МІНІСТЕРСТВО ОСВІТИ НАУКИ УКРАЇНИ

СТРУКТУРНИЙ ПІДРОЗДІЛ «ФАХОВИЙ КОЛЕДЖ ЕКОНОМІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ПРАТ «ПВНЗ «ЗІЕІТ»

Циклова комісія з інформаційних технологій

ДО ЗАХИСТУ ДОПУЩЕНА

Голова циклової комісії,

спеціаліст в/к

_____ С.О. Сабанов

# ВИПУСКНА РОБОТА МОЛОШОГО СПЕЦІАЛІСТА СТВОРЕННЯ ВЕБЗАСТОСУНКУ ДЛЯ КОМПЛЕКТУВАННЯ ТА ПРОДАЖУ КОМП'ЮТЕРНОЇ ТЕХНІКИ

Виконав

ст. гр. ІПЗ – 118к9 _____ М.Є. Гриценко

Керівник

доц. _____ О.А. Жеребцов

Запоріжжя

2022

СТРУКТУРНИЙ ПІДРОЗДІЛ «ФАХОВИЙ КОЛЕДЖ ЕКОНОМІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ПРАТ» «ПВНЗ» «ЗІЕІТ»
Циклова комісія з інформаційних технологій

ЗАТВЕРДЖУЮ

Голова циклової комісії,
спеціаліст в/к
_____ С.О. Сабанов
17 січня 2022 р.

ЗАВДАННЯ

НА ВИПУСКНУ РОБОТУ МОЛОДШОГО СПЕЦІАЛІСТА

студенту гр. *ІПЗ-118К9,*

спеціальності: *121 – «Інженерія програмного забезпечення»*

*Гриценко Микиті Євгеновичу*

1.Тема: Розробка вебзастосунку для комплектування та продажу комп'ютерної техніки

затверджена наказом по інституту: № 09.2 від 04 березня 2022 року

2. Термін здачі студентом закінченої роботи: 18 червня 2022 року

3. Перелік питань, що підлягають розробці:

1. *Провести огляд предметної області з метою виявлення абстракцій.*
2. *Провести огляд та аналіз популярних аналогів, зробити висновки про вимоги до проекту.*
3. *Здійснити підбір стеку програмної реалізації застосунку.*
4. *Здійснити проектування архітектури застосунку.*
5. *Створити вебзастосунок, який буде задовільняти відповідним вимогам.*
6. *Здійснити перевірку роботи програмного продукту.*
7. *Оформити звіт за результатами роботи*

Дата видачі завдання: 17 січня 2022 року

4. Календарний графік підготовки дипломної роботи

| № етапу | Зміст | Терміни виконання | Готовність по графіку %, підпис керівника | Підпис керівника про повну готовність етапу, дата |
|---|---|---|---|---|
| 1 | Формулювання (корегування) теми випускної роботи молодшого спеціаліста та збір практичного матеріалу за темою випускної роботи | 17.01.22-26.02.22 | | |
| 2 | **І атестація** І розділ випускної роботи молодшого спеціаліста | 28.03.22-02.04.22 | | |
| 3 | **ІІ атестація** ІІ розділ випускної роботи молодшого спеціаліста | 10.05.22-14.05.22 | | |
| 4 | **ІІІ атестація** ІІІ розділ випускної роботи молодшого спеціаліста, висновки та рекомендації, додатки, реферат | 30.05.22-04.06.22 | | |
| 5 | Перевірка випускної роботи молодшого спеціаліста програмою «Антиплагіат» | 30.05.22-18.06.22 | | |
| 6 | Доопрацювання випускної роботи молодшого спеціаліста, підготовка презентації, отримання відгуку керівника і рецензії | 06.06.22-11.06.22 | | |
| 7 | **Попередній захист випускної роботи молодшого спеціаліста** | **14.06.22-18.06.22** | | |
| 8 | Подача випускної роботи молодшого спеціаліста на кафедру | за 3 дні до захисту | | |
| 9 | **Захист випускної роботи молодшого спеціаліста** | **20.06.22-25.06.22** | | |

Керівник _____ О.А. Жеребцов
(підпис) (прізвище та ініціали)

«____» _____ 2022 р.

Завдання отримав до виконання _____ М.Є. Гриценко
(підпис студента) (прізвище та ініціали)

«____» _____ 2022 р.

# РЕФЕРАТ

Дипломна робота містить 69 сторінок, чотири таблиці, 26 рисунків, один додаток, дев’ять лістингів, дев’ять бібліографічних посилань.

Метою роботи є розробка веб-застосунку для комплектування та продажу комп'ютерної техніки.

Об’єктом дослідження є сучасна система конфігураторів комплектуючих для персонального комп’ютеру.

Предметом дослідження є веб-застосунок для зручної фінансової роботи.

Здійснено детальний огляд предметної області та сучасних аналогів. Виявлено, що розробка веб-додатку зі зборкою комплектуючих для персонального користувача - підприємця є доцільною. Проект реалізовано за допомогою таких засобів як Java-фреймворк Spring Boot із стороннім сервером та JavaScript із використанням бібліотеки JQuery з боку клієнта. Здійснено проектування моделі предметної області, програмування сутностей та алгоритмів на базі клієнт-серверної архітектури.

Програмний продукт є легким у використанні, має привабливий та інтуїтивно зрозумілий інтерфейс, достатній функціонал. Застосунок дозволяє заощаджувати час користувача за рахунок мінімального пошуку підходящих комплектуючих.

Даний програмний продукт стане в нагоді фізичним особам – підприємцям, та іншим учасникам малого бізнесу для зручної зборки та покупки комплектуючих для комп’ютеру.

КОНФІГУРАТОР, ВЕБ-ЗАСТОСУНОК, SPRING BOOT, JAVA, JS, JQUER

ЗМІСТ

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

| Скорочення | Повна назва | Пояснення/переклад |
|---|---|---|
| REST | Representational State Transfer | Архітектурний стиль взаємодії компонентів розподіленого застосування в мережі |
| БД | База даних | Сукупність даних, що зберігаються відповідно до схеми даних, маніпуляція яких виконується відповідно до правил моделювання даних |
| SQL | Structured query language | Мова структурованих запитів |
| NoSQL | Not only SQL | Система управління базо даних. Застосовується у системах, в яких вирішуються проблеми з масштабністю та доступністю даних. |
| MySQL | | Вільна система керування реляційними базами даних, яка була розроблена компанією «TcX» для підвищення швидкодії обробки великих баз даних. |
| MongoDB | | Документоорієнтована NoSQL система управління базами даних. Використовує JSON-подібні документи і схему бази даних. |
| Kafka | | Сервіс, що дозволяє в реальному часі і з високою пропускною здатністю передавати повідомлення між різними системами |
| UI | User Interface | Користувацький інтерфейс |
| URL | Uniform Resource Locator | Уніфікований локатор ресурсів або адреса ресурсу |
| URI | Uniform Resource Identifier | Уніфікований (однаковий) ідентифікатор ресурсу. Універсальний ідентифікатор ресурсу. Ідентифікує абстрактний або фізичний ресурс. |
| PaaS | platform as a service (платформа як послуга) | Категорія послуг хмарних обчислень, дозволяє клієнтам створювати, запускати та керувати модульними пакетами. Складається з платформи та одного або більше додатків. |

| | | |
|---|---|---|
| JMX | Java Management Extensions | Технологія призначена для контролю і управлінням додатками, системними об’єктами |
| Maven | | Система автоматичної зборки проектів |
| Gradle | | Система автоматичної зборки проектів |
| DTO | Data Transfer Object | Шаблон проектування, використовується для передачі даних між підсистемами програми |
| POJO | Pain Old Java Object | Простий Java-об'єкт, не успадкований від якогось специфічного об’єкта і не реалізує ніяких службових інтерфейсів |
| JavaBeans | «біни» | Класи в мові Java, написані за певними правилами. Використовуються для об'єднання декількох об'єктів в один |

# ВСТУП

Україна являє собою величезний ринок збуту товарів для персонального комп’ютеру. Intel, Nvidia, AMD – відомі компанії гіганти, у сфері обчислювальних машин, які випускають усе більше й більше нових моделей свого продукту, збільшуючи їх продуктивність.

Через це збільшується кількість комплектуючих для персонального та робочого комп’ютерів, а також з цим зростає кількість необхідного та розважального програмного забезпечення, яке потребує деяку кількість потужностей комп’ютеру. Так для програмістів та для звичайного користувача виникає необхідність підвищувати ці «потужності» через придбання нових комплектуючих, або нового комп’ютеру, через це зростає необхідність правильного підбору деталей.

Більшість покупок здійснюється через Інтернет, так електронна комерція стала невід’ємною частиною нашого життя. Через необхідність купувати нові компоненти для персонального комп’ютеру, треба враховувати купу важливих речей, для правильного підбору компонентів, або збірки нового комп’ютеру.

Найчастіше при купівлі треба перечитувати їх характеристики, пощастить якщо на сайти де придбаються ці деталі будуть всі необхідні дані, а якщо ні – треба витрачати час на їх пошук. Після того як користувач знаходить підходящі йому товари, перед ним стає питання: «Чи достатньо цих комплектуючих?», «Чи кращі ці деталі за мої?», тощо.

Тому створення веб-застосунку з автоматизації зборки персональних машин та добірки комплектуючих є актуальною задачею.

# РОЗДІЛ 1
# ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ

## 1.1 Огляд предметного середовища

### 1.1.1 Визначення електронної комерції

Електронна комерція («e-commerce») – прискорення більшості бізнес-процесів за рахунок їх проведення електронним образом. Тут об’єднується багато різних технологій, таких як: Інтернет, електронна пошта, обмін інформації усередині компанії та обмін інформацією із зовнішнім світом. Простіше, це ведення бізнесу через Інтернет. Притаманні риси e-commerce: достатність інформації, необмежена кількість покупців і продавців, мінімальні операційні витрати.

Багато компаній в світі вже перевели своїй бізнес на електронний ринок.

### 1.1.2 Веб-застосунок, як Business-to-consumer («B2C»)

Веб-застосунок для продажу комп'ютерної техніки можна визначити як інтернет-магазин, що в свою чергу є програмним продуктом, спеціально розроблений для спрощення покупок та продажу через веб-застосунок. Такий застосунок є орієнтований на продаж кінцевому споживачу, що визначає термін – «Business-to-consumer».

Інтернет-магазин  різновид сайтів, але з набором застосунків які спеціально орієнтовані на електронну комерцію. Продаж через веб-застосунок дозволяє:

- Не мати спеціального приміщенні для продажу;
- Розширити клієнтську базу;
- Не бути прив’язаним до одного самого місця;
- Наймати менше персоналу, або бути зовсім без персоналу;
- Простіше аналізувати продажі;

Але саме тому, що інтернет-магазин це різновид сайтів, він має ті самі ризики:

- DDOS атаки;
- Технічні ризики (Ризики з обладнанням);
- Злом інтернет даних;
- Тощо;

Всі ризики можливо звести до мінімуму якщо найняти професійну компанію-розробника, а використовувати якісне обладнання. Якісне обладнання та регулярний технічний огляд зменшать проблеми з цим самим обладнанням. Професійна компанія-розробник, зробить якісну DDOS захист, та шифрування даних компанії та її користувачів.

## 1.2 Огляд аналогів

Докладніший аналіз аналогічних продуктів дозволить зробити висновки про:

- Кількість та якість сервісів найбільш наближених до того, щоб вирішити поставлене завдання;
- Позитивні сторони цих продуктів, які можна буде запозичити та вдосконалити;
- Необхідні функції REST сервісу для застосунку;
- Які розробки можуть бути використані надалі;
- На скільки затребуваним може бути підсумковий продукт порівняно з аналогами;

Оскільки розроблюваний продукт є веб-застосунком, то перегляд аналогів буде проходити в цьому сегменті. Для цього використовуватися буде пошукова система Google, яка являє собою найпопулярнішу пошукову систему в світі.

Використовуватимуться наступні запити:

- «комп'ютерні комплектуючі»
- «купити комплектуючі для комп'ютеру»
- «buy components for computer»
- «computer components»

Основні критерії веб-сайтів на розгляд:ї

1. Зручність та зрозумілість інтерфейсу.
2. Наявність каталогів.
3. Легка навігація.
4. Детальна інформація про товари.
5. Наявність фільтрів та пошуку товарів.
6. Наявність можливості реєстрації, входу.

Після пошуку та огляду аналогів було обрано три сайти для подальшого детального розгляду та проведення ретельного порівняльного аналізу: Compx, Magazun, Newegg.

### 1.2.1 Compx

Compx – Один з крупніших українських інтернет-магазинів комп’ютерної техніки та комплектуючих (офіційний сайт: https://compx.com.ua/). Надає послуги з продажу комплектуючих для комп’ютеру, периферії для офісу, або ігор, також продає товари із сфери майнінгу. Є наявність придбати вже готову зборку комп’ютеру.

Має приємний зовні інтерфейс, який також зрозумілий та легкий у використанні. Сайт має просту навігацію, містить пошукову строку, яка дозволяє просто та швидко знайти необхідний товар. Товари на сайті мають детальну інформацію. На сайті присутня система реєстрації та входу, яка спрощує купівлю товарі методом того, що товари можна додати до обраного та потім подивитись у своєму кабінеті, також

завдяки своєму акаунту можна робити замовлення наперед, якщо товар закінчився або ще не доїхав до складу. Кожну категорію товарів можна відфільтрувати, що також полегшує пошук необхідних.

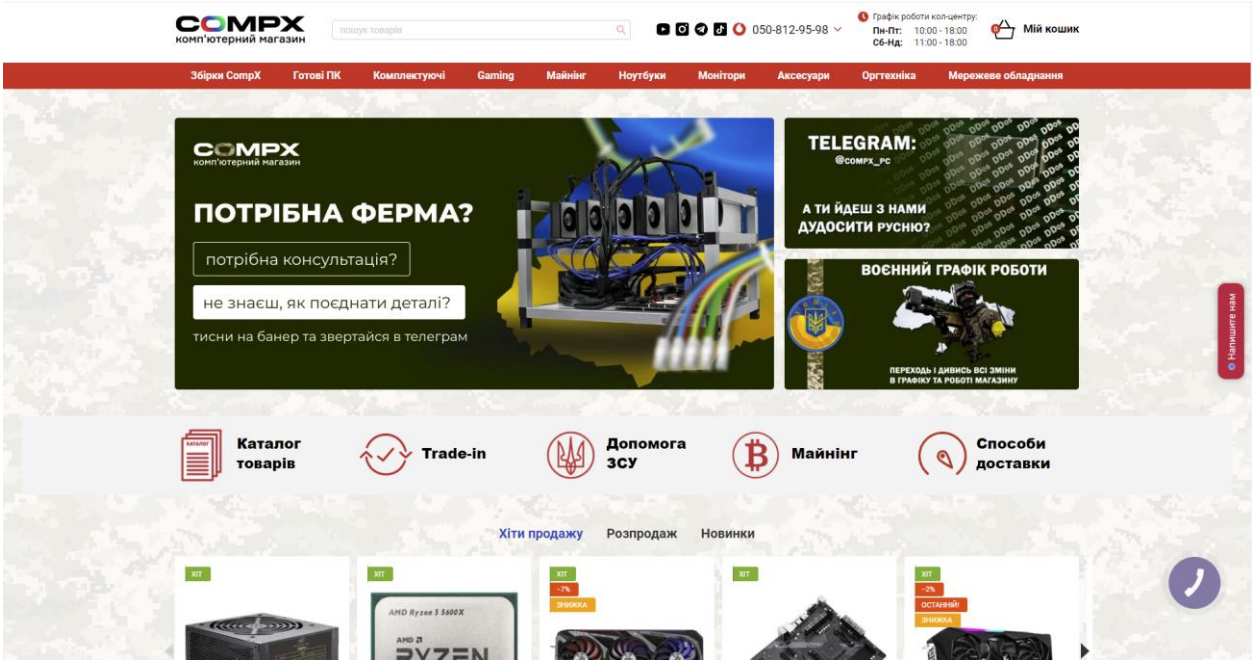



Рисунок 1.1 – Вигляд веб-застосунку compx

Це магазин має великий рейтинг відгуків на українському сайті hotline, а саме: 8.5.

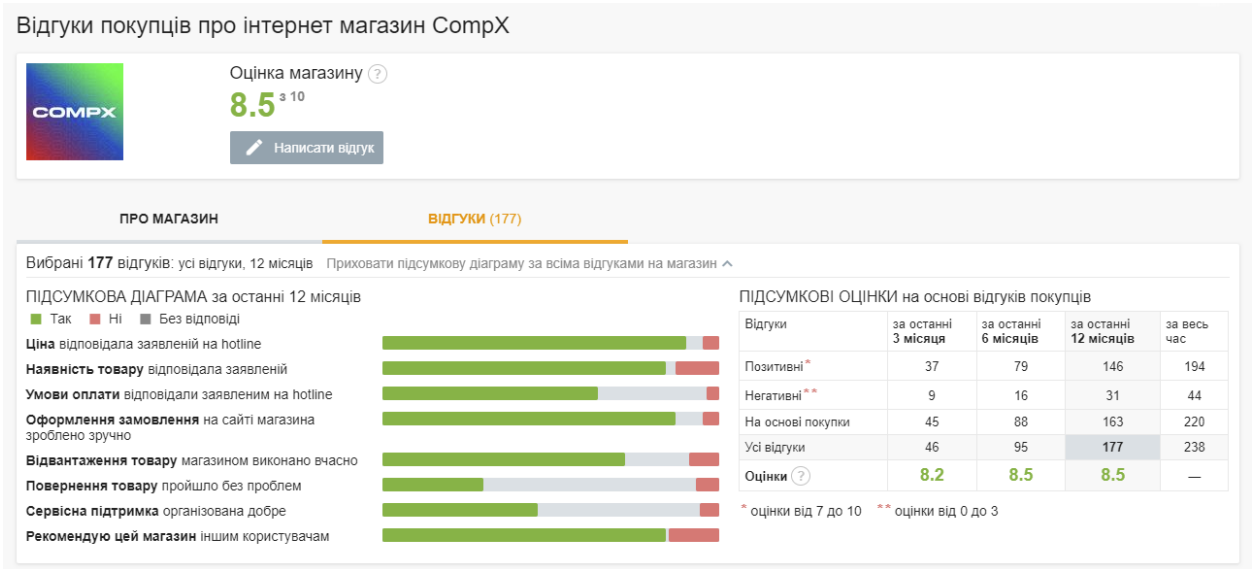



Рисунок 1.2 – Рейтинг сайту від покупців

Плюси застосунку:

- Простий вигляд та проста навігація каталогу товарів;
- Багато інформації про товар;
- Простий спосіб оформлення замовлення;
- Відображення хітів продажу та популярних товарів, категоріїв;

Мінуси:

- Зручність використання фільтрів;
- Місцями не правильний вибір області для фокусу уваги;
- Погана система пошуку за допомоги поля текстового введення;
- Відсутня можливість зборки комп’ютеру з автоматичним підбором підходящих комплектуючих;
- Відсутня можливість швидкого перегляду товару;
- Лише дві локалізації сайту;

### 1.2.2 Newegg

Newegg – північноамериканський інтернет-магазин з великим охопленням у Європі та Китаї (офіційний сайт: https://www.newegg.com/). Надає послуги з продажу електронних товарів, не лише пов’язаних з комп’ютером, а також офісної, інтернет, автомобільної, спортивної, сфер діяльності.

