Plag

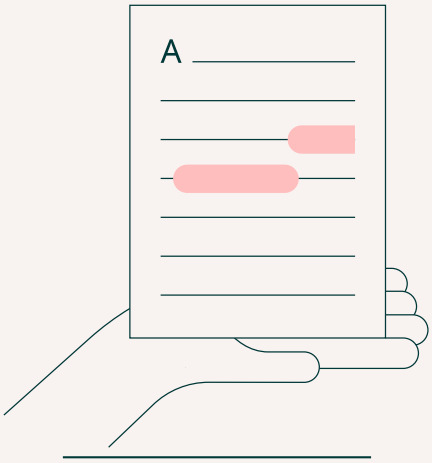
Звіт про оригінальність

● Оцінка схожості 2%

● Ризик плагіату НИЗЬКИЙ

Olga Kagalo 2025-06-20 21:59

Посилання на звіт: 10oek / Посилання користувача: qEAc



Ось вона –
Ваша звіт про оригінальність!

Ми раді повідомити, що перевірка вашого документа завершена, і результати вже готові! Наші алгоритми старанно працювали, щоб знайти збіги в наших базах даних.

На наступних сторінках ви знайдете результати перевірки:

Бали
Збіги
Посилання

Ваш документ було перевірено за такими джерелами:

- ✓ База даних інтернет-джерел
- ✓ База даних наукових статей
- ○ Глибока перевірка (наш вдосконалений алгоритм)

Бали

Збіги тексту	2%
Перефразування	0%
Цитований текст	0%
Неправильне цитування	0%
Збігів не знайдено	98%

Ризик плагіату НИЗЬКИЙ

Ризик плагіату вказує, як збіги тексту розподілені по документу. Вищий ризик виникає, коли збіги з'являються близько один до одного, наприклад, у тому самому абзаці або розділі.

Оцінка схожості 2%

Оцінка схожості показує, скільки слів або символів у вашому документі збігаються з текстами інших документів, включаючи перефразовані тексти або неправильні цитати.

Збіги

ВСТУП

Інтернет у сучасному світі виступає не лише глобальним середовищем для обміну інформацією, а й ефективною платформою для реалізації різноманітних видів діяльності — від комунікації до ведення бізнесу, навчання, розваг і державного управління. Центральною ланкою цього середовища виступає вебсайт — віртуальний ресурс, який об'єднує інформаційні, функціональні та візуальні компоненти для досягнення певної мети. Розмаїття сайтів вимагає їхнього системного аналізу, що передбачає чітке визначення, загальну характеристику та класифікацію.

У XXI столітті розвиток вебтехнологій значно змінив як зовнішній вигляд сайтів, так і їхню функціональність. Технології штучного інтелекту, автоматизація обслуговування, голосові інтерфейси, персоналізований контент — усе це змінює підхід до створення сайтів. Особливої популярності набуває інтеграція з мобільними додатками, створення вебдодатків (PWA), які працюють навіть без постійного доступу до Інтернету.

Також набирає обертів тенденція до використання CMS (Content Management System) — систем управління вмістом, що дозволяють створювати та редагувати сайти без глибоких знань програмування (наприклад, WordPress, Joomla, Drupal).

Проєктування веб-сайту інтернет-магазину «М-фурнітура» з використанням фреймворку Spring та системи управління базами даних PostgreSQL є доцільним і важливим з низки об’єктивних технічних, бізнесових і безпекових причин. Нижче наведено обґрунтування цієї важливості

1. Надійність і масштабованість фреймворку Spring. Фреймворк Spring є одним із найпопулярніших серед Java-розробників завдяки своїй модульності, розширюваності та підтримці сучасних принципів архітектури, таких як MVC (Model-View-Controller), Dependency Injection, RESTful API, Spring Security тощо.

Переваги для інтернет-магазину «М-фурнітура»:

Масштабованість: Spring дозволяє зручно нарощувати функціонал (додавати модулі оплати, кабінет користувача, аналітику тощо).

Підтримка REST API: Зручно реалізувати інтеграцію з мобільним додатком, CRM, платіжними системами.

Інверсія керування (IoC): Забезпечує гнучку і модульну побудову логіки магазину — від оформлення замовлень до зберігання історії покупок.

Spring Boot: пришвидшує розгортання, полегшує конфігурацію, підвищує продуктивність.

2. Потужність і надійність PostgreSQL як СУБД

4 PostgreSQL — це потужна об’єктно-реляційна система керування базами даних з відкритим кодом, яка ідеально підходить для комерційних проєктів з великою кількістю транзакцій і користувачів.

Переваги PostgreSQL для «М-фурнітура»:

Надійність і відповідність ACID: транзакції в інтернет-магазині (реєстрація, купівля, повернення товарів) будуть виконуватись коректно та без втрати даних.

Підтримка складних запитів: дозволяє гнучко працювати з товарами, категоріями, замовленнями, клієнтами.

Розширення (PostGIS, PL/pgSQL): можливість реалізації додаткових сервісів, таких як пошук по геолокації.

Висока продуктивність і масштабованість: PostgreSQL витримує великі навантаження, що важливо при збільшенні кількості товарів і клієнтів.

3. Безпека та надійність

Інтернет-магазин обробляє персональні дані клієнтів, замовлення, платіжну інформацію — все це вимагає високого рівня безпеки. Spring Security надає комплексні механізми:

авторизація та автентифікація користувачів;

захист від CSRF, XSS, SQL-ін’єкцій;

шифрування паролів та контроль сесій.

Також PostgreSQL має вбудовані механізми контролю доступу, шифрування та журналювання дій, що підвищує загальну безпечність системи.

4. Гнучкість та майбутнє розширення функціоналу

Поєднання Spring і PostgreSQL забезпечує гнучку архітектуру, яка дозволяє поступово розширювати сайт — наприклад:

додати систему знижок і акцій;

реалізувати аналітику продажів;

інтегрувати чат-бот або систему підтримки клієнтів;

створити панель адміністратора для управління товарами.

