в проекті наявні у таблиці Б.6.

Таблиця Б.8 – Матриця імовірності та впливу

Ймовірність	Вплив загрози (ризику)				
	Дуже малий 0,05	Малий 0,1	Середній 0,2	Великий 0,4	Дуже великий 0,8
0,9					
0,7					
0,5		R8(0,16), R7(0,1). R10(0.15)	R3(0,2)		
0,3		R5(0,12), R6(0,12), R1(0,18)	R4(0,21)		
0,1					

Таблиця Б.9 – Шкала оцінювання за рівнем ризику

№	Назва	Межі	Ризики, які входять (номера)
1	Прийнятні	$0,005 \leq R \leq 0,05$	10. 9, 8, 7, 5, 6, 2, 1
2	Виправдані	$0,05 < R \leq 0,14$	3, 4
3	Недопустимі	$0,14 < R \leq 0,72$	

Таблиця Б.10 – Заходи реагування на виявлені ризики проекту

ID	Статус ризику	Опис ризику	Ймовірність виникнення	Вплив ризику	Ранг	План А (заходи запобігання виникненню ризику)	Тип стратегії реагування	План Б (заходи усунення наслідків ризику)
1	Відкритий	Неправильно визначене ТЗ проекту	Низький	Середній	1	Визначити технічні аспекти та умови реалізації проекту	Ухилення	Погодити всі питання із керівником
2	Відкритий	Неправильно описані критерії завершеності проекту	Низький	Низький	1	Узгодити критерії із керівником	Зменшення	Збільшити абстракцію для оцінок, щоб вони були трішки вище ніж вони є для перестраховки

Продовження таблиці Б.10 – Заходи реагування на виявлені ризики проекту

3	Відкритий	Виявлення критичних помилок при юніт тестуванні проекту	Середній	Середній	2	Робити раннє тестування, можливо використати підхід розробки через тестування (TDD)	Зменшення	Написання додаткових тестів для перевірки, вести час на відлагодження проекту
4	Відкритий	Виявлення критичних помилок при інтеграційному тестуванні проекту	Низький	Високий	2	Написання хороших юніт тестів через що інтеграційні тести будуть менше падати	Зменшення	Написання додаткових інтеграційних тестів та виділити час на відлгодження проекту

Продовження таблиці Б.10 – Заходи реагування на виявлені ризики проекту

5	Відкритий	Нераціональне розподілення часу	Низький	Низький	1	Планування через календарний план,	Зменшення	Використання різних підходів до time-management (Pomodoro technic, spaced working)
6	Відкритий	Часте внесення правок у ТЗ	Низький	Середній	1	Зменшити рівень абстракції до конкретних задач	Зменшення	Збільшення абстракції для таких задач на стороні розробки, робити трішки більше і трішки надійніше на про всяк випадок

Продовження таблиці Б.10 – Заходи реагування на виявлені ризики проекту

7	Відкритий	Проблеми із 3d party libraries, наприклад проблеми із пошуковим двигуном	Середній	Низький	1	Читання документації перед вибором інструменту, пошук можливих критичних проблем на форумах	Ухилення	Заміна інструменту на аналоги
8	Відкритий	Неправильна конфігурація для мікро сервісного додатку під час розгортання проекту на локальному та тестовому середовищах	Середній	Середній	1	Читання документації та пошук можливих проблем на форумах	Ухилення	Пошук вирішень проблем на форумах

Продовження таблиці Б.10 – Заходи реагування на виявлені ризики проекту

9	Відкритий	Технічні поломки техніки під час розробки	Низький	Середній	1	Підготовка іншого апаратного забезпечення для розробки програмного продукту	Ухилення	Використання апаратного забезпечення університету
10	Відкритий	Розробка непотрібного функціоналу	Середній	Низький	1	Написання чітких та розгорнути критеріїв функціональності додатку та пункти по яким буде зрозуміло, що функціональна одиниця виконана	Зменшення	По етапне уточнення функціоналу із керівником відносно поставлених задач