Приємні сторінки на яких потенційного клієнта нічого не відволікає, легка навігація по сайту, багато ненав’язливої, корисної інформації про товари. Наявно створення облікового сторінки, в якій присутня історія замовлень, відвідування, створення та перегляду обраних товарів. Присутні фільтри для дрібного пошуку кожного товару, а також система пошуку.

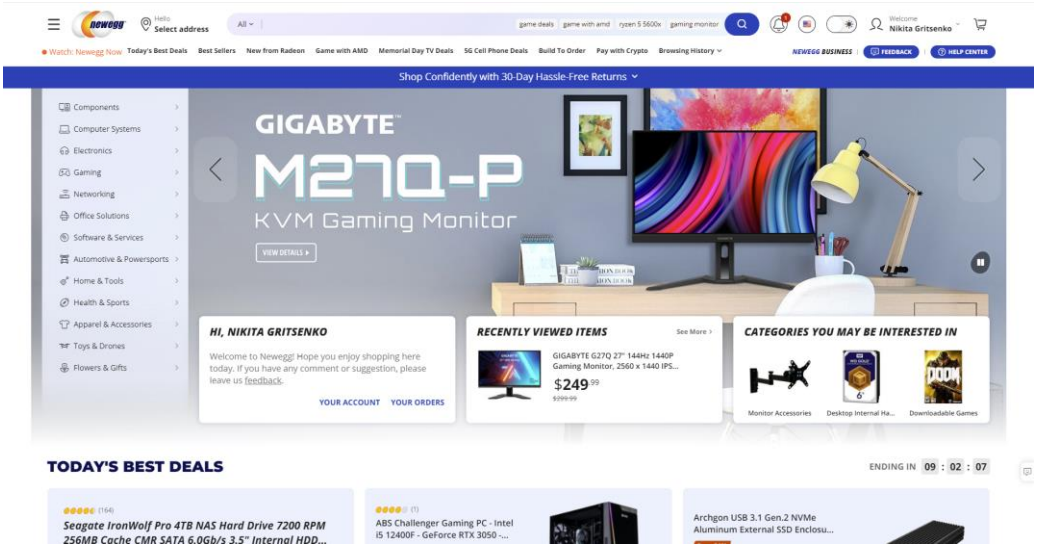



Рисунок 1.3 – Звичайний вигляд застосунку newegg

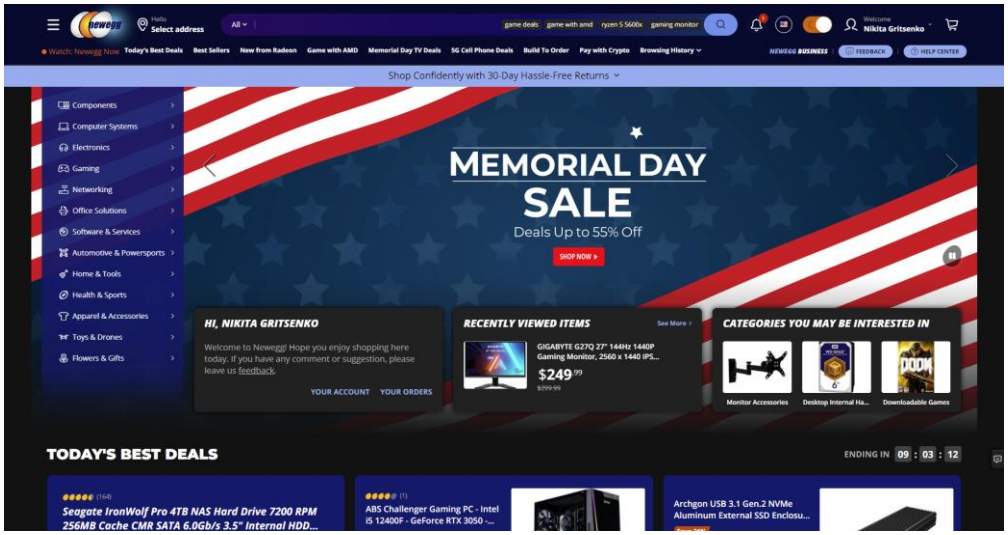



Рисунок 1.4 – Темна тема сайту

Плюси:

- Простий інтерфейс;
- Приємна навігація;
- Легкі до застосування фільтри товару;
- Наявність швидкого перегляду товару;
- Детальний опис інформації та характеристик комплектуючих;
- Багато локалізацій, переводів веб застосунку;

- Детальна система пошуку текстового введення, з підбором підходящих комплектуючих до шуканого товару;
- Відображення хітів продажу та популярних товарів, категорій;

Мінуси:

- Невиправдано важка система облікових записів;
- Відсутня можливість зборки комп’ютеру з автоматичним підбором підходящих комплектуючих;
- Важке оформлення замовлень;

### 1.2.3 Magazun

Magazun – Популярний Український інтернет-магазин комп’ютерної техніки та комплектуючих (офіційний сайт: https://magazun.com/). Має зручний та привабливий інтерфейс, що допомагає зацікавити користувача на перегляд каталогу товарів. Веб-застосунок має детальну навігацію по сайту, а також багато корисної інформації про наявні товари. На сайті наявна детальна фільтрація окремий грум комплектуючих, та рядок для пошуку

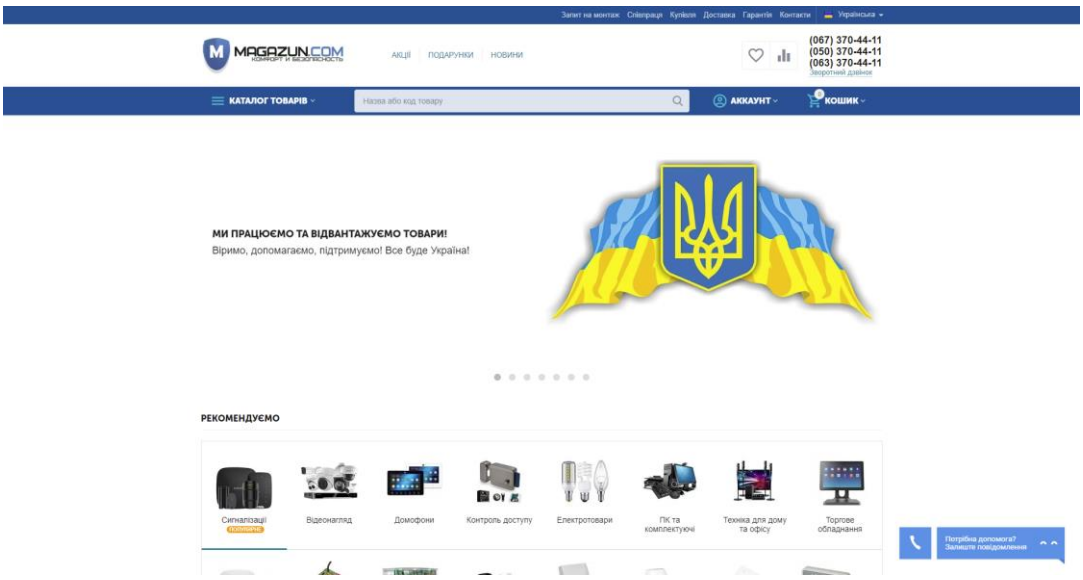



Рисунок 1.5 – Огляд головної веб-сторінки magazun

Відгуки про компанію на сайті hotline гарні, та мають віще середнього рейтинг, а саме: сім і шість з десяти.

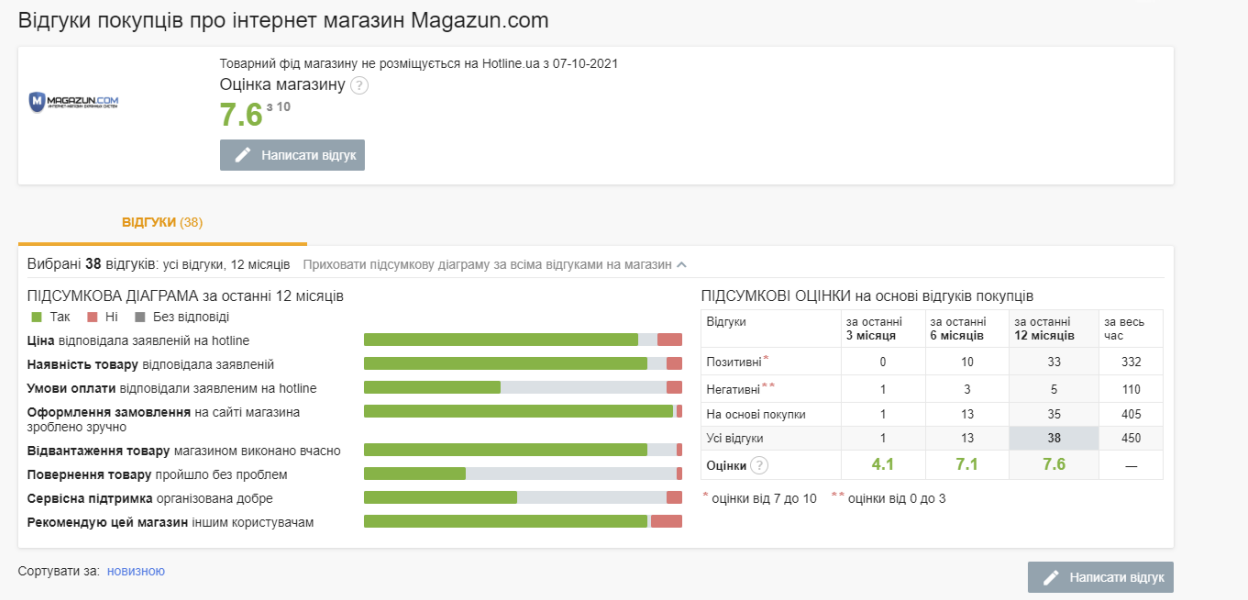



Рисунок 1.6 - Діаграма відгуків сайту magazun

Плюси:

- Простий інтерфейс;
- Легка фільтрація товарів;
- Багато інформації про товар;
- Просте оформлення замовлень;
- Коректний пошук через рядок пошуку;

Мінуси:

- Відсутня можливість зборки комп’ютеру з автоматичним підбором підходящих комплектуючих;
- Відсутня можливість швидкого перегляду товару;
- Лише дві локалізації сайту;
- Навантажене навігаційне меню;
- Відсутнє відображення хітів продажу та популярних товарів або категорій;

- Мало характеристик комплектуючих;
- Довга обробка текстового пошуку через рядок пошуку;

## 1.3 Вимоги базового функціоналу до програмного продукту

Можна виділити ряд даних, на які варто спиратися при подальшому складанні вимог майбутнього застосунку, таких як:

1. Результати аналізу аналогічних продуктів для виділення їх функціональних можливостей, аспектів, які можна запозичити.

2. Вимоги до реалізації функціоналу сервісу, того як він повинен реагувати та на які повинні відповідати запити.

Після обробки всього обсягу представлених даних, були виділені основні вимоги до функціоналу застосунку, що розробляються:

1. Наявність базового функціоналу у вигляді можливостей створити акаунт, вхід до застосунку, які забезпечують швидке створення замовлень та безпеку даних про замовлення.

2. Наявність легкої навігації по застосунку.

3. Надання користувачеві зручний і зрозумілий універсальний сервіс, не перевантажений зайвою інформацією.

4. Наявність конструктору комп’ютера з автоматичним підбором підходящих комплектуючих.

5. Коректний пошук через рядок пошуку.

6. Інформаційну достатність товарів, та детальну характеристику комплектуючих.

7. Функціоналу для додання товарів та аналітиці даних про замовлення та прибуток.

Виходячи з наступних вимог було здійснено рішення щодо функціоналу майбутнього застосунку:

8. Наявність необов’язкової авторизації та реєстрації.

9. Наявність наступних обов'язкових сторінок:

5.1 Головна сторінка, на якій повинна розташовуватися популярні товари та каталоги, категорії товарів, зворотній зв’язок, де й будуть створюватись нові поїздки.

5.2 Сторінка з каталогами окремих груп товарів.

5.3 Сторінка з конструктором комп’ютеру.

5.4 Відокремлена сторінка для адміністрації, аналітиків та власника, де буде розташовуватись інформації про замовлення, прибуток та діаграми загальної інформації.

5.5 Сторінки окремих товарів.

5.6 Сторінка створення замовлення, купівлі.

5.7 Фільтрів для кожних товарів.

5.8 Система текстового пошуку через рядом пошуку в навігаційному меню.

## 1.4 Висновки за розділом

Здійснено порівняльна характеристика існуючих аналогів, та встановлено їх переваги для реалізації.

Висунуті базові вимоги до функціоналу розробляємого продукту з метою спрощення порядку документального оформлення та розуміння плану розробки застосунку.

# РОЗДІЛ 2
# ЗАСОБИ ТА МЕТОДОЛОГІЇ РОЗРОБКИ ДОДАТКУ

## 2.1 REST архітектура

REST сервіс – сервіс використовується для передачі стану представлення. Інакше кажучи він визначає стиль обміну даних між різними клієнтськими частинами.

Для доступу до різних ресурсів сервісу, він повинен мати ідентифікатор – URI.

Якщо є REST сервіс, та клієнт (застосунок який використовує сервіс), то ця архітектура називається – Restful.

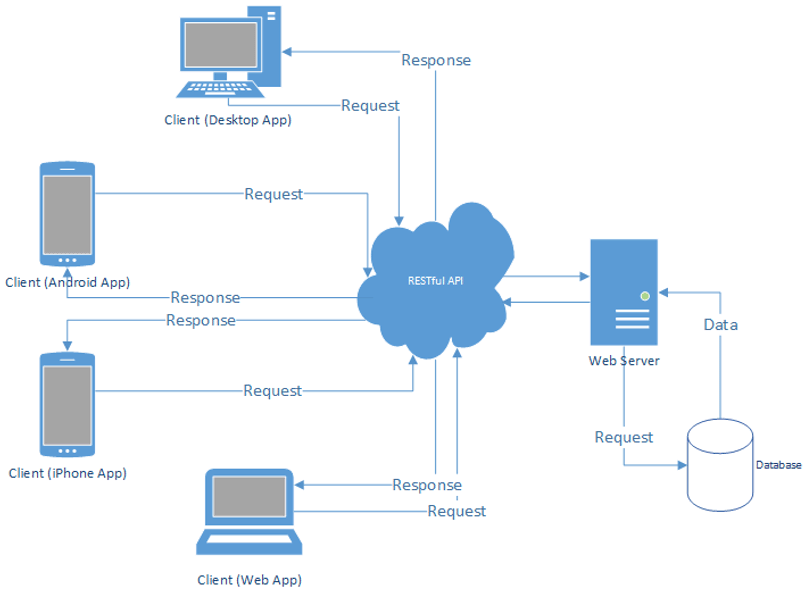



Рисунок 2.1 – Принцип роботи Restful архітектури

### 2.1.1 REST stateless та stateful

Для того, щоб архітектура була повністю Restful, вона не повинна зберігати стан користувача, або його дані на самому серверу, тобто бути у стані – stateless.

Щоб зрозуміти концепцію, клієнт відправляє запит до сервісу, серверу, в якому знаходиться URL адрес запиту, вказуючи на яку частину він хоче отримати доступ.

Наприклад до інформації про свої замовлення, для цього він потрібен додати до відправляючихся даних, свої дані користувача. Тоді сервер порівняє дані користувача до даних к яким він намагається отримати доступ, це – stateless.

А якщо, користувач до цього авторизувався у клієнті, і ці дані зберіглись в базі даних через сервер, та зберігаються на сервері напряму. Після чого він намагається отримати доступ до своїх заказів, но вже без передачі своїх авторизованих даних. При обробці запиту сервер знає сам, що за користувач до нього звернувся, завдяки тому, що зберігає ці дані сам у собі, то це приклад – stateful архітектури. Така архітектура взаємодії не є Restful, та є неправильною.

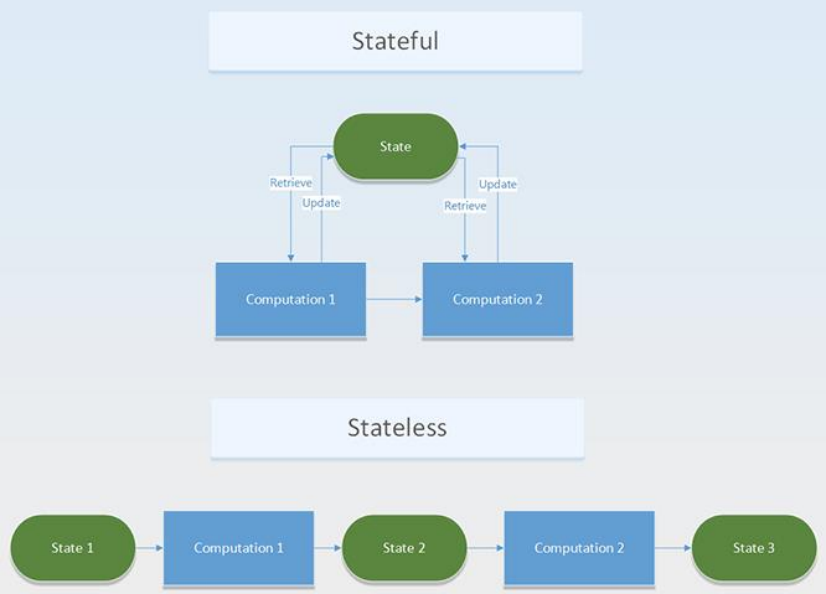



Рисунок 2.2 – Робота архітектур Stateless та Stateful

Простими словами, для stateful авторизований користувач, обмежується своїм станом на сервері, та для доступу до них не відправляє ні які підтверджувані дані. А для stateless авторизований користувач, для доступу до своїх даних, повинен відправляти підтверджувані дані, інакше отримає відмову.