5. Підтримка та розвиток у спільноті

Обидві технології (Spring і PostgreSQL) активно підтримуються спільнотою розробників і мають багату документацію. Це спрощує обслуговування, модернізацію проєкту, пошук готових рішень, модулів і бібліотек.

Вибір Spring + PostgreSQL як технічної бази для створення інтернет-магазину «М-фурнітура» є стратегічно виправданим. Ці технології забезпечують:

стабільну роботу магазину;

високий рівень захисту даних;

можливість масштабування та розвитку;

сучасний підхід до розробки;

відповідність міжнародним стандартам безпеки та якості.

Це дозволяє не лише реалізувати базовий функціонал, а й забезпечити фундамент для успішного зростання онлайн-бізнесу.

1 АНАЛІЗ ЗАВДАННЯ ТА ОГЛЯД СУЧАСНИХ ЗАСОБІВ РОЗРОБКИ WEB-САЙТІВ

1.1 Аналіз об’єкту проєктування

В даному дипломному проєкті потрібно спроєктувати веб-сайт інтернет-магазину «М-фурнітура» з використанням фрейворку Spring та PostgreSQL. Основною метою є створення інтерактивного онлайн-ресурсу, який би дозволив користувачам переглянути продукцію інтернет-магазину та додати її у кошик, 14 заповнити форму зворотного зв’язку.

14 Основна вимога – по можливості швидке і комфортне користування web-сайтом, яке забезпечує зручний та зрозумілий інтерфейс. Веб-сайт для інтернет-магазину «М-

фурнітура» повинен забезпечити розв'язання наступних задач:

Представлення інформації про компанію, її історію, місцезнаходження, контактну інформацію та робочі години;

Показ продукції, яку випускає фірма, з можливістю додати товар до кошика, ціну та зображення;

Надання можливості зворотного зв'язку з клієнтами через форму зворотного зв'язку або електронну пошту;

Доступний та зрозумілий опис наявного доставлення.

В процесі розробки веб-сайту необхідно виконати наступні вимоги: інтерактивність, можливість заповнення форми зворотного зв’язку, зручність використання та дизайн.

1.2 Визначення етапів проєктування

На основі вказаних в пункті 1.1 вимог розробка веб-сайту інтернет-магазину «М-фурнітура»буде відбуватися в такі етапи:

визначення цільової аудиторії;

вибір концепції веб-сайту;

вибір програмних засобів веб-розробки;

розробка інтерфейсу веб-сайту;

організація серверної та клієнтської частини;

тестування проєкту.

1.3 Загальна характеристика та класифікація сайтів

Вебсайт (або просто сайт) — це сукупність вебсторінок, пов’язаних між собою гіперпосиланнями, що розміщуються на одному доменному імені та мають єдину структуру. Усі сайти функціонують завдяки взаємодії серверів, клієнтів та протоколів обміну даними (HTTP, HTTPS), використовуючи мову розмітки HTML та допоміжні технології (CSS, JavaScript, PHP тощо).

Основні функції сайтів:

Інформаційна: надання текстового, графічного, відео- та аудіоконтенту користувачам;

Комунікативна: забезпечення зворотного зв’язку між користувачами, адміністраторами

або організаціями;

Рекламна: просування товарів, послуг, брендів;

Торгівельна: забезпечення електронної комерції;

Освітня: доступ до навчальних матеріалів, онлайн-курсів, інтерактивного навчання;

Розважальна: надання користувачам ігрового, музичного, відео- та іншого розважального контенту.

Загальна структура будь-якого сайту передбачає головну сторінку (home page), внутрішні сторінки, навігаційне меню, інтерфейс користувача, системи адміністрування та підтримки баз даних.

Класифікація сайтів за призначенням

В залежності від мети створення та використання сайти поділяються на кілька основних типів.

Інформаційні сайти. Їх основна мета — інформувати користувача про певні події, явища чи поняття. Прикладами є новинні портали, енциклопедії, тематичні блоги. Такі сайти часто оновлюються, мають архів новин, розділи та підрозділи для зручності пошуку.

Корпоративні сайти. Призначені для представлення компаній або установ в Інтернеті. Містять інформацію про історію, послуги, команду, контактні дані. Часто включають окремі модулі: новини, прайс-листи, фотогалереї, форму для зворотного зв’язку.

Інтернет-магазини (e-commerce). Забезпечують можливість купівлі-продажу товарів чи послуг онлайн. Характеризуються наявністю кошика, платіжної системи, особистого кабінету користувача, системою фільтрів, пошуку товарів.

Освітні сайти. Пропонують доступ до навчальних курсів, тестів, електронних підручників, відеолекцій. До цієї категорії належать платформи дистанційного навчання (наприклад, Coursera, Prometheus, Moodle).

Промо-сайти. Розробляються для просування конкретного товару, події чи кампанії. Відрізняються креативним дизайном, інтерактивністю, короткотерміновим життєвим циклом.

Соціальні мережі та форуми. Спрямовані на створення та підтримку комунікації між користувачами. Найвідоміші приклади — Facebook, Instagram, Reddit, форуми за інтересами.

Персональні сайти та блоги. Відображають погляди, діяльність або хобі окремої особи. Часто містять текстові матеріали, фотоальбоми, відеоблоги, записи про подорожі, хобі тощо.

Класифікація сайтів за функціональністю

Окрім призначення, сайти можна класифікувати за функціональним наповненням:

Статичні сайти. Містять фіксований контент, який рідко оновлюється. Здебільшого складаються з HTML-сторінок без динамічних елементів. Переваги: швидке завантаження, простота розробки. Недоліки: відсутність гнучкості.

Динамічні сайти. Контент генерується у режимі реального часу залежно від запитів користувача. Використовують мови програмування та бази даних. Приклади — інтернет-магазини, освітні платформи.

Адаптивні сайти. Автоматично змінюють інтерфейс відповідно до типу пристрою (смартфон, планшет, ПК). Забезпечують зручність перегляду на будь-якому екрані.

Односторінкові сайти (landing page). Складаються з однієї вебсторінки, на якій розміщено повну інформацію про продукт або послугу. Найчастіше використовуються для рекламних кампаній або презентацій.

Класифікація за рівнем доступу та взаємодії

Також сайти можна класифікувати за рівнем доступності контенту та інтерактивності:

Публічні сайти. Доступні для необмеженого кола користувачів. Приклад — більшість новинних, освітніх, інформаційних ресурсів.

Приватні сайти. Вимагають авторизації або підписки. Наприклад, корпоративні портали, внутрішні ресурси навчальних закладів.

Інтерактивні сайти. Забезпечують активну участь користувача: залишення коментарів, створення профілів, оцінювання матеріалів, участь в обговореннях.

Пасивні сайти. Не мають засобів зворотного зв’язку або участі — лише для ознайомлення з контентом.

2. ВИБІР ПРОГРАМНИХ ЗАСОБІВ РОЗРОБКИ

2.1 Вибір мов програмування та фреймворків

Java, глобально визнана мова програмування, часто обирається для побудови

надійних, масштабованих та безпечних додатків корпоративного рівня. В рамках розробки платформи електронної комерції вибір Java як основної мови є обґрунтованим, завдяки низці переваг, як технічних, так і організаційних.

По-перше, об'єктно-орієнтований характер Java сприяє створенню чіткої, зрозумілої та підтримуваної архітектурної структури. Інкапсуляція, наслідування та поліморфізм полегшують ефективну організацію коду та розподіл функціональності між різними модулями, що призводить до високонадійної системи. У контексті інтернет-магазину такий підхід дозволяє незалежно розробляти вузли для каталогізації товарів, управління користувачами, обробки замовлень та інтеграції з платіжними шлюзами. Ці 12 модулі взаємодіють через чітко визначені інтерфейси, спрощуючи технічне обслуговування, розширення та поточну експлуатацію системи.

Ще однією ключовою перевагою Java є її платформа незалежність, що реалізується через Java Virtual Machine (JVM). Це дозволяє додатку працювати на будь-якій операційній системі – Windows, Linux, macOS тощо – без жодних змін у коді. Для сайту електронної комерції, розробленого для хмарного розгортання, наприклад, AWS або Google Cloud, ця міжплатформна здатність значно спрощує процедури розгортання.

Java також може похвалитися високим рівнем стабільності, що особливо важливо для великих комерційних підприємств. Наявність версій Long Term Support (LTS), таких як Java 11 і Java 17, гарантує регулярні оновлення безпеки та підтримку спільноти. Це робить Java надійним вибором для довгострокової розробки систем.

З точки зору продуктивності, Java перевершує багато інших мов, що зумовлено JIT-компіляцією (Just-In-Time), ефективним збиранням сміття та добре оптимізованою стандартною бібліотекою. В рамках систем електронної комерції це забезпечує здатність обробляти тисячі запитів на секунду без значних втрат продуктивності. Наприклад, у періоди високого попиту, як-от розпродажі чи рекламні акції, система, написана на Java, може обробляти всі замовлення, не відчуваючи затримок. Крім того, Java підтримує багатопоточність, полегшуючи ефективне використання апаратних ресурсів. Ця здатність є важливою в системах, які одночасно керують багатьма користувачами.

Варто також відзначити величезну колекцію Java-сумісних бібліотек, фреймворків та інструментів розробки, доступних розробникам. Розробники мають у своєму розпорядженні інструменти для ведення журналів (Logback, SLF4J), тестування (JUnit, Mockito), створення веб-сервісів (Spring Boot), взаємодії з базами даних (JPA, Hibernate) та інтеграції систем (REST, Kafka), що дає змогу швидко створювати повністю функціональні рішення.

Велика кількість кваліфікованих Java-розробників у поєднанні з проактивною

глобальною спільнотою, яка забезпечує постійну підтримку та обмін знаннями, також є важливою. Це забезпечує легкий доступ до досвіду для довгострокової підтримки та еволюції проєкту.

Отже, вибір Java як основної мови програмування для платформи електронної комерції «M-Furnitura» виправданий завдяки її надійності, міжплатформним можливостям, масштабованості, потужній екосистемі та активній спільноті.

Spring Boot – це фреймворк, який значно спрощує розробку та налаштування веб-додатків на базі Java. У контексті розробки платформи електронної комерції «M-Furnitura» Spring Boot підтримує архітектуру backend, забезпечуючи швидку розробку, гнучку структуру проєкту та інтеграцію з низкою бібліотек.

Вперше запустивши Spring Boot service в IntelliJ, IDE автоматично налаштувало все, заощадивши мені день ручного налаштування. Наприклад, замість ручного налаштування контейнерів сервлетів, налаштувань баз даних або функцій безпеки, Spring Boot автоматично ініціалізує всі компоненти на основі залежностей проєкту. Це дозволяє зосередитись безпосередньо на бізнес-логіці.

Крім того, Spring Boot пропагує «думний» підхід до створення додатків, що пропонує найкращі практики та стандартні оптимальні налаштування, які за потреби можна легко переписати. Ця стратегія прискорює етап розробки, забезпечує послідовність та знижує ймовірність помилок, пов'язаних з конфігурацією.

У налаштуваннях платформи електронної комерції, де присутні різні об'єкти (товари, користувачі, замовлення, категорії), Spring Boot дозволяє надзвичайно швидку реалізацію операцій CRUD через інтеграцію з Spring Data JPA. Наприклад, створення REST-контролера для обробки запитів на перегляд товарів або оформлення замовлення вимагає лише кілька десятків рядків коду, оскільки фреймворк автоматично підключає HTTP-запити до відповідних методів коду сервісу.