### 2.1.2 Restful архітектура

Restful повинна мати само-документовані повідомлення, тобто запит й відповідь повинні зберігати в собі усі необхідні дані для обробки.

Передача в Restful архітектурі виконується завдяки: HTTP методу, URI ідентифікатору, заголовку (application/json, text/plain, тощо), та тілу у форматі вказаному в заголовку.

Таблиця 2.1 – Передача в Restful архітектурі

| Метод | URI | Заголовок | ДІЯ |
|---|---|---|---|
| GET | /api/order | Authorization, Accept: application/json | Отримати свої замовлення. Перевірка за авторизацією |
| GET | /api/admin/order | Authorization, Accept: application/json | Отримати всі замовлення. Перевірка за авторизацією |
| GET | /api/order/1 | Authorization, Accept: application/json | Отримати одне замовлення |
| POST | /api/order | Content-Type: application/json | Додати, зробити, замовлення |
| PATCH | /api/order/1 | Authorization, Content-Type: application/json | Змінити замовлення |
| DELETE | /api/order/1 | Authorization | Видалити замовлення |

## 2.2 Вибір інструментарію

### 2.2.1 Мова програмування Java

Java – це гарно структурована мова програмування, з об’єктно-орієнтованою структурою, яка розроблена так, щоб мати якомога більше можливостей реалізації. Це мова програмування загального призначення, розробники можуть писати та запускати застосунки на будь-якій платформі, від Windows до Linux з Android.

Синтаксис Java «Сі-подібний», мова надає динамічні можливості (такі як відображення і зміна середовища виконання), зазвичай недоступні в компільованих мовах. На ній написано багато веб-додатків: від ecommerce-проектів до великих порталів, від освітніх платформ до урядових ресурсів. Наприклад використовується в популярних веб-застосунках: Google, Amazon, Twitter, Youtube, та інших. Швидко працює з Big Data направленням (Обробці великої кількості даних).

Особливості та переваги мови:

- Незалежний від платформи;
- Дотримується концепції ООП;
- Багаті бібліотеки з відкритим кодом;
- Автоматичне звільнення пам’яті;
- Портативність;
- Об’єктно-орієнтована мова;
- Захищеність;
- Багато-потоковий (Multi-Threaded);

Для написання REST API Java підходить як не найкраще, статистика мов для написання REST сервісів (За ресурсом: “insights.stackoverflow.com”, рисунок 3):

- Мова Java (5-е місце в рейтингу за популярністю мови, після SQL, Python та HTML/CSS з JavaScript), якщо прибрати мови для стилізації та розмітки залишивши

лише back-end сторону, то конкурує за перше місце з Python. Популярний фреймворк для back-end: Spring Boot.

- Мова JavaScript (1-е місце в рейтингу за популярністю мови) популярний фреймворк для back-end: Node.js.

- Мова Python (3-е місце в рейтингу за популярністю мови) з популярним фреймворком: Django.

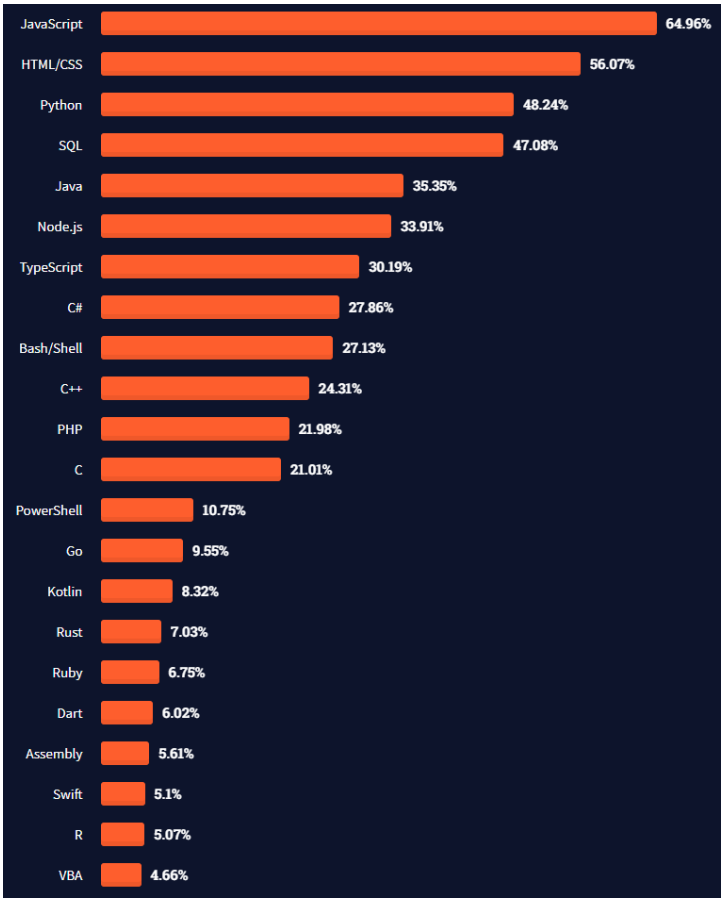



Рисунок 2.3 – Статистика популярних та найчастіше використовуваних мов програмування 2021

Якщо обирати кращу мову для написання back-end веб-застосунку, треба порівняти ще популярні фреймворки цієї сфери з цих мов. Для прикладу будемо дивитись на рейтинг найпопулярніших фреймворків серед професійних програмістів (За ресурсом «insights.stackoverflow.com», рисунок 2.4).

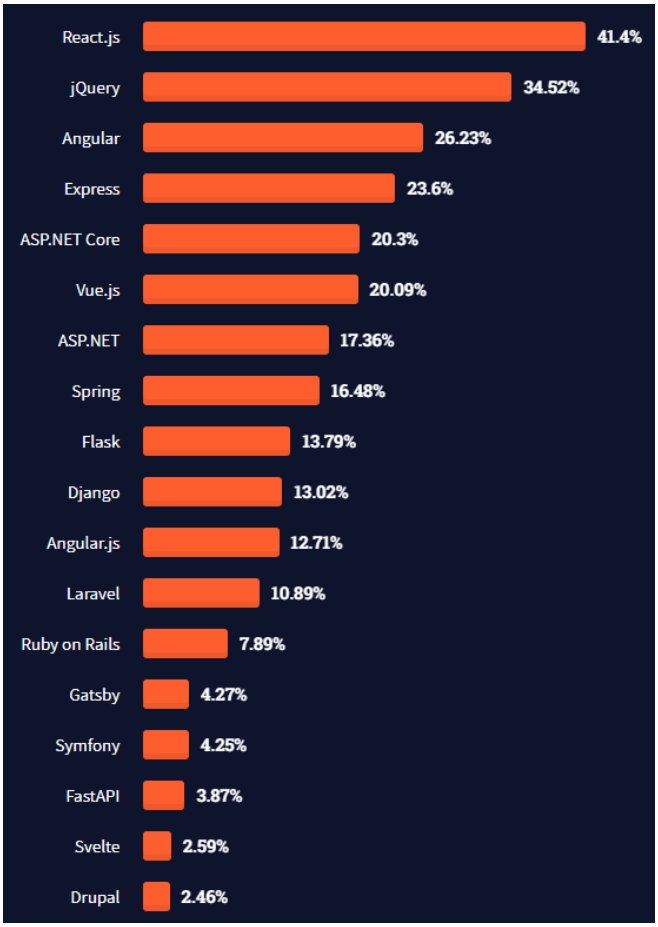



Рисунок 2.4 – Рейтинг найчастіше використовуваних фреймворків 2021

За рейтингом Java з фреймворком більше використовується ніж Python з фреймворком Django. Тож обрання мови програмування Java є розумним вибором.

### 2.2.2 Spring Boot

REST сервіси часто потребує роботу з Big Data, як і в нашому випадку, наприклад при роботі з додаванням точок на карті або їх видаленням.

На даний момент Фреймворки Java для REST сервісу:

- Apache CFX
- Jersey
- Restlet
- Spring Boot

- Quarkus

#### 2.2.2.1 Apache CFX

Apache CFX – фреймворк, з відкритим вихідним кодом, який дозволяє створювати REST сервіси. Для створення сервісу потрібно налаштувати багато XML конфігурацій, порівняно з фреймворками Spring, який налаштовую конфігураційні файли самостійно, це є більш трудозатратною частиною.

Також Apache CFX частіше використовують задля малих бізнесів, коли Spring Boot використовують більш професійні, більші бізнеси.

За статистикою веб сервісу «g2» (“www.g2.com”). Spring Boot відповідає їх потребам бізнесу краще, а також якість підтримки продукту.

#### 2.2.2.2 Jersey

Інша технологія для створення REST сервісу, завдяки зростанню популярності, а також розвитку Spring Boot, цей фреймворк почав вважатись «вмираючим», який не поспіває за його конкурентами. Підходить для легких REST сервісів, які мало навантаженні. Є більш важким для підтримки своєї продукції.

#### 2.2.2.3 Restlet

Як і Spring забезпечує Java анотації для забезпечення простої обробки HTTP-запитів та відповідей з прив’язуванням їх до коду, але не підтримує особливості Spring, такі як Spring Security, це значить що все це потрібно реалізовувати розробнику самостійно.

#### 2.2.2.4 Spring Boot

Фреймворк технології Spring, підтримує усі особливості самої технології. Автоконфігурує необхідні компоненти, з підтримкою їх перевизначення, якщо потрібно.

Ідеально підримує REST сервіси.

Дозволяє просто й зручно створити сервіс, за 5-6 строчок коду.

Дозволяє необов’язковим надавати метрики та JMX, можна лиш оголосити Maven або Gradle.

Автоматично упаковує у jar файл, або у веб-контейнер.

Добре працює з технологією docker.

За замовчуванням використовую JSON, автоконвертує запит JSON у Java об’єкт.

Обробляє веб-контейнери.

Відмінна документація та підтримка на форумах. Дозволяє працювати з більшістю шаблонів коду (наприклад: YAML, properties, XML). Надає автоконфігурацію.

Порівняно з Quarkus, надає швидші операції по введенню-виводу, а також більше безпеко захищений.

#### 2.2.2.5 Quarkus

Використовує Substrate VM замість JVM (“Java Virtual Machine”), а також Graal, завдяки цьому застосунки на Quarkus завантажується швидше, потребують менше пам’яті та навантаження на процесор.

Важке завантаження Graal VM, невелика, порівняно зі Spring, документація та допомога на форумах, таких як stackoverflow. Є більш молодим фреймворком ніж Spring.

Таблиця 2.2 – Різниця Spring Boot та Quarkus [6]

| | Spring Boot | Quarkus |
|---|---|---|
| Впровадження залежностей | Використовує CDI. В даний час реалізовано тільки підмножина функцій CDI | Використовує надійний Контейнер для впровадження залежностей |
| Збереження даних | Використовує знайомі фреймворки, такі як Hibernate | Заснований на абстракції Spring Data |
| Час завантаження програми | Швидкий час завантаження | Більш повільний час завантаження - як правило, повільніше, ніж у проектів, створених на Java Enterprise, через Абстракції зверху Spring Framework |
| Споживання пам'яті | Менше споживання пам'яті при завантаженні і при великому навантаженні | Більш високе споживання пам'яті |
| Зрілість | Новий фреймворк - документація та спільнота менш активні | Набагато більш зрілий, з відкритим вихідним кодом і багатим функціоналом. Відмінна документація та підтримка спільноти |

### 2.2.3 Середа розробки IntelliJ IDEA Ultimate

На даний момент популярними середовищами розробки під Java є: IntelliJ IDEA (Community та Ultimate), Eclipse, NetBeans.

Якщо відштовхуватись від рейтингу на платформі stackoverflow, то IntelliJ IDEA лідирує, находячись на четвертому місці в рейтингу, по популярним середовищам програмування з усіх мов.

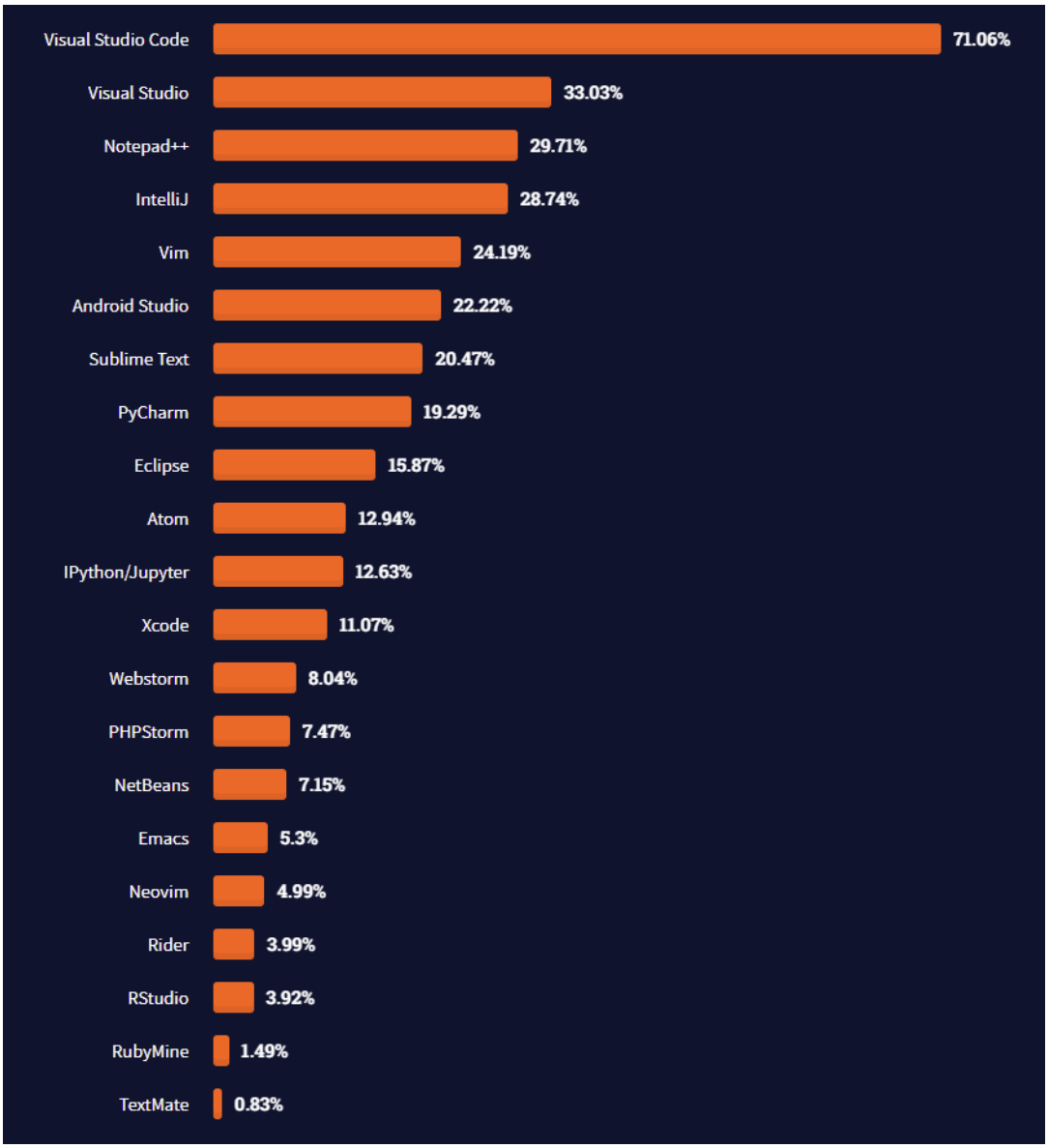



Рисунок 2.5 – Популярні середовища програмування

IntelliJ IDEA – надає як безкоштовну версію так і платну, в якій більш розширений функціонал.

Саме середовище дозволяє як маніпулювати візуальною складовою, такою як шрифти, кольори, тощо, так і легко додавати та й керувати різними плагінами, з додатковою підтримкою всім необхідним до них. У платній версії вже присутні деякі плагіни.

Переваги Intellij IDEA Ultimate, перед конкурентами:

- Тестувати, перевіряти, писати REST запити
- Писати SQL запити
- Підключатись до бази даних
- Легко писати міграцію баз даних.
- Генерувати однотипний код (Такий як Getter та Setter)
- Легко й зручно рефакторити код
- Знаходити й виправляти різні помилки коду
- Підтримка багатьох інших мов програмування та розмітки
- Розумне доповнювання та рекомендації коду
- Зручна підтримка до автосборок проектів
- Зручна підтримка для Spring Boot
- Допомога з маніпуляцією Big Data

Тож Intellij IDEA це розумний вибір для кожного програміста та кожного офісу Java розробників.