Ще однією ключовою перевагою є вбудований веб-сервер (Tomcat, Jetty або Undertow), який забезпечує виконання програми як звичайного процесу Java, не вимагаючи спеціального веб-контейнера. Це особливо вигідно під час розгортання сервісу за допомогою контейнерів Docker або в хмарних середовищах.

Spring Boot також пропонує глибоку інтеграцію з іншими модулями екосистеми Spring, зокрема:

Spring Security – для реалізації аутентифікації та авторизації;

Spring Data JPA – для доступу до бази даних;

Spring Web – для створення REST API;

Spring Actuator – для моніторингу та діагностики стану додатків;

Spring Kafka (якщо потрібна обробка повідомлень);

Spring Validation – для перевірки вхідних запитів.

Завдяки модульному дизайну фреймворку архітектура мікросервісів може бути легко реалізована в нашому проєкті: кожен сервіс (наприклад, сервіс користувачів, сервіс продуктів, сервіс замовлень) працює як окремий додаток Spring Boot, який можна незалежно масштабувати, оновлювати або розгортати.

Spring Boot також пропонує гнучку систему конфігурації (через application.properties або application.yml), яка полегшує безперебійне налаштування параметрів доступу до бази даних, портів сервера, властивостей безпеки тощо. Ці налаштування можна винести в окремі файли, що корисно під час розгортання в різних середовищах (розробка, тестування, виробництво).

Підсумовуючи, Spring Boot в проєкті платформи електронної комерції є життєво важливою технологією, яка забезпечує:

Прискорену розробку;

Дотримання стандартів;

Просте тестування;

Створення гнучкої, модульної архітектури;

Просту інтеграцію інших компонентів системи.

Її використання не тільки виправдано з технічної точки зору, але й економічно вигідно, зменшуючи витрати на обслуговування, адаптацію та масштабування системи.

Не зважаючи на те, що «М-фурнітура»розроблена з використанням Java та Spring Boot, без JavaScript наш інтерфейс виглядав би надто статично. Отож, ми додали трохи клієнтських скриптів, аби зробити сайт більш «живим» та комфортним для користувача.

JavaScript – це стандартна мова програмування, що використовується у всіх веб браузерах. Вона надає можливість робити сторінки динамічними, реагувати на дії користувача, валідувати дані безпосередньо на боці клієнта та маніпулювати DOM-елементами без потреби перезавантажувати сторінку. Навіть у проєктах без складного фронтенду чи SPA (Single Page Application), як-от у нашому випадку, JavaScript виконує

низку корисних функцій, які значно покращують зручність користування сайтом.

Використання JavaScript у проєкті:

Покращення взаємодії з користувачем. JavaScript дає змогу реалізувати прості інтерактивні можливості, на кшталт спливаючих сповіщень, підтвердження дій (наприклад, перед видаленням товару з кошика), відкриття модальних вікон, сортування таблиць, обробка івентів тощо. Це створює відчуття “живого” сайту та покращує досвід взаємодії з ним.

Клієнтська валідація форм. До відправлення даних на сервер JavaScript може перевірити їхню коректність. Наприклад, переконатися, що всі поля заповнені та формат email-адреси відповідає вимогам. Це дозволяє уникнути зайвих запитів до серверу, зекономити ресурси та прискорити взаємодію.

Анімовані елементи інтерфейсу. Хоча повноцінні анімації не є необхідними для функціонування інтернет-магазину, навіть елементарні ефекти при наведенні чи натисканні покращують візуальне сприйняття сайту. Вони реалізуються за допомогою простих JS-скриптів або ж ідуть разом із CSS.

Асиметричні оновлення (AJAX). У випадку потреби впровадити часткове оновлення сторінки (наприклад, фільтрація товарів без повного перезавантаження), це можливо завдяки JavaScript. За допомогою асинхронних запитів JavaScript може отримати нові дані з серверу та оновити лише конкретну частину сторінки.

Зворотній зв’язок та інтерактивність. Приміром, після додавання товару до кошика можливо одразу ж показати сповіщення з підтвердженням чи оновити лічильник товарів – все це реалізовано на рівні JavaScript. Такі функції, хоч і не критичні з точки зору логіки, значно покращують зручність для користувача.

Стандартизація та підтримка у всіх браузерах. JavaScript – це мова, що підтримується всіма сучасними браузерами без використання додаткових плагінів. Це робить її надійним інструментом для впровадження функцій, які коректно працюватимуть на будь-якому пристрої користувача.

У розробці комплексних SPA зазвичай використовують бібліотеки на зразок React або Angular. Але для класичного серверного рендерингу (як у нашому випадку) їх використання є надмірним. Базового JavaScript у поєднанні з Bootstrap та HTML/CSS цілком достатньо для реалізації необхідних клієнтських сценаріїв.

Крім того, використання чистого JavaScript сприяє зменшенню кількості зовнішніх залежностей, спрощенню структури проєкту та зниженню порогу входження для нових

учасників команди.

JavaScript – це легкий, гнучкий та універсальний інструмент, що дає змогу втілювати взаємодію з користувачем на клієнтському рівні без зайвих ускладнень. У проєкті інтернет-магазину «М-фурнітура» він використовується для базової динаміки, покращення UX та інтерактивності. Його застосування є виправданим як з технічної, так і з практичної точки зору.

Для створення зручного, адаптивного та привабливого інтерфейсу інтернет-магазину «М-фурнітура» було обрано фреймворк Bootstrap. Це популярна front-end бібліотека, що дозволяє значно полегшити верстку, прискорити процес розробки інтерфейсу та забезпечити кросбраузерність та адаптивність без глибокого знання CSS.

Bootstrap – це набір CSS та JavaScript-компонентів, розроблений для швидкого дизайну веб інтерфейсів. Він містить готові стилі для найпоширеніших елементів сторінки: кнопок, таблиць, форм, навігаційних панелей, карток, модальних вікон та інших. Його ключова перевага – можливість швидко створити акуратний та сучасний інтерфейс, навіть не будучи професійним вебдизайнером.

Переваги використання Bootstrap у проєкті:

Швидкий старт та скорочення часу на верстку. Оскільки проєкт не орієнтований на фронтенд (тобто основна увага зосереджена на серверній логіці), нам було важливо оперативно створити простий, але охайний інтерфейс. Bootstrap надав набір готових класів і шаблонів, що дозволило зосередитися на функціональності, а не на ручній стилізації.

Адаптивний дизайн. Bootstrap базується на системі гнучких сіток (grid system), що автоматично адаптує сторінку під різні розміри екранів: від мобільних телефонів до десктопів. Це надзвичайно важливо для інтернет-магазину, оскільки значна частина користувачів може переглядати сайт з мобільних пристроїв.

Уніфікований вигляд усіх компонентів. Замість самостійного створення стилів для кожного елементу (кнопки, інпуту, таблиці), ми застосували готові стилі Bootstrap. Це забезпечило єдиний стиль інтерфейсу без додаткових зусиль з боку верстальника або дизайнера.

Кросбраузерна сумісність. Bootstrap протестовано та підтримується більшістю сучасних браузерів. Це забезпечує стабільне відображення інтерфейсу незалежно від браузера, що значно скорочує витрати на тестування й виправлення багів.

Готові компоненти для поширених завдань.

Наприклад, у проєкті було використано наступні компоненти Bootstrap:

Форми – для введення даних користувача (реєстрація, оформлення замовлення).

Картки товарів – для представлення продукції.

Навігаційне меню – для легкої навігації сайтом.

Модальні вікна – для сповіщень або підтвердження дій.

Усі ці елементи не потребували складної кастомізації, оскільки мали достатньо професійний вигляд за замовчуванням.

За потреби Bootstrap дозволяє кастомізувати стилі або розширювати функціональність за допомогою власного CSS чи JavaScript. Це зручно для додавання унікальних елементів або адаптації вигляду до брендингу.

Bootstrap має велику кількість прикладів, інструкцій та готових рішень у відкритому доступі. Це полегшує навчання, пошук рішень та зменшує залежність від професійних фронтенд-розробників.

Хоча існують інші CSS-фреймворки, наприклад Bulma, Foundation або Tailwind CSS, вибір Bootstrap був обумовлений його простотою, поширеністю та наявністю компонентів “з коробки”. Tailwind, приміром, вимагає більшої кількості кастомної верстки, що не відповідає нашим вимогам до швидкої розробки. А інші фреймворки менш поширені та мають обмежену підтримку.

Використання Bootstrap у проєкті інтернет-магазину «М-фурнітура» дозволило значно спростити реалізацію інтерфейсу, забезпечити адаптивність та візуальну узгодженість без залучення окремих UI-фахівців. Його застосування є виправданим з технічної, економічної та організаційної точки зору, особливо в умовах, коли головним пріоритетом була стабільна серверна логіка.

У проєкті інтернет-магазину «М-фурнітура» для зберігання та обробки даних була обрана технологія JPA (Java Persistence API) у парі з бібліотекою Hibernate, як її реалізацією. Цей вибір – сучасний стандарт для взаємодії з базами даних у Java-додатках, який дає можливість трансформувати таблиці з бази даних на Java-об’єкти, спрощуючи роботу з транзакціями та уникаючи надлишкової кількості SQL-запитів у коді.

Java Persistence API (JPA) – це офіційна специфікація Java EE (зараз Jakarta EE), яка описує механізм ORM (Object-Relational Mapping). Її функціонал дозволяє перетворювати об’єкти Java на записи в реляційній базі даних та у зворотному напрямку. JPA не є

реалізацією, а лише стандартом. Hibernate, яка використовується в нашому проєкті, є найпопулярнішою реалізацією JPA. Вона має такі переваги як:

ORM (Object-Relational Mapping) підхід. Замість ручного написання SQL-запитів, ми описуємо сутності (Entity) у вигляді Java-класів, застосовуючи анотації (наприклад, @Entity, @Table, @Id, @Column). Такий підхід робить код чистим, логічним і зрозумілим, особливо при роботі з об’єктно-орієнтованим програмуванням.

Швидка розробка. Завдяки Spring Data JPA, ми змогли відійти від написання стандартних CRUD-запитів (створення, читання, оновлення, видалення), застосовуючи інтерфейси JpaRepository. Це дозволило сконцентруватися на бізнес-логіці, а не на повторюваному SQL-коді.

Легкість налаштування

Інтеграція зі Spring Boot дає змогу без ускладнень підключити базу даних, задавши декілька параметрів у application.properties, і вся система працюватиме без потреби у складних конфігураціях.

Міграція структури бази через JPA. У проєкті реалізовано підхід, при якому схема бази даних формується автоматично на основі Java-сутностей. Це дозволяє уникнути ручного створення таблиць та синхронізує модель даних у коді з реальною базою.

Можливість розширення. У майбутньому можливе додавання складних зв’язків між сутностями, наприклад @OneToMany, @ManyToOne, @ManyToMany. Це забезпечить зручне описання взаємин між товарами, категоріями, замовленнями та користувачами, не потребуючи складних SQL-запитів.

Підтримка транзакційності. JPA інтегрована зі Spring Transaction Management, що забезпечує контроль цілісності даних. Наприклад, при оформленні замовлення, всі дії (створення замовлення, оновлення складу, збереження інформації про покупця) можуть виконуватися в одній транзакції. У випадку помилки — зміни не буде збережено.

Кешування та оптимізація запитів. Hibernate, як реалізація JPA, має підтримку кешу другого рівня, що може підвищити продуктивність при роботі з часто використовуваними об'єктами. Крім того, можна керувати завантаженням зв’язаних об’єктів (lazy vs eager fetching), оптимізуючи продуктивність.

Надійність і масштабованість. JPA – це стандарт, що використовується в тисячах Java-проєктів у всьому світі. Він є масштабованим, безпечним та добре задокументованим, що робить його надійним вибором для великих вебзастосунків, у тому числі й для інтернет-магазинів.