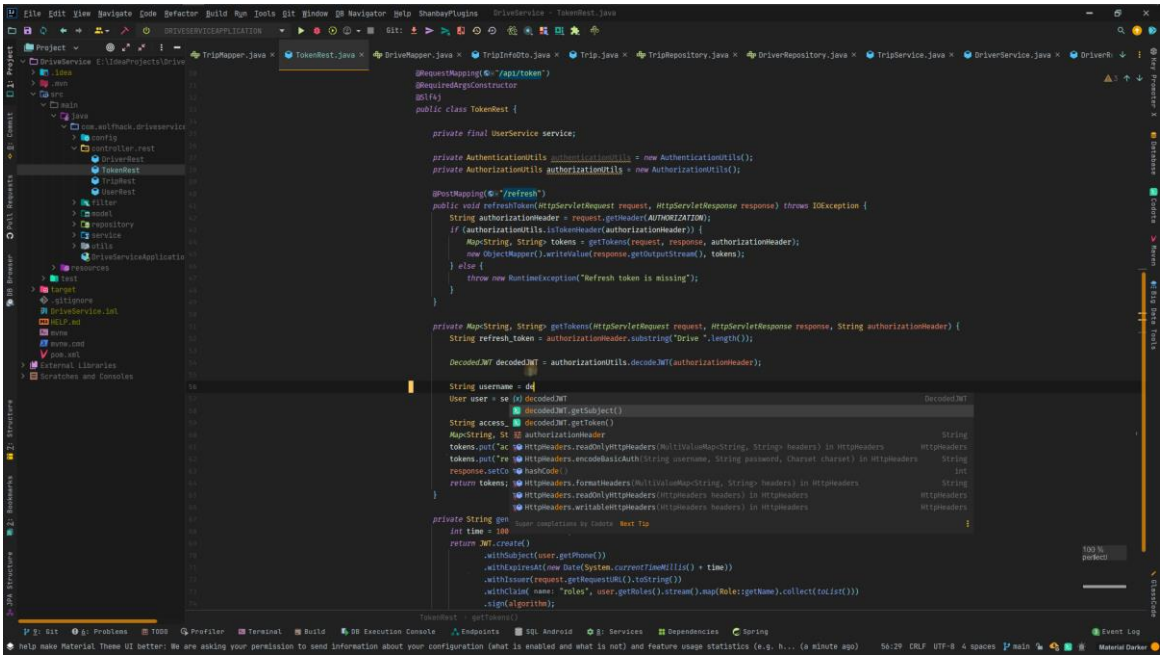


Рисунок 2.6 – Вигляд середовища програмування

### 2.2.4 СУБД MySQL

База даних – це набір деякої інформації, що зберігається у впорядкованій формі. База даних для програми необхідна для довготривалого зберігання інформації та організації її за певними правилами. Для створення, ведення та звернення до БД використовується СУБД.

Для створення застосунку була обрана СУБД MySQL.

MySQL - реляційна база даних (Дані зберігаються в виді таблиці). Система MySQL вважається гнучкою і простою в використані, також швидка в своєї роботі, підходить для управління великими базами даних.

Підтримує можливість криптографії, роботу з великим текстом, функцію «REGEXP» (забезпечує гнучке зіставлення з шаблоном регулярного виразу), та повнотекстове індексування полів «VARCHAR» та «TEXT».

MySQL є серверною СУБД. Позитивно її виділяє серед інших СУБД широке поширення, простота роботи, висока швидкість роботи та обробки даних.

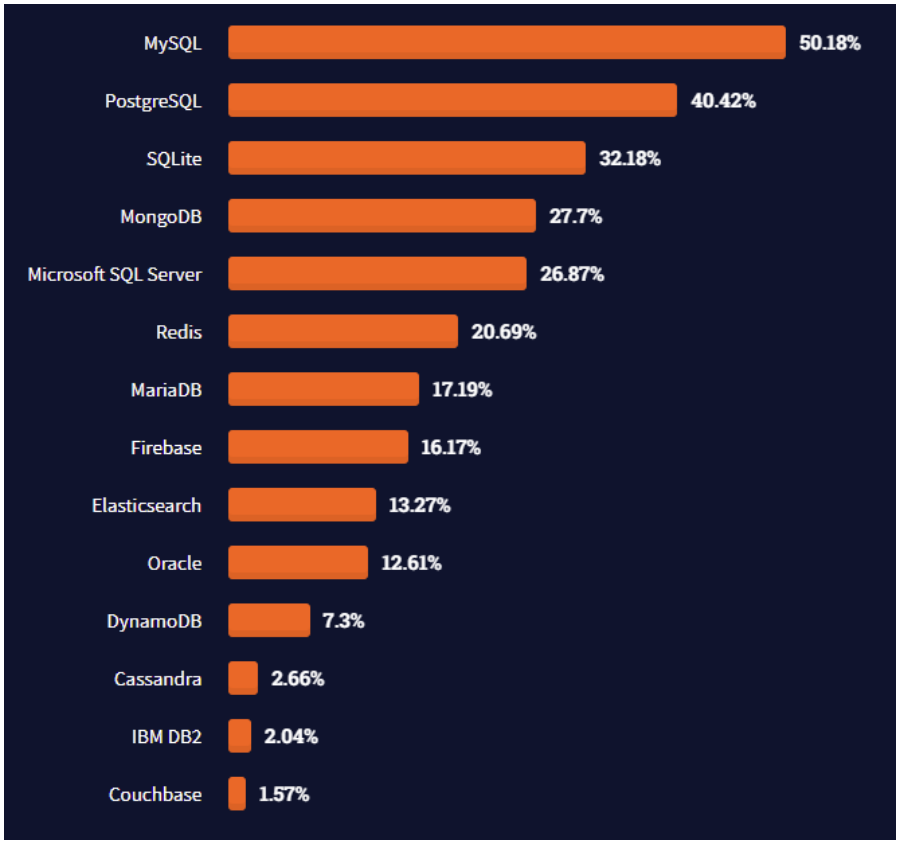



Рисунок 2.7 – Популярність СУБД

Використання у якості СУБД MySQL є найбільш поширеним у світі, цей показник становить половину від загальної кількості (Рис. 2.7).

### 2.2.5 ДСУБД MongoDB

Документо-орієнтована система управління базою даних (NoSQL субд, документо-орієнтовна СУБД) – база даних працює на основі вилучення та управління документально-орієнтованою інформацією, реалізується за допомогою NoSQL підходу. Позитивно її виділяє: зручна та швидка робота з великим потоком даних, та загальним пошуком по базі даних, завдяки простого функціоналу пошуку та швидкості роботи обробки даних та запитів, а також малим споживанням ресурсів комп’ютеру.

Окрім документо-орієнтованих NoSQL баз даних є ще бази даних на основі графів. Вони представляють дані у вигляді набору вузлів (елементів даних),

об’єднаних ребрами. Вузли містять фрагменти даних, в той час ребра представляють відношення між ними. Зазвичай цю базу даних використовують для представлення зв’язків між записами даних, наприклад як працюють соцмережі. Тож цей підхід не може бути використаний в e-commerce системі магазину.

Серед NoSQL документо-орієнтовних баз даних, най популярніші бази даних це: MongoDB та Elasticsearch.


Рисунок 2.8 – Приклад вигляду даних

MongoDB – використовує пари ключ-значення, які зберігає у формі JSON документів. За допомогою цього поля можуть відрізнятись від документа к документу, а структура даних змінюватись з часом. Модель документа завжди зіставляється з об’єктами у коді застосунку, це спрощує роботу з даними.

Після зберігання ці документи групуються у колекції, в залежності від їх вмісту та використання.

MongoDB легко масштабується з зростанням даних та проектом, а також має географічне розподілення, що дозволяє зіставляти дані з точками на карті. Все це робить її легкою, гнучкою та структурною, а також використовувати з початку проекту та довгі роки.

Elasticsearch – розподілена база даних, працює як пошукова система та система аналітики Restful, здатна виконувати повнотекстовий пошук, навіть якщо потрібно шукати частину з тексту, може виправляти помилки, централізовано зберігає дані за для їх миттєвого пошуку, точної настройки релевантності та ефективної аналітики.

Дозволяє виконувати різні типи пошуку, а також їх комбінації, наприклад: структурований пошук, неструктурований, метричний та географічний.

Таблиця 2.3 – Плюси використання

| Elasticsearch | MongoDB |
|---|---|
| Можна розширити функціонал додавши такі інструменти як: Kibana (аналітика), Logstash (збір даних з подальшим аналізом журналів) та іншими. | Легко масштабується з простих систем до складних (Де потрібен лише один сервер, та де більш складні системи). |
| Масштабований пошук у реальному часі з огранюванням та процесом фільтраці. | Постійно забезпечує високу продуктивність та швидкість. |
| | Висока надійність завдяки самовідтворенню (реплікації) та балансуванню навантаження. |

### 2.2.6 Операційна система Ubuntu 22.04

Ubuntu – операційна система, для комп’ютерів, ноутбуків та серверів, найпопулярніша система Linux, та одна з найпопулярніших дистрибутив. Основні цілі – надання сучасного й водночас стабільного програмного забезпечення для пересічного користувача із сильним акцентом на простоту встановлення та користування [7].

Linux система підходить для використання як сервер, завдяки своїй стабільності, надійності, гнучності, а також безпеці даних

Плюси використання Linux Ubuntu:

- Добре захищений від вірусів, ніж Windows.
- Стабільніший за Windows.
- Ядро на базі Linux більш потужна і універсальна, ніж Windows, вона легка, що робить її швидше, ніж Windows.
- Linux має вбудовану підтримку SSH. За допомогою цього можна легко управляти серверами.
- Linux має більше можливостей терміналу та легше маніпулювати їм, ніж командним рядком Windows.
- На Linux набагато легше працювати з програмним забезпеченням Docker.

### 2.2.7 Контейнеризація додатків docker

Docker — програмне забезпечення для автоматизації розгортання та керування додатками в середовищах з підтримкою контейнеризації [8]. Є собою набір продуктів платформи Docker, як послуги («PaaS»), використовує віртуалізацію на рівні операційної системи, доставляє програмне забезпечення в пакетах, які мають назву – контейнери.

Docker дуже поширено використовується в розробці веб-застосунків серед професійних програмістів, та IT компаній, але не тільки в цій сфері розробки.

Система контейнеризації створена для швидкої, легкої та портативної розробки додатків — настільних і хмарних. Комплексна кінцева платформа Docker включає інтерфейси користувача, CLI, API та інтерфейси безпеки, які розроблені для спільної роботи протягом усього життєвого циклу доставки додатків [9].

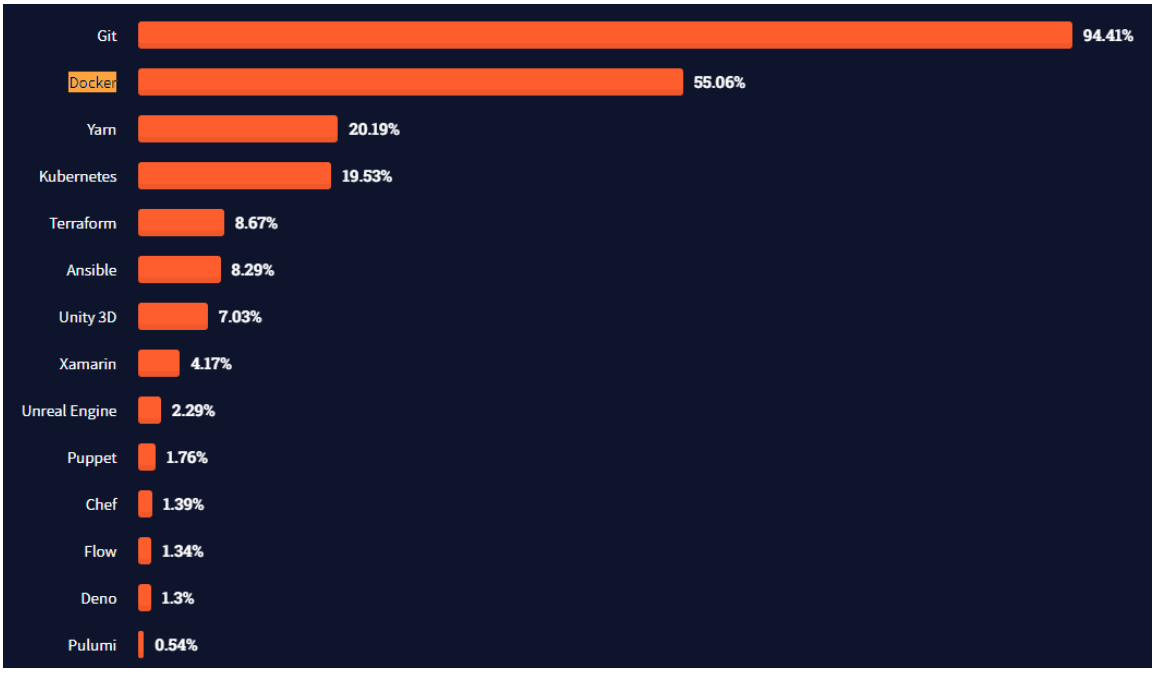



Рисунок 2.9 – Популярні професійні інструменти для розробки застосунків 2021

Використання Docker дозволить зробити застосунок більш портативним, зручним, легким та дозволить працювати швидке, займаючи менше місця на комп’ютері. Добре працює з Linux-подібними архітектурами операційних систем, але може використовуватись й з системою Windows та MacOS. Але розкриває свій повний потенціал з архітектурами Linux.

## 2.3 Висновки за розділом

Архітектура продукту буде стандартною як для e-commerce застосунків, а саме тришаровою: клієнт, сервер, бази даних.

Враховуючи загальні тенденції розробки програмних продуктів було обрано створення веб-застосунку на базі мови Java з середовищем розробки було взято найпопулярнішу, у цій сфері, середовище програмування – IntelliJ IDEA Ultimate.

Для розробки REST сервісу було обрано фреймворк Spring Boot, який є зрілим, добре відомим та стабільним. Він володіє широкими можливостями, шаблонами проектування і відрізняється високим ступенем безпеки. Він також має відмінну документацію та підтримку спільноти, яка допомагає вирішувати більшість проблем.

За СУБД було обрано – MySQL, як базу даних для збереження даних пов’язаних з замовленнями, системою облікових записів, та інших невеликих даних.

За ДСУБД було обрано – MongoDB, як базу даних для збереження великої кількості даних з якою потрібно буде працювати постійно, майже при кожному запиті в застосунку – товарами, комплектуючими. Це дозволить зробити швидкий та легкий пошук по товарам, що також полегшить роботу з конструктором пк.

Розробка проекту буде вестись на операційній системі Linux – Ubuntu.

# РОЗДІЛ 3
# ПРОГРАМНИЙ МОДУЛЬ

## 3.1 Проектування системи

На рисунку 3.1 зображено діаграму прецедентів створюваного застосунку.

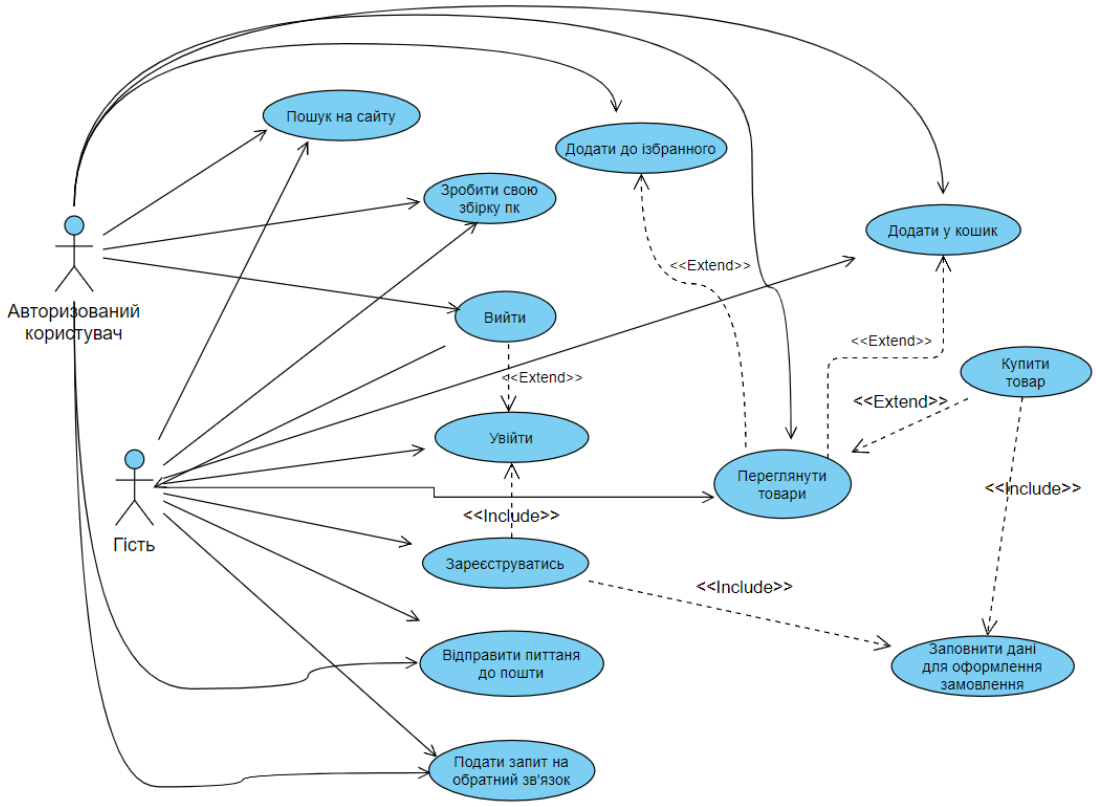



Рисунок 3.1 – Діаграма прецедентів

Діаграма містить два базових акторів: користувач та гість.

Гість є незареєстрованим користувачем з майже повним функціоналом, що складається з можливості пошуку, створення замовлень, покупки товарів, додати товар до кошика, відправки заявок на зворотній зв’язок через телефон або пошту,

зборки, підбору комплектуючих через конструктор комп’ютеру, також гість має можливість увійти або зареєструвати обліковий запис.

Авторизований користувач має розширений функціонал. Функціонал такий самий як й у гостя, але зареєстрований користувач заповнює дані після реєстрації, що у подальшому полегшить заповнення даних при оформлені замовлень, також авторизований користувач може зберігати товари у обране, щоб переглянути їх пізніше.