У нашому проєкті JPA використовується для таких сутностей:

Product (Товар) — містить поля назви, опису, ціни, кількості на складі.

User (Користувач) — зберігає дані покупців.

Order (Замовлення) — об’єднує інформацію про покупки та зв’язується з товарами та користувачами.

Для кожної сутності створено відповідні @Entity класи та інтерфейси JpaRepository. Це дозволило мінімізувати ручне написання запитів та зменшити ризики помилок.

Використання JPA дало змогу зробити зберігання та обробку даних у проєкті інтернет-магазину ефективним, зручним і зрозумілим. Завдяки інтеграції зі Spring Boot та використанню Spring Data JPA, реалізація повноцінного доступу до бази даних стала простою та гнучкою. Це значно покращило якість і швидкість розробки проєкту.

2.2 Вибір середовища розробки та баз даних

На етапі створення софту визначальне значення має вірний вибір інструментів, що забезпечать ефективність, надійність та зручність у розробці. Одним з центральних елементів у цьому процесі є інтегроване середовище розробки (IDE). IDE дає змогу об'єднати всі стадії розробки — від написання коду до його компіляції, запуску, тестування, налагодження і розгортання.

У дипломному проєкті для реалізації інтернет-магазину «М-фурнітура» було обрано середовище розробки IntelliJ IDEA, яке надає всі необхідні інструменти для повноцінної розробки софту на базі Java та Spring Boot.

IntelliJ IDEA – це потужне IDE, розроблене компанією JetBrains, яке з часу свого виходу здобуло широке визнання серед мільйонів розробників по всьому світу. Середовище регулярно оновлюється, підтримує найновіші версії мов програмування та фреймворків, включає функціонал штучного інтелекту для підказок, і має чудову інтеграцію з системами контролю версій, сервісами CI/CD, базами даних, хмарними інструментами тощо. Для проєктів на базі Spring Boot IntelliJ IDEA надає безпрецедентну підтримку: від автозаповнення анотацій до вбудованого запуску мікросервісів.

Середовище розробки важливе не тільки для написання коду. Воно також дозволяє організовувати структуру проєкту, інтегрувати зовнішні залежності, працювати з шаблонами, перевіряти синтаксис та логіку, керувати процесом налагодження, виконувати юніт- та інтеграційне тестування, аналізувати продуктивність і здійснювати рефакторинг коду. Усі ці аспекти є надзвичайно важливими у розробці навіть невеличких застосувань, не кажучи вже про повноцінні вебзастосунки з базою даних,

авторизацією та декількома модулями.

В межах реалізації дипломного проєкту IntelliJ IDEA забезпечила повний цикл розробки — від генерації Spring Boot-проєкту до запуску локального сервера, підключення до PostgreSQL, написання контролерів, сервісів, репозиторіїв, тестування запитів та логування. Завдяки високому рівню зручності і підтримки сучасних технологій, саме це середовище було обрано як основне IDE у проєкті.

IntelliJ IDEA — один з найбільш функціональних IDE, і велика частина його переваг полягає у величезній кількості вбудованих інструментів, що дають змогу розробнику зосередитись на написанні якісного коду, а не витрачати час на рутинні дії. Серед ключових можливостей варто виділити:

Підтримка Java та інших мов. IntelliJ IDEA була розроблена з основним акцентом на Java, але з роками IDE стала мультифункціональною. Вона підтримує Kotlin — мову, яку також розвиває JetBrains, а також Groovy, Scala, JavaScript, TypeScript, SQL, HTML/CSS та інші. Усі ці мови можна використовувати одночасно в одному проєкті, а IntelliJ самостійно розпізнає, яку мову застосовано, і підлаштовує під це автозаповнення, перевірку синтаксису, інтерпретацію помилок та інші функції. Для Java IntelliJ IDEA надає максимально повну підтримку: усі ключові елементи мови (класи, інтерфейси, абстракції, потоки, анотації, лямбди) обробляються у реальному часі, підказки з'являються миттєво, а помилки синтаксису або логіки підсвічуються ще до компіляції.

Автодоповнення коду (Code Completion). Автодоповнення – одна з найцінніших функцій для кожного розробника. IntelliJ аналізує контекст, тип змінної або об'єкта, очікуване повернення методу, і пропонує найбільш релевантні варіанти. Наприклад, якщо об'єкт є списком, IDE запропонує методи add(), remove(), get() тощо. Якщо це клас Spring Controller – система підкаже всі допустимі анотації @GetMapping, @RequestParam, @Autowired та допоможе їх коректно використати. Це зменшує кількість помилок та прискорює процес написання коду.

Рефакторинг. Інструменти рефакторингу в IntelliJ IDEA дають змогу вносити структурні зміни у код, не порушуючи його цілісності. Наприклад, можна безпечно перейменувати змінну, клас або метод — середовище знайде всі посилання на них в усьому проєкті й оновить їх автоматично. Також доступні складніші рефакторинги: виділення частини коду в окремий метод або клас, інверсія залежностей, заміна наслідування на композицію тощо.

Перевірка коду у реальному часі. Одна з найпотужніших функцій – перевірка коду на ходу. IntelliJ безперервно сканує весь код у проєкті й попереджає про можливі помилки, навіть якщо вони не критичні. Наприклад, IDE вкаже, якщо якась змінна не

використовується, якщо є дублікат коду або якщо метод не закриває ресурси належним чином. Це допомагає підтримувати код у чистому, зрозумілому й безпечному вигляді.

Величезна екосистема плагінів – ще одна сильна сторона IntelliJ IDEA. Користувач має можливість встановити розширення для Docker, Kubernetes, Redis, Kafka, Swagger, GraphQL, Lombok, JPA Visualizer та ін. Наприклад, у дипломному проєкті використовувався плагін 6 для роботи з PostgreSQL: він дозволив напряму підключити базу до IDE, виконувати SQL-запити, перевіряти структуру таблиць та бачити зв'язки між ними. Усі плагіни можна знайти в інтегрованому магазині, їх встановлення займає декілька кліків.