На рисунку 3.2 можна побачити діаграму прецедентів адміністратора

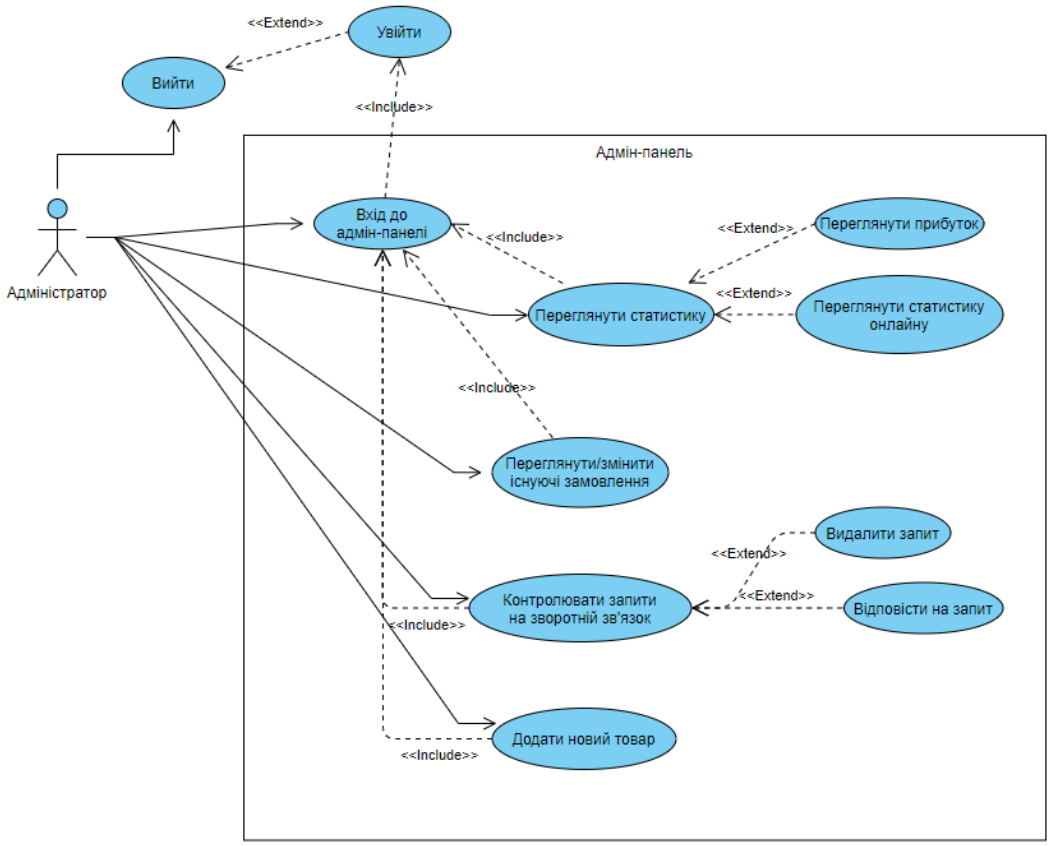



Рисунок 3.2 – Діаграма прецедентів адмін-панелі

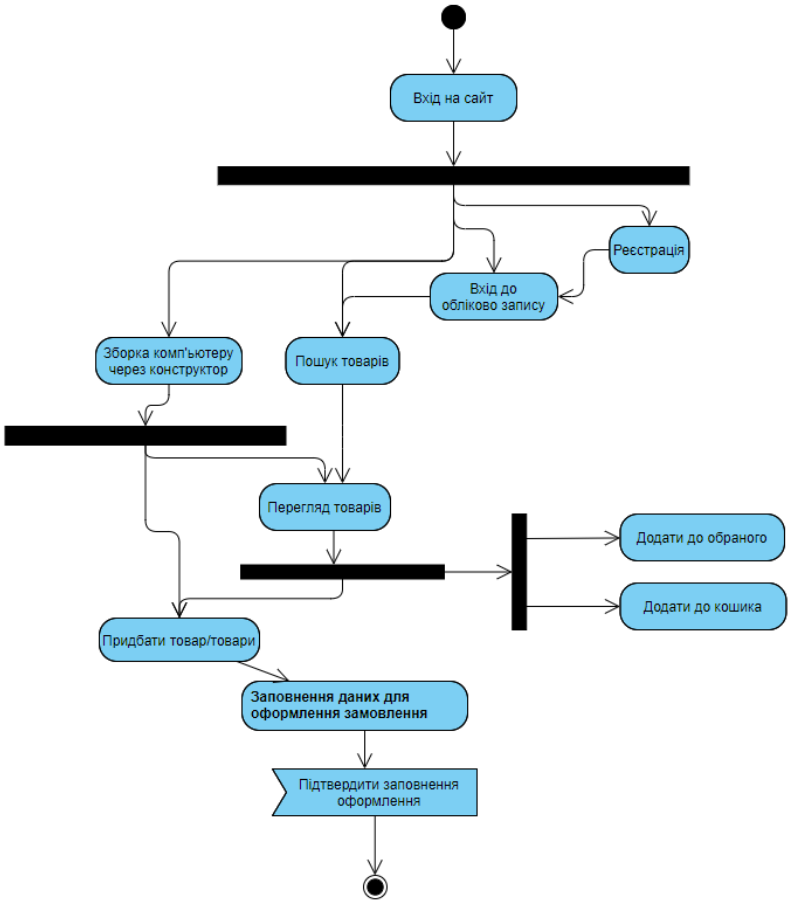



Рисунок 3.3 – Діаграма активності застосунку

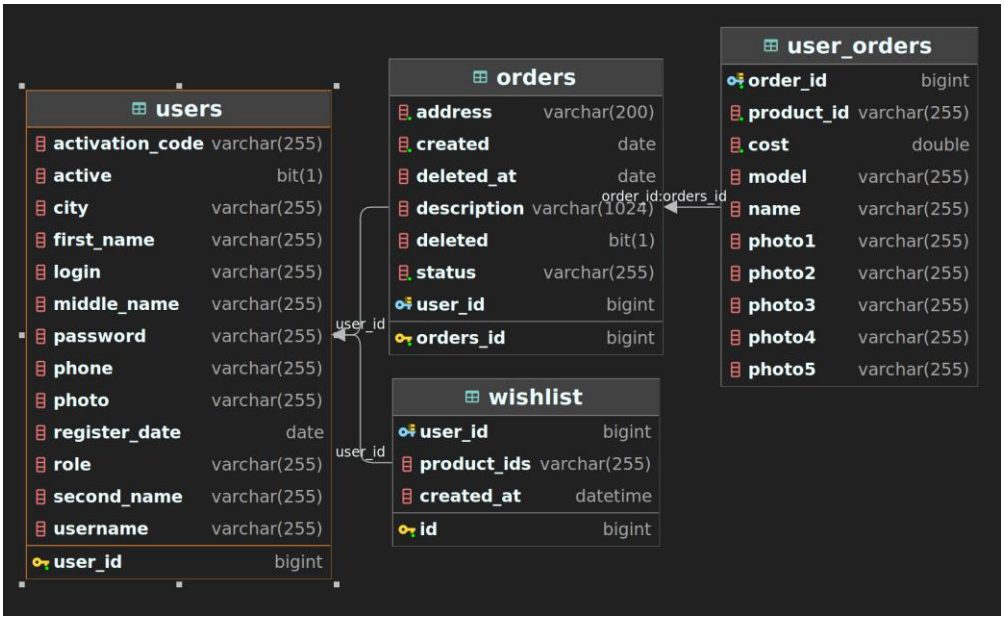



Рисунок 3.4 – Діаграма бази даних користувачів та замовлень

На Рисунку 3.4 зображено діаграму бази даних, яка зберігає інформацію про користувачів та їх ролей, а також про замовлення.

На діаграмі знаходяться три схеми, а саме: схема користувача, замовлень, та схема замовлень певного користувача. Схеми пов’язані таким чином: замовлення мають певного, свого замовника, але товарів які замовляють можуть бути багато, для цього існує третя схема.

Таблиця користувачів зберігає інформацію про користувача або замовника. Користувач має логін та пароль для входу, рядок який показує чи активована облікова сторінка, яка активується через активаційний код, який надходить до користувача на пошту. Якщо замовник не авторизований то ці поля будуть для нього пусті, база даних запам’ятає користувача. Завдяки цим даним замовник зможе простіше оформлювати замовлення.

Таблиця замовлень зберігає при собі інформацію того, коли було створено замовлення, його статус опис та адресу. Коли замовлення виконалось, прибуло до замовника, воно видаляється, але видаляється лише для користувача, а з бази даних ні, а також створюється дата коли замовлення було видалено. Поки замовлення йде до замовника, його можна перевірити у обліковій сторінці, або створити обліковий запис за електронною адресою, за яким було створено замовлення, та побачити його вже тоді.

Як можна бачити на діаграмі відсутня схема товарів, але присутня стовпець з id товару. Це тому, що база даних з товарами зберігається на іншій СУБД, а саме MongoDB, діаграма бази даних продуктів можна побачити на рисунку 3.5.

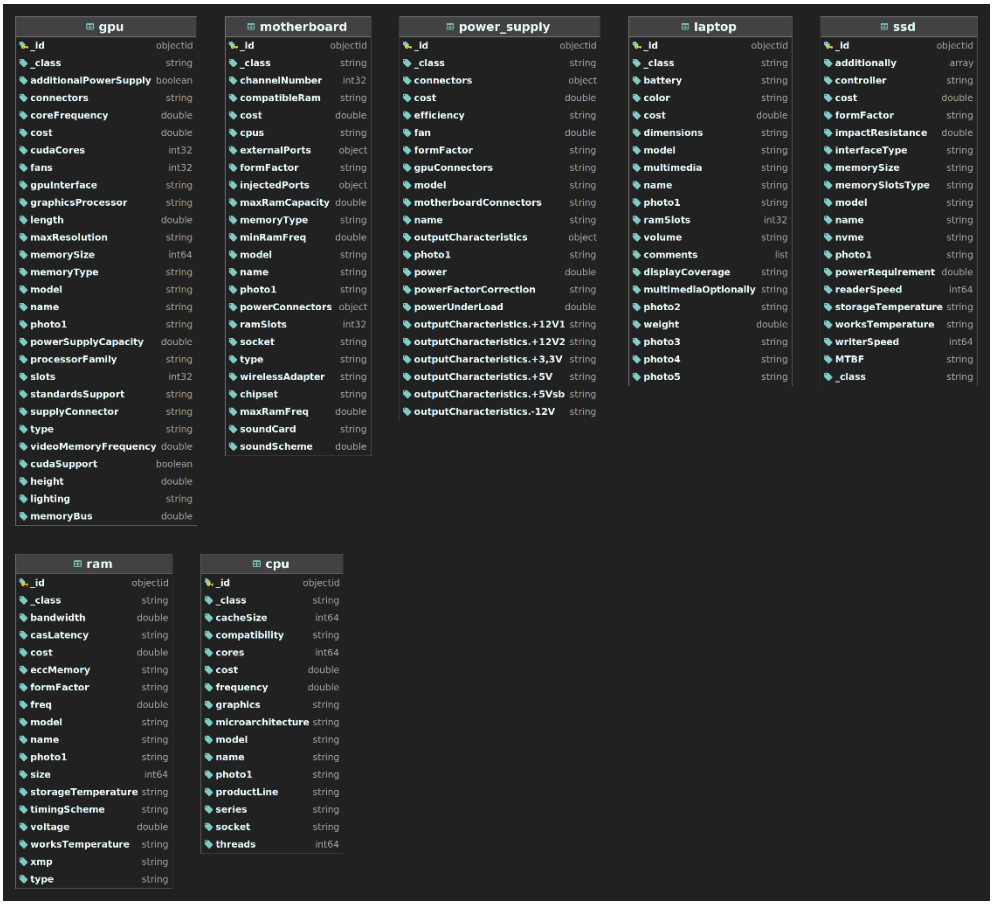



Рисунок 3.5 – Діаграма бази даних товарів

## 3.2 Програмування системи

### 3.2.1 Налаштування проекту

Для початку розробки застосунку, або після переносу його до інших серверів, потрібно його налаштувати. Налаштувати адреси підключення до баз даних, паролі з логіном, порт сервера, потрібно налаштувати інструмент контейнерізації – Docker. Все це робиться через відповідні файли.

На рисунку 3.6 – 3.8 можна побачити їх розташування в проекті, та вигляд всередині:

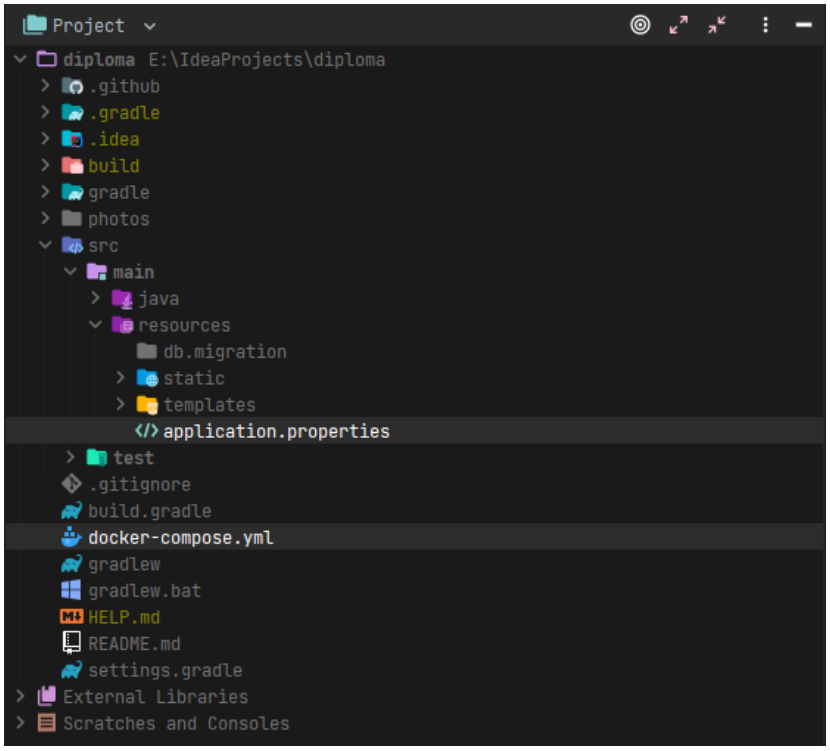
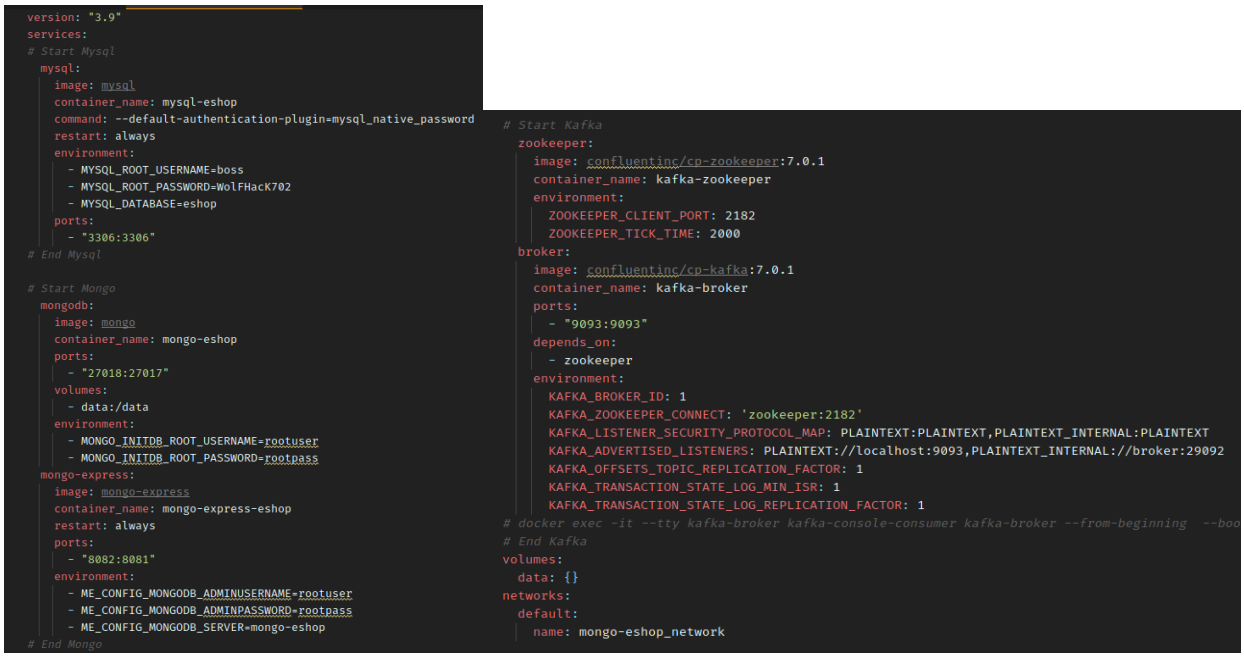



Рисунок 3.6 – файли налаштування застосунку та інструменту docker



```
version: "3.9"
services:
# Start Mysql
  mysql:
    image: mysql
    container_name: mysql-eshop
    command: --default-authentication-plugin=mysql_native_password
    restart: always
    environment:
      - MYSQL_ROOT_USERNAME=boss
      - MYSQL_ROOT_PASSWORD=WolFHacK702
      - MYSQL_DATABASE=eshop
    ports:
      - "3306:3306"
# End Mysql

# Start Mongo
  mongodb:
    image: mongo
    container_name: mongo-eshop
    ports:
      - "27018:27017"
    volumes:
      - data:/data
    environment:
      - MONGO_INITDB_ROOT_USERNAME=rootuser
      - MONGO_INITDB_ROOT_PASSWORD=rootpass
  mongo-express:
    image: mongo-express
    container_name: mongo-express-eshop
    restart: always
    ports:
      - "8082:8081"
    environment:
      - ME_CONFIG_MONGODB_ADMINUSERNAME=rootuser
      - ME_CONFIG_MONGODB_ADMINPASSWORD=rootpass
      - ME_CONFIG_MONGODB_SERVER=mongo-eshop
# End Mongo
# Start Kafka
  zookeeper:
    image: confluentinc/cp-zookeeper:7.0.1
    container_name: kafka-zookeeper
    environment:
      ZOOKEEPER_CLIENT_PORT: 2182
      ZOOKEEPER_TICK_TIME: 2000
  broker:
    image: confluentinc/cp-kafka:7.0.1
    container_name: kafka-broker
    ports:
      - "9093:9093"
    depends_on:
      - zookeeper
    environment:
      KAFKA_BROKER_ID: 1
      KAFKA_ZOOKEEPER_CONNECT: 'zookeeper:2182'
      KAFKA_LISTENER_SECURITY_PROTOCOL_MAP: PLAINTEXT:PLAINTEXT,PLAINTEXT_INTERNAL:PLAINTEXT
      KAFKA_ADVERTISED_LISTENERS: PLAINTEXT://localhost:9093,PLAINTEXT_INTERNAL://broker:29092
      KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: 1
      KAFKA_TRANSACTION_STATE_LOG_MIN_ISR: 1
      KAFKA_TRANSACTION_STATE_LOG_REPLICATION_FACTOR: 1
# docker exec -it --tty kafka-broker kafka-console-consumer kafka-broker --from-beginning  --boo
# End Kafka
volumes:
  data: {}
networks:
  default:
    name: mongo-eshop_network
```

Рисунок 3.7 – Приклад налаштування docker

```
server.port= 8088

spring.kafka.bootstrap-servers=localhost:9093

spring.jpa.hibernate.ddl-auto=update
spring.datasource.url=jdbc:mysql://localhost:3306/eshop
spring.datasource.username=root
spring.datasource.password=WolFHacK702
spring.main.allow-bean-definition-overriding=true

spring.data.mongodb.authentication-database=admin
spring.data.mongodb.username=rootuser
spring.data.mongodb.password=rootpass
spring.data.mongodb.database=products
spring.data.mongodb.port=27018
spring.data.mongodb.host=localhost

spring.servlet.multipart.max-file-size=10MB
spring.servlet.multipart.max-request-size=10MB
storage.location=/photos

spring.flyway.enabled=false
spring.flyway.baseline-on-migrate=true

spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=true

logging.level.org.springframework.data.mongodb.core.MongoTemplate=DEBUG
```

Рисунок 3.8 – Приклад налаштування застосунку

### 3.2.2 Основні компоненти багатофайлового проекту

Під час розробки проекту було використано багато-файлову систему проекту.