Зручність роботи з проєктами. 6 Однією з основних переваг IntelliJ IDEA є високий рівень зручності у створенні, налаштуванні та веденні проєктів. Розробник отримує можливість зосередитися на бізнес-логіці та архітектурі програми, оскільки більшість рутинних завдань середовище бере на себе.

Створення Maven і Spring Boot-проєктів. При створенні нового проєкту IntelliJ IDEA пропонує зручний майстер (wizard), що дає змогу обрати тип проєкту: Java, Kotlin, Maven, Gradle, Spring Boot тощо. У випадку дипломного проєкту використовувався Maven-проєкт зі Spring Boot. IntelliJ автоматично створює всі необхідні файли — pom.xml, структуру папок, базовий клас запуску Application.java, а також додає залежності, які обирає розробник під час конфігурації (наприклад, 1 Spring Web, Spring Security, Spring Data JPA, PostgreSQL Driver та ін.).

Інтеграція з Spring Initializr дає змогу створювати Spring Boot-проєкти без виходу з IDE. Також середовище саме підкаже, які залежності найкраще використовувати для типу застосунку, що розробляється. Наприклад, якщо обирається модуль безпеки, IDE запропонує додати Spring Security, якщо потрібна робота з БД — JPA та драйвер відповідної СУБД.

Інтеграція з Git. У сучасній розробці неможливо уявити собі проєкт без використання 9 систем контролю версій, зокрема Git. IntelliJ IDEA має вбудовану повноцінну підтримку Git, що дає змогу виконувати всі основні дії без переходу до терміналу. Можна створювати нові гілки, робити коміти, зливати гілки, відстежувати зміни в коді, вирішувати конфлікти, переглядати лог змін, додавати .gitignore — усе це доступно прямо з графічного інтерфейсу.

Завдяки інтеграції з GitHub розробник може буквально в кілька кліків створити новий репозиторій, завантажити проєкт, клонувати наявний репозиторій, створити pull request чи провести code review. Ці функції особливо корисні в командній роботі або при веденні портфоліо на GitHub, як це було реалізовано у дипломному проєкті.

Окрім Git, IntelliJ підтримує також інші VCS: Mercurial, Perforce, Subversion — усе це доступно через плагіни та налаштування.

Організація структури проекту. IDE чітко розмежовує папки проєкту: основний код (src/main/java), ресурси (src/main/resources), тестовий код (src/test/java), зовнішні бібліотеки, залежності, налаштування тощо. Завдяки цьому легко орієнтуватися у великому проєкті. Крім того, IntelliJ автоматично індексує весь код і дає змогу миттєво переходити до будь-якого класу, методу або змінної, використовуючи лише кілька клавіш.

Spring Boot є ключовим фреймворком у дипломному проєкті, а IntelliJ IDEA забезпечує глибоку інтеграцію з ним, що значно полегшує процес розробки.

Після того, як додано залежності Spring Boot у pom.xml, IDE автоматично розпізнає проєкт як Spring-проєкт. Вона активує додаткові інструменти та можливості, зокрема: автозаповнення анотацій, підсвітку конфігурацій у файлах application.properties або application.yml, індексацію бінів та створення графу залежностей. Наприклад, при додаванні анотації @Service або @Controller, IntelliJ IDEA розуміє роль цього класу в контексті Spring, дозволяє відстежувати ін’єкції, переходити від місця створення до місця використання та навпаки. Це значно прискорює налагодження проєкту та зменшує кількість помилок.

Запуск і налагодження API. Інтегрований інструмент Spring Boot Run Configuration дає змогу запускати застосунок без додаткового налаштування. Достатньо натиснути одну кнопку — і проєкт компілюється, збирається у .jar та запускається на вбудованому сервері (наприклад, Tomcat).

Також IntelliJ дозволяє налагоджувати застосунок у реальному часі. Можна встановлювати breakpoint-и, переглядати значення змінних, дивитися call stack, відстежувати потоки. Це особливо важливо під час налагодження бізнес-логіки або роботи з базою даних.

Інспекція залежностей. Інструменти Maven у IntelliJ дозволяють відстежувати всі залежності проєкту, бачити їх ієрархію, версії, конфлікти. Якщо якась бібліотека відсутня або викликає конфлікт, IDE відразу підсвітить це. Також можливе автоматичне оновлення залежностей до останніх стабільних версій.

Підтримка Devtools і Live Reload. Плагін Spring Boot Devtools інтегрується з IntelliJ без додаткових налаштувань. Це дає змогу вмикати автоматичне перезавантаження застосунку після зміни коду або конфігурацій. У поєднанні з інструментами для Front-End (наприклад, LiveReload або плагін для браузера), оновлення веб-інтерфейсу можна бачити миттєво, що прискорює цикл «розробка–перевірка».

IntelliJ IDEA виявилася ключовим інструментом під час розробки дипломного проєкту – інтернет-магазину «М-фурнітура», який реалізували, використовуючи Java, Spring Boot, JPA, Spring Security та PostgreSQL. Широкий функціонал IDE значно прискорив процес створення програмної частини застосунку.

Створення контролерів. Контролери в Spring Boot відповідають за обробку HTTP-запитів та видачу відповідей. В середовищі IntelliJ створення контролерів стало зручним та швидким завдяки вбудованим шаблонам, автодоповненню анотацій та перевірці коректності написаного коду.

При створенні класу контролера IDE автоматично пропонує анотації @RestController, @RequestMapping, @GetMapping, @PostMapping та інші.

У дипломному проєкті були розроблені контролери для наступних функціональних блоків:

реєстрація та авторизація користувачів;

додавання, редагування та видалення товарів;

оформлення замовлень;

пошук та фільтрація товарів за категоріями.