Увесь код застосунку знаходиться у каталозі «diploma», шлях java > com > wolfhack > diploma, де java каталог значущий, що в ньому знаходиться код java, наступні три каталоги це означають сайт компанії яка розробляє продукт, де com це домен, “wolfhack” – назва компанії, а останній каталог це назва розроблюваного проекту.

Це вважається «доброю модою» у професійних розробників java, це робиться для того щоб організувати класи, які належать до одної категорії або схожої функціональності.

Увесь код клієнтської частини, front-end, знаходиться у каталозі – resources, біля каталогу java. В ньому зберігається налаштування застосунку, шаблони, сторінки, код стилізації, переносу бази даних (міграції), та скрипти формату java-script (js).

На рисунку 3.9 зображено частину компонентів, із групи «java» (каталог який об’єднує back-end частину функціоналу, програмного коду) .


а) б)

Рисунок 3.9 – Компоненти проекту веб-застосунку з каталогу java

Розберемо детальніше компоненти кожного підкаталогу. Самим першим та верхнім класом є клас, який запускає роботи всього застосунку – стартова точка застосунку.

Далі йдуть в проекті такі каталоги:

- config

- controllers
- exception
- listener
- models
- repository
- service
- util
- validator

Розглянемо призначення класів та підкаталоги кожної з вище виведених каталогів:

Таблиця 3.1 – каталоги та їх значення

| Каталог | Призначення |
|---|---|
| config | Призначений для класів які налаштовують інші інструменти застосунку, наприклад – WebSecurityConfig налаштовує безпеку застосунку, а точніше до яких сторінок має доступ звичайний користувач або гість, та користувачі з іншими ролями, а також тип зберігання та хешування паролю. |
| controllers | Групує класи контролерів.<br>Підкаталог page – виводять сторінки за певними запитами, посиланнями, а також видає певні данні на сторінки. Наприклад – видає список товарів відеокарт, тощо.<br>Підкаталог rest – призначений для обробки rest запитів, за певними посиланнями видає або приймає інформацію у форматі JSON |

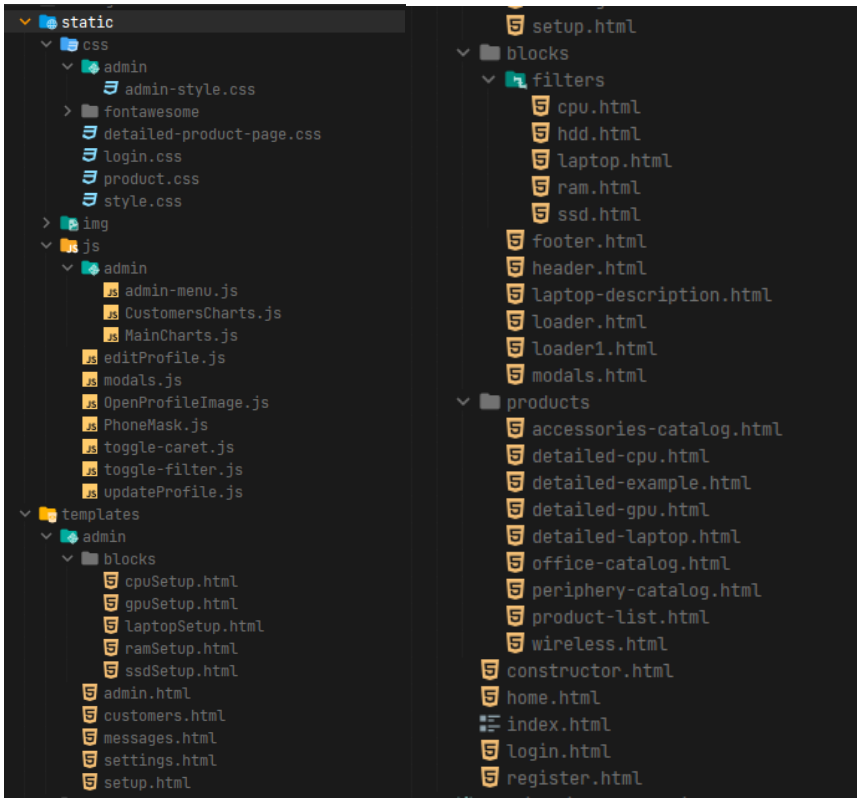
Продовження таблиці 3.1

| Каталог | Призначення |
|---|---|
| exception | Групує в собі два різни види класів, які пов’язані з помилками у ході виконання застосунку, наприклад коли користувач намагається перейти до сторінки якої не існує, або отримати сторінку певного товару, а такого товару немає.<br>В самій папці зберігаються класи цих помилок, з їх функціоналом, в подкаталозі – handler зберігаються класи які виловлюють ці помилки, де вони виникли в застосунку, їх причину, та виводить користувачу. |
| listener | Призначений для групування класів «прослуховувачів», які потрібні для дистанційного контролювання процесу застосунку. Хто зареєструвався та коли, хто увійшов, хто і що придбав та час. |
| models | Призначений для класів які описують моделі та їх дані, які вони повинні мати: користувача, окремого товари, а також для класів які переводять моделі JSON у нашу модель Java, та навпаки, останні знаходяться в подкаталозі dto mapper.<br>В подкаталозі dto знаходяться облегшені моделі, які ближчі до формату JSON, їх називають DTO – Data Transfer Object, або POJO. |
| repository | В цьому каталозі знаходяться інтерфейси з автоматичною реалізацією, призначення яких – робити запити до баз даних. |
| service | В цьому каталозі знаходиться основна логіка, де обробляються дані які надходять зовні (з клієнту), або які надходять із баз даних. |

Продовження таблиці 3.1

| Каталог | Призначення |
|---|---|
| util | Каталог в якому знаходяться класи які розширюють функціонал. Наприклад клас FileUploadUtil призначений для збереження файлів які надходять з клієнту у системі. |
| validator | Каталог в якому групуються класи та атрибути які перевіряють вхідні дані. Наприклад клас AuthenticationValidator з атрибутом AuthenticationConstraint перевіряють чи аутентифікований користувач |

На рисунку 3.10 можна побачити компоненти веб-сторінок (Веб сторінки та їх фрагменти у форматі html), а також компоненти стилізації сторінок та скрипти javascript.




а) б)

Рисунок 3.10 – Компоненти веб-сторінок (html, css, js файли)

Низка компонентів з форматом «.js» - реалізують частину динамічного фронтенду, оновлення сторінок. Під розумінням динамічного фронтенду мається всі що динамічно відкриваються елементи, елементи форм, модальний вікон, та відправлення деяких ajax запитів на сервер та отримання інформації із нього.

### 3.2.3 Основні інструменти розробки

Під час розробки веб-застосунку магазину було використано інструменти:

- Lombok – для генерації однотипного коду
- Spring Data MongoDB

- Spring Boot DevTools
- Spring Data JPA (Java Persistence API)
- Validation

Надалі розглянемо більш детально призначення кожного з них.

Lombok – допомагає програмістам писати код швидше, завдяки тому, що він дозволяє уникнути прописування однотипного коду, такого як: прописування конструкторів, створенню геттерів та сеттерів, тощо. Завдяки своєму функціоналу, мова програмування Java стає більш високорівневою.

Spring Data MongoDB та JPA – інструменти які допомагають уникненню детального прописування функціоналу для взаємодії з базою даних, таких як – відправленню запитів до бази даних. Достатньо лише прописати інтерфейс з ткою назвою методу яку дію треба щоб він робив - абстрактно описати. Але не рекомендується описувати важкі дії, а прописувати їх самому.

Spring Boot DevTools – призначений лише для розробки, економить час коли програміст оновив код, додав щось до нього або змінив, дозволяє не перезавантажувати застосунок, а лише оновити частку коду яку він змінив, також інструмент може зам у реальному часі оновлювати нову частину, або робити це коли за вказаними тригерами, за вказівками користувача.

Validation – Спрощує написання, а також логіку перевірки даних, завдяки цьому інструменту код виглядає чистіше, та робить усі перевірки перед тим, як дані потраплять до контролерів обробки, або сторінки клієнту.

### 3.2.4 Програмування front-end частини

Основна концепція програмування фронтенду з огляду на застосування фреймворку Sprnig Boot MVC побудована на використанні обраного шаблонізатору Thymeleaf. Цей шаблонізатор працює на мові Java, тому легко взаємодіє з даними у вигляді об’єктів класів, в ньому є вбудовані методи, звичайні методи Java та методи які допомагають будувати сторінки розмітки.

На прикладі лістингу 3.1 можна побачити частку того як саме працює дана методика. На вигляд thymeleaf виглядає як звичайна мова розмітки html, але якщо у html коді знаходится частина – «th:», це означає що це частина thymeleaf. На лістингу можна побачити «th:fragment», що вказує назву цієї частини коду, фрагменту, для того щоб його можна було вставити в будь-якій іншій сторінці. Також можна побачити частини де є присутній символ долару, це значить що дані в цьому блоці вставляться динамічно з сервера, за цим символом іде фігурні душки, в яких вказується назва, так званий id, щоб шаблонізатор шукав дані з цим id в даних яких він отримає від серверу.

«th:insert» - це приклад того як працює раніше вказаний «th:fragment», а саме тут шаблонізатор вставляє фрагмент коду розмітки за вказаним шляхом та назвою.

Далі в коді йде момент де саме починається заповнювання сторінки списком продуктів з сервера – це блок «th:each» задає загальні правила та стилі для усіх елементів, що будуть отримані від серверу, в нашому прикладу – товари. Після чого динамічно вказано назву яку ми дали одному елементу із списку елементів, та запити до його даних, які ми поміщаємо на сторінку. Приклад із посиланням на окремий продукт. Ми вказуємо «th:href», що означає що посилання буде братись за адресою сайту, після чого береться поточна адреса, до якої ми додаємо модель та назву продукту, прибираючи з назви символ «-», замінюючи його на пробіл: «*th:href*="${#httpServletRequest.requestURI} + '/' + ${element.model} + '/' + ${element.name.replace(' ', '-')}"».

Лістинг 3.1 – Частина коду заповнення списком продуктів на сторінці

```
<section class="block" th:fragment="product-list" id="product-list-block">
  <div class="head">
    <h2 class="heading" th:text="${title}" id="main-content"></h2>
    <div class="inputBox">
      <input type="search" name="" required="">
      <label>Введите название товара для поиска</label>
      <a href="#">
        <i class="fas fa-search"></i>
      </a>
    </div>
  </div>
  <div class="flex" style="align-items: inherit;">

    <th:block th:insert="${filterURL} :: ${filter}"></th:block>

    <div class="content">
      <div class="container">
        <div class="contentBox" th:each="element : ${products}">
          <div>
            <a th:href="${#httpServletRequest.requestURI} + '/' +
${element.model} + '/' + ${element.name.replace(' ', '-')}" class="Name"
th:text="${element.name} +'('+ ${element.model} + ')'"></a>
            <h2 class="serial" th:text="'Код:' + ${element.id}"></h2>
            <br>
            <img class="preview" th:src="'/'+${element.getPhotos()[0]}">
            <br>
            <p class="availability">Есть в наличии</p>
            <div class="cost">
              <p class="money" th:text="${element.cost} + ' грн'"></p>
              <div class="icons">
                <a href="#">
                  <i class="fas fa-balance-scale"></i>
                </a>
                <a href="#" class="buy-button">
                  <i class="fas fa-shopping-cart"></i>
                </a>
              </div>
            </div>
          </div>
        </div>
      </div>
    </div>
  </div>
</section>
```

На лістингу 3.2 добре відображено фрагмент коду де відбувається перевірка на аунтентифікованість, та притаманності користувача до ролі адміна. Як раніше було вказано частини де знаходиться фрагмент коду «th:» відноситься до шаблонізатору, але в цьому прикладі перевірка аутентифікації та ролі бачимо фрагмент коду «sec:»,

це означає що цей фрагмент це взаємодія шаблонізатору thymeleaf з інструментом Spring Boot – Spring Security.

Шаблонізатор перевіряє чи аутентифікований користувач, якщо ні, то цей фрагмент коду не буде відображений користувачу, а якщо так то він побачить цей фрагмент. Далі йде перевірка, чи у аутентифікованого користувача притаманна роль адміна, якщо так то він буде бачити цей фрагмент.

Лістинг 3.2 – Вихідний код частини з перевіркою аутентифікованості та ролі користувача

```
div th:fragment="settings" id="settings" sec:authorize="isAuthenticated()">
   <script type="text/javascript" th:src="@{/js/updateProfile.js}"></script>
   <div class="formBox-form" method="POST" enctype="multipart/form-data"
id="refreshSettings">
      <i class="fas fa-times-circle close" onclick="settings()"></i>
      <div class="inputBox admin" sec:authorize="hasRole('ADMIN')">
         <a href="/admin" class="link-btn">Админ панель</a>
      </div>
```

На лістингу 3.3 відображено приклад просто ajax запиту з методом GET, який отримує дані для створення діаграми прибутку по місяцям.

Для того щоб відправити простий запит ajax, потрібно вказати посилання за яким буде відправлятись запит, після чого тип методу, та headers, далі вказуємо що потрібно зробити при вдалої спроби отримати дані, та якщо сталась помилка. Як можна побачити при вдалої спроби, дані які отримує клієнт передаються до методу побудови діаграми, яка відображає ці дані у вигляді лінійної діаграми, або у вигляди графіка.

Лістинг 3.3 – Вихідний код ajax запиту для отримання даних про місячні прибутки

```
$.ajax({
    url: "http://localhost:8088/api/admin/earning",
    type: "GET",
    headers: {
                'X-CSRF-TOKEN': $('meta[name="csrf-token"]').attr('content')
            },
    success: function (data) {
```

```
            buildEarningChart(data);
        }
    })
    function buildEarningChart(data) {
        const Earning = document.getElementById('Earning').getContext('2d');
        const EarningChart = new Chart(Earning, {
            type: 'line',
            data: {
                labels: ['January', 'February', 'March',
                    'April', 'May', 'June',
                    'July', 'August', 'September',
                    'October', 'November', 'December'],
                datasets: [{
                    label: 'Earning',
                    data: data,
                    backgroundColor: [
                        'rgba(255, 99, 132, 1)',
                        'rgba(54, 162, 235, 1)',
                        'rgba(255, 206, 86, 1)',
                        'rgba(75, 192, 192, 1)',
                        'rgba(153, 102, 255, 1)',
                        'rgba(255, 159, 64, 1)',
                        'rgba(54, 162, 235, 1)',
                        'rgba(255, 206, 86, 1)',
                        'rgba(75, 192, 192, 1)',
                        'rgba(153, 102, 255, 1)',
                        'rgba(255, 159, 64, 1)',
                        'rgba(255, 99, 132, 1)'
                    ],
                    fill: {
                        target: true,
                        above: 'rgba(248, 126, 21, 0.5)'
                    },
                    tension: 0.1,
                    borderColor: 'rgb(248, 126, 21)',
                    borderWidth: 3,
                    pointBorderWidth: 5,
                }]
            },
            options: {
                responsive: true,
            }
        });
    }
```

### 3.2.5 Програмування backend частини

На рисунку 3.7 – відображено діаграму класів, яка відображає сторінки застосунку та класи для обробки, відправки, запитів на сервіс.

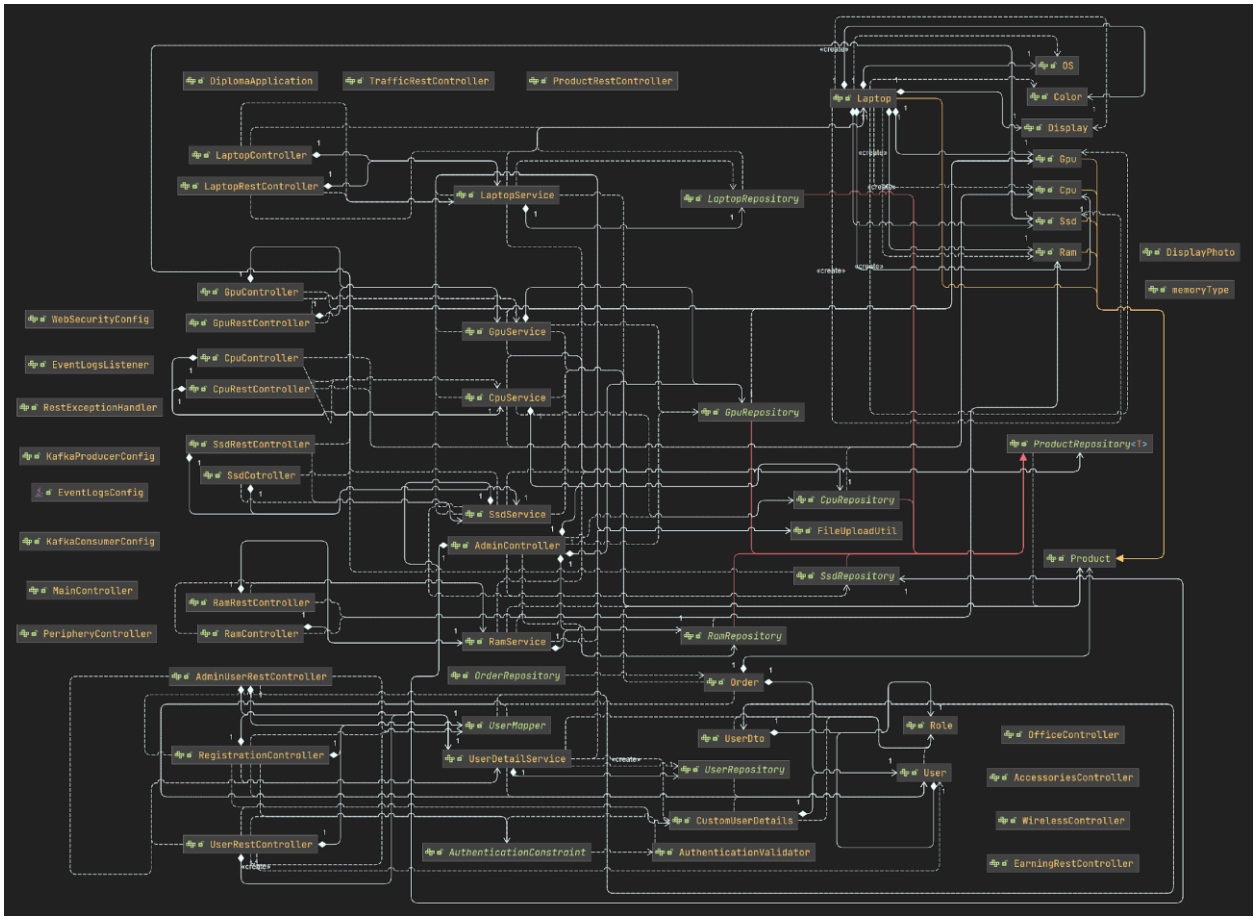



Рисунок 3.7 – Діаграма класів

Розберемо детальніше діаграму класів яку можна бачити на рисунку 3.7. У застосунку є моделі, які являють собою: товари, користувачів, ролі, у кожної моделі є свій репозиторій, за яким ми звертаємось до таблиці цієї моделі в базі даних, та сторінка, а може декілька. Наприклад сторінка усіх ноутбуків, чи одного конкретного. А також вони мають свій контролер який передає дані з клієнта на сервер та навпаки через систему REST. Кожен зі сторінки чи REST контролеру має свій сервіс, в якому відбувається основна бізнес логіка. Саме це зображено на діаграмі.

Розглянемо один з найважливіших класів-сервісів та його роботу – сервіс конструктору. Він включає в себе усі методи для знаходження відповідних підходящих комплектуючих, з подальшим переданням даних на клієнт.

Клас-сервіс має помітки: @Service та @RequiredArgsConstructor. Перший атрибут вказує що цей клас є сервісом, та його потрібно ініціалізувати у всьому застосунку, де він потрібен, лише один раз при запуску цього застосунку, другий атрибут додає усі поля до конструктора та ініціалізує один раз при запуску застосунку, як з прикладом першого атрибуту.

Серед полів класу присутні репозиторії усіх товарів, для того щоб можна було звертатись до бази даних.

На лістингу 3.4 розберемо детальніше один метод більш детально, так можна буде приблизно зрозуміти як працюють інші методи, а саме метод який знаходить усі процесори підходящі до материнської плати – «findCpuByMotherboardCompatible».

Лістинг 3.4 – Вихідний код методу знаходження процесорів до материнської плати

```
private final MotherboardRepository motherboardRepository;
private final CpuRepository cpuRepository;

public Page<Cpu> findCpuByMotherboardCompatible(Pageable pageable, String
motherboardCode) {
    if (motherboardCode.equals("")) {
        return cpuRepository.findAll(pageable);
    }
    return cpuRepository.findAllBySocket(pageable,
getMotherboard(motherboardRepository, motherboardCode).getSocket());
}

private Motherboard getMotherboard(MotherboardRepository motherboardRepository,
String motherboardCode) {
    Motherboard motherboard = motherboardRepository.findById(motherboardCode)
            .orElseGet(Motherboard::new);
    return motherboard;
}
```

Метод отримує за параметри клас відповідальний за налаштування сторінки, а точніше скільки елементів йому потрібно, та строку з кодом товару (його id), який обрав користувач. На повернення відповідає класом Page з нашим процесором в середині.

Увійшовши в метод – спочатку проходить перевірка чи не дорівнює код товару пустому рядку (нулю), після чого звертаючись до бази даних, ми передаємо за параметри: налаштування сторінки, та отримавши материнську плату за її кодом, ми передаємо властивий роз'єм процесора («Socket»), та отримуємо відповідь з бази даних, з кількістю вказаною у класі pageable.

Розглянемо класи контролер сторінок та REST контролер для моделі процесора. Почнемо з REST контролеру (Лістинг 3.5), цей клас помічений трьома атрибутами: @RestController, @RequestMapping, @RequiredArgsContructor, перший вказує що цей контролер відноситься саме то REST, також не може бути ініціалізований окрім при запуску, другий атрибут вказує URL адресу, за якою можна звернутись до цього класу, а за типом методу, до методу класу відповідно.

Із полів всередині присутній сервіс моделі – CpuService, та три методи для отримання всіх об’єктів, одного за кодом (за його id), та метод передавання об’єкту, або його змінення якщо вказано код товару.

Лістинг 3.5 – Вихідний код RestController-у

```
@RestController
@RequestMapping("/api/cpu")
@RequiredArgsConstructor
public class CpuRestController {

    private final CpuService cpuService;

    @GetMapping("/{id}")
    public Cpu getCpuById(@PathVariable("id") String id) {
        return cpuService.findById(id);
    }


    @GetMapping
    public Page<Cpu> getCpus(Pageable pageable) {
        return cpuService.findAll(pageable);
    }

    @PostMapping
    @ResponseStatus(HttpStatus.CREATED)
    public void addCpu(Cpu cpu) {
        cpuService.save(cpu);
```

```
    }
}
```

На лістингу 3.6 можна побачити MVC контролер, або контролер сторінки, в ньому також три атрибути з яких відрізняється лише перший, який помічає що це звичайний контролер, але з методів можливо зробити окремо REST методи, а також відрізняється шлях доступу до контролеру, бо заборонено мати один й самий шлях для різних контролерів. З полів в нього присутні сервіси моделі та конструктору комп’ютера, а серед методів два на отримання сторінки та один з передавання даних до серверу безпосередньо із веб-застосунку.

Лістинг 3.6 – Вихідний код MVC контролеру сторінки товару

```
@Controller
@RequiredArgsConstructor
@RequestMapping("/accessories/cpu")
public class CpuController {

    private final CpuService cpuService;
    private final PCConstructorService constructorService;

    @GetMapping
    public String getPageCPU(@RequestParam(value = "motherboard", required =
false, defaultValue = "") String motherboardCode,
            Model model, Pageable pageable){
        model.addAttribute("title", "CPU products")
                .addAttribute("filterURL", "blocks/filters/cpu")
                .addAttribute("filter", "cpu")
                .addAttribute("products",
constructorService.findCpuByMotherboardCompatible(pageable, motherboardCode));

        return "products/product-list";
    }

    @GetMapping("/{model}/{product}")
    public String getPageProducts(@PathVariable(value = "product") String
productName,
                                  @PathVariable(value = "model") String
productModel, Model model) {
        Cpu cpu = cpuService.findByNameAndModel(productName, productModel);
        model.addAttribute("title", "Процессор " + productName)
                .addAttribute("product", cpu);

        System.out.println(cpu.getProductLine());
```

```
        return "products/detailed-cpu";
    }

    @PostMapping
    public String addCPU(
            @ModelAttribute("Laptop") Cpu cpu,
            @RequestParam("photo1") MultipartFile photo1,
            @RequestParam("photo2") MultipartFile photo2,
            @RequestParam("photo3") MultipartFile photo3,
            @RequestParam("photo4") MultipartFile photo4,
            @RequestParam("photo5") MultipartFile photo5) throws IOException {
        if (cpuService.exists(cpu)) {
            return "redirect:/admin/setup?error=cpu=exists";
        }
        cpuService.save(cpu, new MultipartFile[]{photo1, photo2, photo3, photo4,
photo5});

        return "redirect:/accessories/cpu";
    }
}
```

Розберемо як створюються, додаються, товари та що для цього потрібно. Для додання товари до бази даних, треба на сервер відіслати дані моделі, з відповідними за назвою полями та багатокомпонентний файл, який обов’язково повинен бути фотографією або рисунком, відіслати можна з клієнту веб-застосунку, мобільного додатку або через JSON запит.Наприклад відправити дані за URL адресою через застосунок Postman.

Далі дані потраплять до відповідного за шляхом контролеру, та будуть передані на сервіс, в якому завдяки класу утиліти завантаження файлів (Лістинг 3.7) збережеться фотографія/рисунок у відповідній папці, а назва файлу запишеться до моделі, й буде зберігатись в базі даних, для подальшого відображення завдяки утиліти-конфігуратора який допомагає відображати файли на сайті (Лістинг 3.8), а також знайти їх у відповідній папці.

Лістинг 3.7 – Вихідний код утиліти завантаження файлів

```
public class FileUploadUtil {

    public static void trySaveFile(String uploadDirectory, String fileName,
MultipartFile multipartFile) throws IOException {
        if (multipartFile.isEmpty()) {
            return;
```

```
        }
        Path uploadPath = Paths.get(uploadDirectory);

        if (!Files.exists(uploadPath)) {
            Files.createDirectories(uploadPath);
        }

        try (InputStream inputStream = multipartFile.getInputStream()) {
                Path filePath = uploadPath.resolve(fileName);
                Files.copy(inputStream, filePath,
StandardCopyOption.REPLACE_EXISTING);
        } catch (IOException ioException) {
            throw new IOException("Could not save image file: " + fileName,
ioException);
        }
    }

    public static void trySaveFile(String uploadDirectory, String[] fileName,
MultipartFile[] multipartFile) throws IOException {

        if (fileName.length != multipartFile.length) {
            throw new IllegalArgumentException("file names array length
more/less than files array length");
        }

        for (int i = 0; i <multipartFile.length ; i++) {
            trySaveFile(uploadDirectory, fileName[i], multipartFile[i]);
        }
    }

}
```

Лістинг 3.8 – Вихідний код утиліти-конфігуратора відображення файлів

```
@Configuration
public class DisplayPhoto implements WebMvcConfigurer {

    @Override
    public void addResourceHandlers(ResourceHandlerRegistry registry) {
        exposeDirectory("photos", registry);
    }

    private void exposeDirectory(String dirName, ResourceHandlerRegistry
registry) {
        Path uploadDir = Paths.get(dirName);
        String uploadPath = uploadDir.toFile().getAbsolutePath();

        if (dirName.startsWith("../")) {
            dirName = dirName.replace("../", "");
        }

        registry.addResourceHandler("/" + dirName + "/**")
                .addResourceLocations("file:"+ uploadPath + "/");
```

```
    }
}
```

Розглянемо налаштування безпеки, доступу, сторінок, за що відповідає клас-конфігуратор – WebSecurityConfig (Лістинг 3.9). Цей клас ініціалізується як і більшість класів на початку, але працює завжди, він відповідає за фільтрацію запитів, їх доступність, розподілення да передачу ключів безпеки через заголовок – «WWW-Authenticate», хешує пароль при авторизації та реєстрації, а також авторизовані запити до сторінок, форми логіну та виходу. Більшість з функцій працює на задньому фоні за заданими «дефолтними» функціями, роботи яких непотрібно змінювати, тож які не рекомендується перероблювати.

Серед змінених присутні два методи, та один метод налаштування хешування пароля. Перший методи – configure, який налаштовує авторизацію користувача, та другий метод має таку ж саму назву, але інші параметри та направлення. Він призначений для налаштування доступу запитів, доступу до сторінок, форми входи та виходу.

Лістинг 3.9 – Вихідний код класу безпеки та доступу

```
@Configuration
@EnableWebSecurity
@RequiredArgsConstructor
public class WebSecurityConfig extends WebSecurityConfigurerAdapter {

    private final UserDetailsService userDetailsService;

    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws Exception
{
        auth.userDetailsService(userDetailsService)
                .passwordEncoder(encoder());
    }

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http.authorizeRequests()
                .antMatchers("/api/admin/**").hasRole("ADMIN")
                .antMatchers("/admin", "/admin/**").hasRole("ADMIN")
                .antMatchers("/api/**").hasRole("ADMIN")
                .antMatchers("/**").permitAll();
        http.formLogin()
```

```
                .loginPage("/login")
                .defaultSuccessUrl("/")
                .failureUrl("/login?error")
                .permitAll();
        http.logout()
                .logoutUrl("/logout")
                .invalidateHttpSession(true)
                .deleteCookies("JSESSIONID")
                .logoutSuccessUrl("/?logout");
    }

    @Bean
    public PasswordEncoder encoder() {
        return new BCryptPasswordEncoder();
    }
}
```

## 3.3 Опис функцій системи

### 3.3.1 Реєстрація та авторизація

Для реєстрації на сайті користувач може створити новий обліковий запис скориставшись кнопкою реєстрації, яка знаходиться у верхньому меню, яка в свою чергу відкриє модальне вікно реєстрації. Таким самим чином користувач може авторизуватись, але вже натиснувши кнопку для авторизації поряд з кнопкою реєстрації.

Для авторизації та реєстрації потрібно буде лише увести необхідний, але після реєстрації користувач потрібен підтвердити свою електрону пошту шляхом того, що користувач переходить за посиланням яке прийде з листом на пошту, яка була вказана.

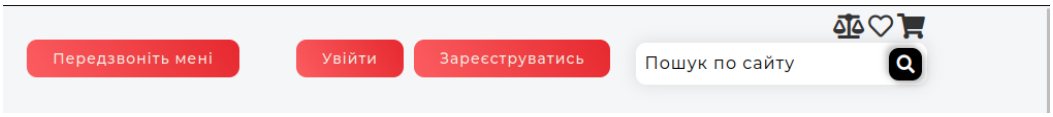



Рисунок 3.8 – Кнопки переходу на сторінки входу та реєстрації

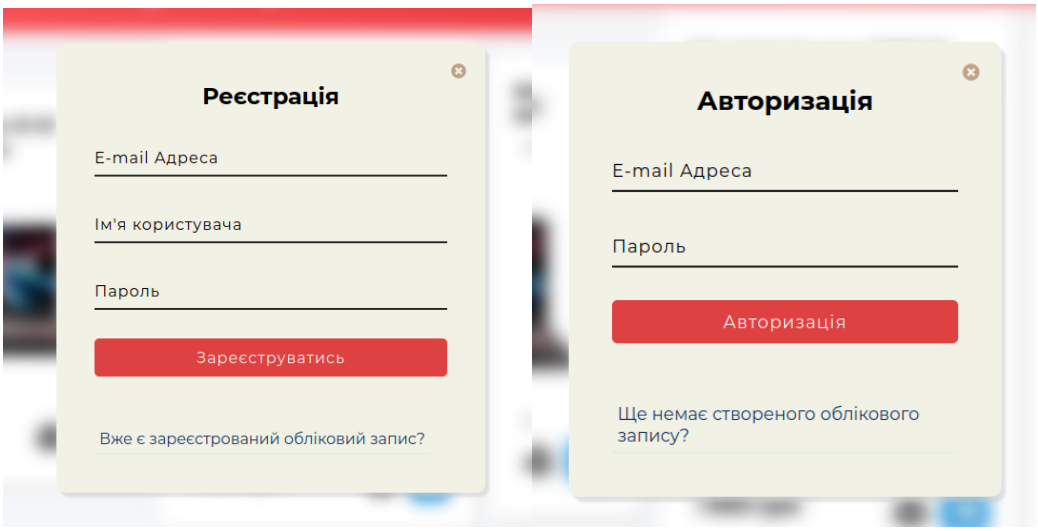



Рисунок 3.9 – Модальні вікна реєстрації та авторизації

Авторизація проходить вбудованим у «Spring Boot Security» у взаємодії з «thymeleaf» механізмом авторизації. Реєстрація проходить таким чином – користувач заповнює дані та натискає кнопку «зареєструватись», після чого дані відправляються на сервер за відповідним URL шляхом, за яким ці дані попадають до контролеру та після перевіряються, та обробляються. Коли дані було оброблено, далі генерується код, за яким буде подальше підтвердження даних від користувача, а поки ці дані зберігаються до бази даних, з поміткою – «false» у полі «active», що означає, що користувач ще не підтвердив ці дані, після чого генерується лист з посиланням, які відправляються на пошту користувачу.

Після реєстрації користувач отримує повідомлення на сайті, що йому потрібно активувати свою облікову сторінку за посиланням яке відправленно йому на пошту. Якщо користувач спробує увійти до облікової сторінки, він не зможе це зробити, йому видасть повідомлення з помилкою та причиною.

### 3.3.2 Самостійна збірка комп'ютеру

Функціонал збірки, комплектування комп’ютеру, є одним із основних, тому він виділений до окремої, своєї, сторінки.

Функційно сторінку можна поділити на дві частини:

- Панелі для кожного типу комплектуючого, в якій відображається тип комплектуючого, фотографія обраного товару та його назва.

- Список товарів для обраної категорії комплектуючого, із фільтрами та системою пошуку.

Коли користувач натискає кнопку з плюсом на панелі відповідної категорії товару, під цією панеллю відкривається список з усіма товарами, вибраної категорії. Після чого користувач повинен обрати необхідний йому товар – натиснувши на його кнопку кошику. Коли він це зробить, список автоматично закриється, а фото та назва почне відображатись на панелі.

Коли вже було обрано один товар, то при розкриті списку іншої категорії товарів, будуть відображатись усі товари які сумісні під обрані товари. Після того як усі необхідні для користувача товари було обрано, йому залишається натиснути кнопку купівлі.

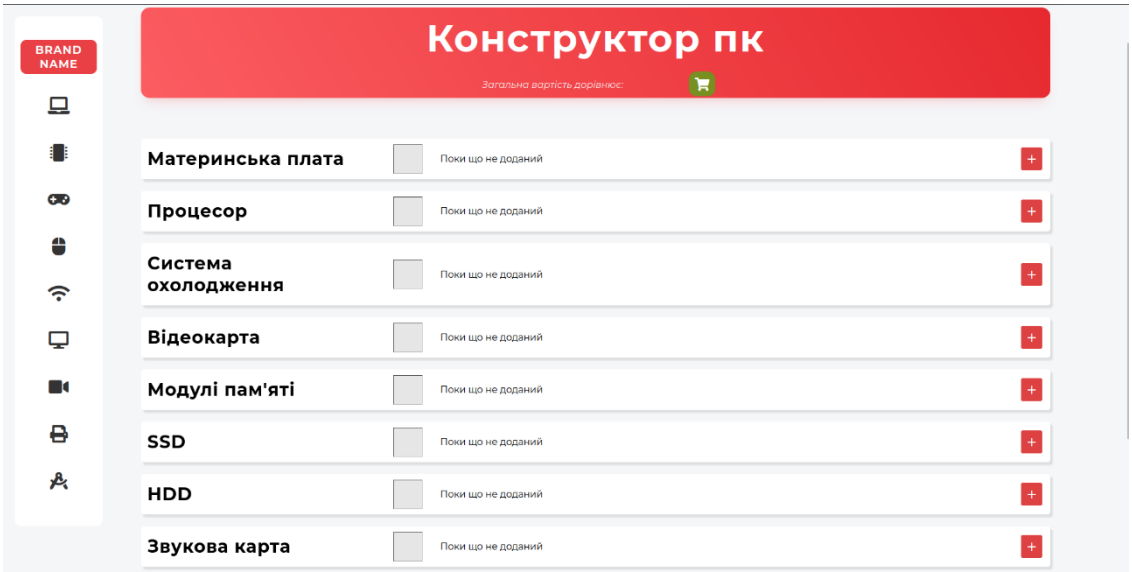



Рисунок 3.11 – Сторінка конструктору

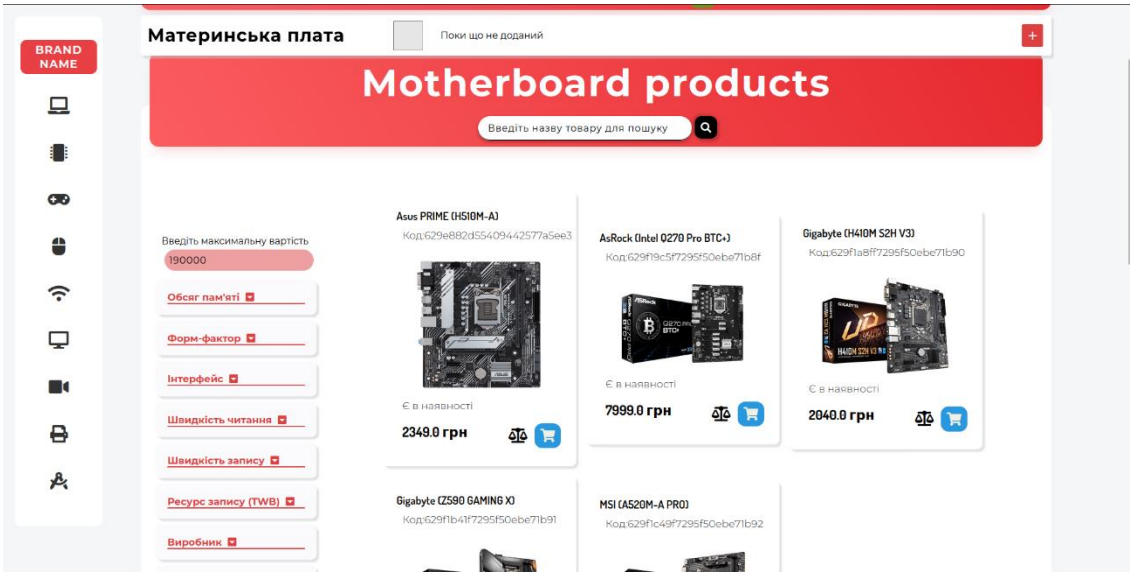



Рисунок 3.12 – Список товарів обраної категорії

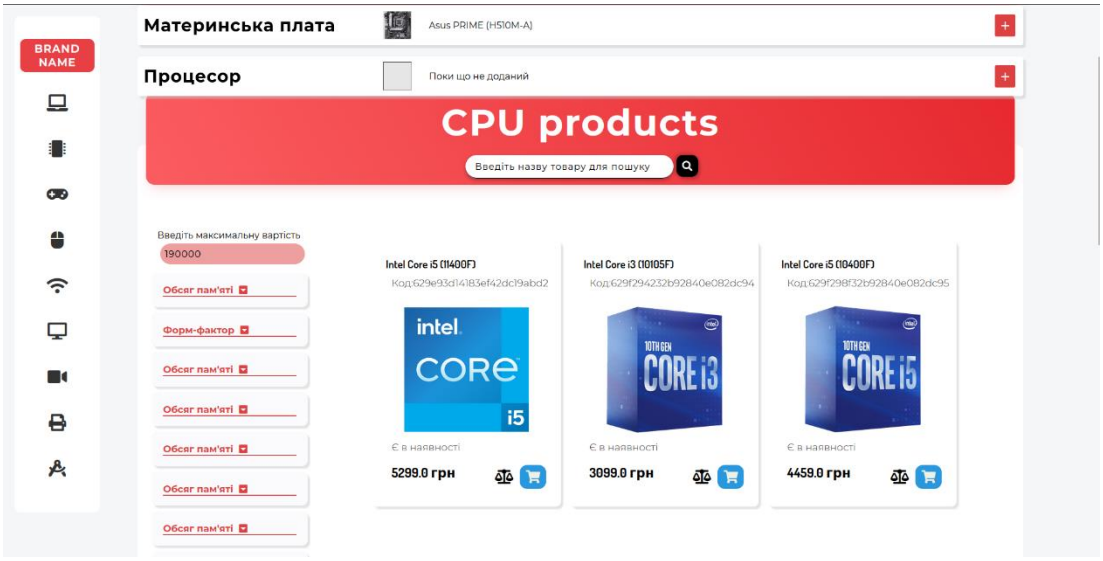



Рисунок 3.13 – Список із сумісними товарами обраної категорії

При обирані товарі, зверху пишеться загальна сума за купівлю компонентів, та з’явиться кнопка для купівлі усіх обраних товарів. Приклад можна побачити на рисунку 3.14.

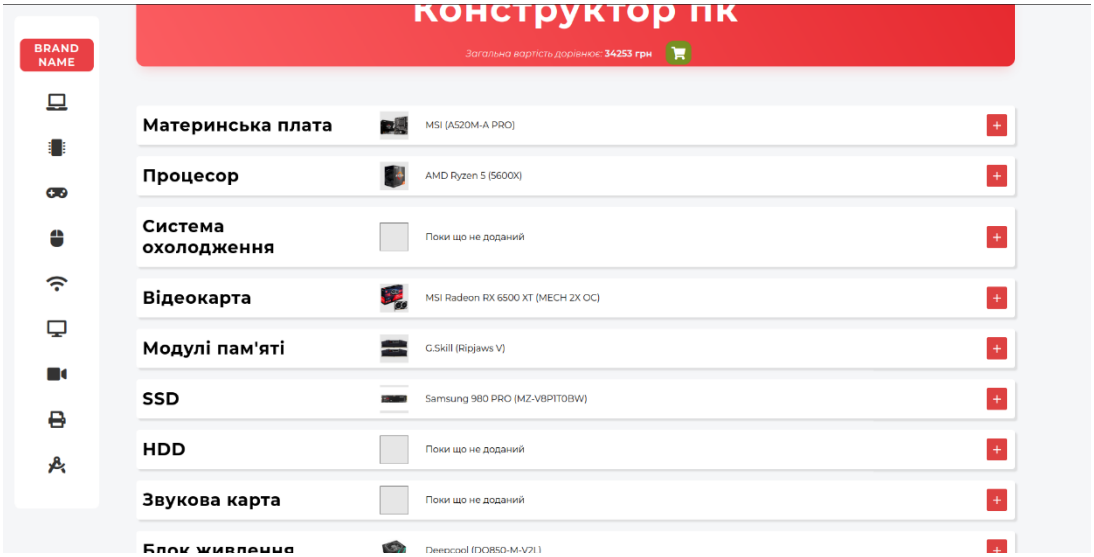



Рисунок 3.14 – Приклад заповненого конструктору

На стороні сервера усі комплектуючі перевіряються на сумісність, роз’єм процесора материнської плати з процесором, графічний інтерфейс відеокарти та чи підтримує цей інтерфейс материнська плата, а також розміри відео карти до материнської плати. Все це підсумовується за кількістю споживання ватт, за цим покажчиком обирається блоки живлення.

### 3.3.3 Управління замовлення підприємцем

Відстежувати статистику прибутку, або замовлення які надійшли чи виконуються, можна на сторінках для адмінів. Для того щоб туди потрапити треба авторизуватись, та мати права адміністратора, або аналітика. Після чого перейти у настройки облікової сторінки, де можна буде побачити кнопку адміністрації, яка буде відображатись тільки якщо є необхідні ролі у облікової сторінки. Коли натискається кнопка користувач переходить до головної адміністративної сторінки (Рисунок 3.15).


Рисунок 3.15 – Головна адмін-сторінка

На головній сторінці відображаються головні статистики з прибутку, приводження користувачів з реклами, якщо така є, та інші. Нижче відображенні поточні замовлення на яких відображено які товари замовляють, скільки коштує та адреса з ім’ям замовника.

На наступній сторінці, яку можна бачити зліва у меню, користувач може побачити та змінити замовлення що вже виконуються. Тут показуються ім’я та город замовника, назва товару, його вартість, статус та день коли було створено замовлення.

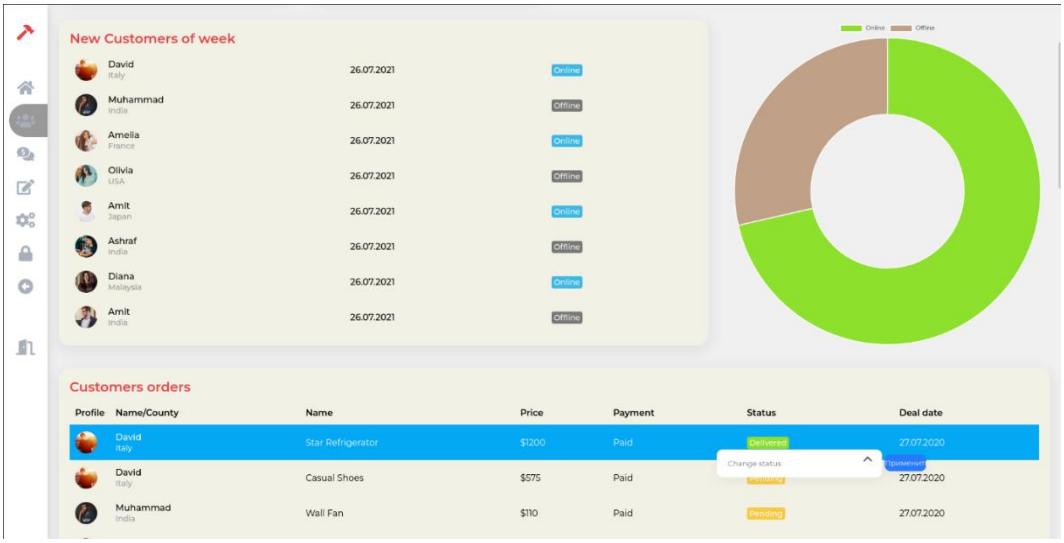



Рисунок 3.16 – Панель контролювання замовлень

На наступній сторінці (Рисунок 3.17) можна контролювати запити на зворотній зв’язок, відповідати, чи видаляти їх.

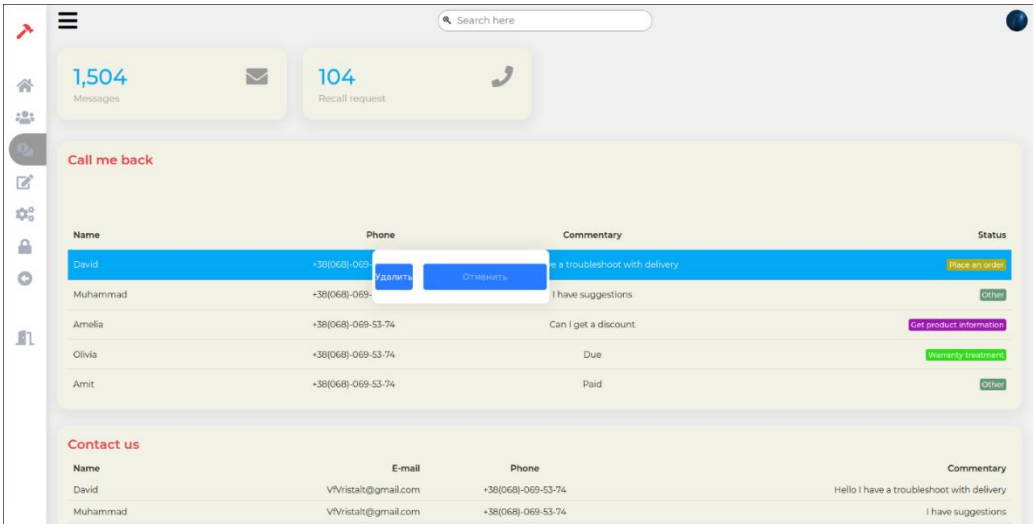



Рисунок 3.17 – Панель керуванням зворотнім зв’язком

### 3.3.4 Запланований функціонал

Заздалегідь був запланований, але не реалізований у першій версії функціонал такий як:

- Контроль замовлень та зворотнім зв’язком
- Додання інших товарів

- Статистика з відвідувань, коментарів, та загального прибутку
- Діаграми прибутку клієнтів із реклами: Youtube, Amazon, Facebook
- Діаграма статистика клієнтів онлайн
- Сторінка купівлі товарів.

## 3.4 Висновки за розділом

Було запроектовано системи з автоматичним підбором сумісних комплектуючих до вже обраних, системи облікових сторінок та систему електронного магазину, що має декількох акторів та низку прецедентів, що були запрограмовані з урахуванням специфіки створення e-commerce веб-застосунків.

Розроблений програмний продукт має дружній для користувача інтерфейс, яким просто користуватись, та який забезпечує e-commerce взаємодію: магазин-замовник.

# ВИСНОВОК

Здійснено порівняльна характеристика існуючих аналогів, та встановлено їх переваги для реалізації.

Висунуті базові вимоги до функціоналу розробляємого продукту з метою спрощення порядку документального оформлення та розуміння плану розробки застосунку.

Встановлено що архітектура продукту буде стандартною як для e-commerce застосунків, а саме тришаровою: клієнт, сервер, бази даних.

Враховуючи загальні тенденції розробки програмних продуктів було обрано створення веб-застосунку на базі мови Java з середовищем розробки було взято найпопулярнішу, у цій сфері, середовище програмування – IntelliJ IDEA Ultimate.

Для розробки REST сервісу було обрано фреймворк Spring Boot, який є зрілим, добре відомим та стабільним. Він володіє широкими можливостями, шаблонами проектування і відрізняється високим ступенем безпеки. Він також має відмінну документацію та підтримку спільноти, яка допомагає вирішувати більшість проблем.

За СУБД було обрано – MySQL, як базу даних для збереження даних пов’язаних з замовленнями, системою облікових записів, та інших невеликих даних.

За ДСУБД було обрано – MongoDB, як базу даних для збереження великої кількості даних з якою потрібно буде працювати постійно, майже при кожному запиті в застосунку – товарами, комплектуючими. Це дозволить зробити швидкий та легкий пошук по товарам, що також полегшить роботу з конструктором пк.

Розробка проекту велась на операційній системі Linux – Ubuntu.

Було запроектовано системи з автоматичним підбором сумісних комплектуючих до вже обраних, системи облікових сторінок та систему електронного магазину, що має декількох акторів та низку прецедентів, що були запрограмовані з урахуванням специфіки створення e-commerce веб-застосунків.

# ПЕРЕЛІК ПОСИЛАНЬ

1. Stack Overflow Developer Survey 2021. [Електронний ресурс]. – Режим доступу: www. URL: https://insights.stackoverflow.com/survey/2021#most-loved-dreaded-and-wanted-new-collab-tools-love-dread (Дата звернення: 10.02.2022)

2. REST: простым языком. REST | by Andriy Iveshchenko | Medium. [Електронний ресурс]. – Режим доступу: www. URL: https://medium.com/@andr.ivas12/rest-простым-языком-90a0bca0bc78#:~:text=REST.%20(REpresentational%20State%20Transfer)%20—,передачи%20состояний%20ресурсов%20по%20HTTP (Дата звернення: 10.02.2022)

3. Определения понятия Stateful и Stateless в контексте веб-сервисов (перевод) | by Dmitry Ermakovich | Medium. [Електронний ресурс]. – Режим доступу: www. URL: https://medium.com/@ermakovichdmitriy/определения-понятий-stateful-и-stateless-в-контексте-веб-сервисов-перевод-18a910a226a1 (Дата звернення: 12.02.2022)

4. Apache CXF vs Spring Boot. [Електронний ресурс]. – Режим доступу: www. URL: https://www.g2.com/compare/apache-cxf-vs-spring-boot (Дата звернення: 10.02.22)

5. REST API: JAX-RS vs Spring. [Електронний ресурс]. – Режим доступу: www. URL: https://www.baeldung.com/rest-api-jax-rs-vs-spring (Дата звернення: 11.02.2022)

6. Quarkus vs Spring Boot: Which Framework is Right for You. [Електронний ресурс]. – Режим доступу: www. URL: https://rollbar.com/blog/quarkus-vs-spring-boot/ (Дата звернення: 11.02.22)

7. Ubuntu. [Електронний ресурс]. – Режим доступу: www. URL: https://uk.wikipedia.org/wiki/Ubuntu (Дата звернення: 31.05.22)

8. Docker. [Електронний ресурс]. – Режим доступу: www. URL: https://ru.wikipedia.org/wiki/Docker (Дата звернення: 31.05.22)

9. Home – Docker. [Електронний ресурс]. – Режим доступу: www. URL: https://www.docker.com/ (Дата звернення: 31.05.22)

# ДОДАТОК А
# ВИХІДНИЙ КОД ПРОГРАМИ