IntelliJ допомагає в режимі реального часу підсвічувати помилки, відсутність залежностей чи некоректне використання анотацій. Наприклад, якщо пропущено параметр @RequestParam, IDE підкаже про необхідність його додати. Також можливе швидке створення відповідних DTO-класів, що суттєво скорочує час на рутинну роботу.

У дипломному проєкті для збереження даних використовувалася реляційна СУБД PostgreSQL. Для взаємодії з нею обрали 1 Spring Data JPA, який автоматизує роботу з базою за допомогою репозиторіїв.

IntelliJ IDEA дала змогу без проблем створити Entity-класи та репозиторії. При створенні класу з анотацією @Entity IDE автоматично додає імпорти, підказує назви колонок, типи полів та їх анотації (@Id, @GeneratedValue, @Column тощо). Завдяки тісній інтеграції з JPA, середовище відображає мапінг між класами та таблицями бази даних.

Крім того, в середовищі є вбудований SQL-редактор та переглядач бази даних. Це дає змогу безпосередньо підключатися до бази, виконувати SQL-запити, переглядати таблиці, робити дампи – все це без використання окремих утиліт, таких як pgAdmin. Таке поєднання зручності й функціональності значно зменшує час налагодження логіки збереження та отримання даних.

Сервісний шар у дипломному проєкті відповідає за бізнес-логіку: перевірку прав доступу, перевірку валідності даних, роботу з репозиторіями. В IntelliJ IDEA за допомогою автогенерації методів, швидкого створення класів та автоматичної вставки залежностей (@Autowired або через конструктори) вдається оперативно налаштувати цей рівень архітектури.

При написанні методів IDEA підказує, які виклики існують у відповідному репозиторії, які параметри потрібно передати. Також можливий швидкий рефакторинг – перейменування методів, виділення частин логіки в окремі сервіси, що дає змогу підтримувати код у чистому та структурованому вигляді.

Для реалізації авторизації та аутентифікації використовувався Spring Security. IntelliJ допомагає правильно налаштувати конфігураційні класи безпеки, підтримує автодоповнення для специфічних методів (authorizeHttpRequests, formLogin, logout та інші), а також дає змогу створити необхідні конфігурації з використанням шаблонів. У разі зміни структури контролерів або додавання нових URL-шляхів, IntelliJ допомагає оперативно оновити правила доступу.

10 Переваги IntelliJ IDEA над іншими IDE. Хоча на ринку існує кілька популярних середовищ розробки для Java, таких як Eclipse та NetBeans, IntelliJ IDEA завоювала визнання серед професіоналів завдяки поєднанню функціональності, зручності інтерфейсу та високої продуктивності.

Порівняння з Eclipse. Eclipse – це безкоштовне середовище, яке тривалий час було стандартом 10 для розробки на Java. Проте, порівняно з IntelliJ, воно має кілька недоліків:

Повільніша індексація та компіляція коду.

Менш інтуїтивний інтерфейс користувача.

Менша інтеграція з сучасними фреймворками (Spring Boot, JPA, Maven).

Складна система плагінів, яка може призводити до конфліктів.

IntelliJ, навпаки, забезпечує майже миттєвий аналіз коду, легку навігацію по проєкту, а також глибоку інтеграцію з усіма популярними інструментами сучасної Java-розробки.

Порівняння з NetBeans

NetBeans – офіційна IDE від Oracle, яка також підтримує Java та інші мови. Однак вона менш популярна серед професіоналів. Основні недоліки:

Відсутність повноцінної підтримки нових стандартів Spring Boot.

Обмежений інструментарій для роботи з Git та іншими VCS.

Менш розвинена система підказок та рефакторингу.

IntelliJ IDEA, навпаки, має активну спільноту, регулярні оновлення, багату документацію, а також платну Ultimate-версію з розширеним функціоналом для корпоративної розробки.

Продуктивність та зручність. IntelliJ оптимізована для роботи з великими проєктами. Вона швидко індексує код, кешує результати компіляції, дозволяє розробнику працювати навіть під час «підвантаження» проєкту. Крім того, зручний UI/UX, кастомізація тем, підтримка різних мов і форматів файлів робить її приємною для щоденного використання.

Велика спільнота і підтримка. Ще одна важлива перевага – активна спільнота користувачів та розробників, яка дозволяє швидко знайти відповіді на питання, отримати підтримку або знайти плагін для будь-якої задачі. Також є офіційна підтримка від JetBrains – компанії-розробника IntelliJ IDEA.

Переваги PostgreSQL для даного дипломного проєкту:

Однією з причин вибору PostgreSQL стало його ліцензування за відкритим кодом. Це означає, що його дозволено безперешкодно використовувати як в освітніх, так і в комерційних умовах, не зазнаючи жодних фінансових зобов'язань. Це особливо доречно для учбових проєктів, стартапів або невеликих колективів, які не мають ресурсів на оплату дорогих комерційних розв'язків. PostgreSQL - безкоштовна альтернатива, яка не поступається за функціональністю.

Також важливою перевагою є підтримка складних SQL-запитів. В багатьох випадках базові CRUD-операції не задовольняють всі вимоги застосунку, особливо коли йдеться про фільтрацію, сортування, агрегацію чи створення звітів. PostgreSQL дозволяє писати комплексні запити, використовувати вкладені підзапити, 5 віконні функції, CTE (Common Table Expressions), що суттєво спрощує роботу з аналітичними даними.

Ще одним аргументом на користь PostgreSQL стала її гнучка система типів даних. На відміну від деяких інших СУБД, PostgreSQL дає змогу 5 створювати власні типи даних, а також підтримує масиви, JSON, XML, ENUM тощо. Для проєкту інтернет-магазину це означає можливість зберігати додаткову інформацію про товари, замовлення чи користувачів, уникаючи необхідності створення великої кількості додаткових таблиць. Наприклад, дані про характеристики товарів можливо зберігати у вигляді JSON-структур.

Щодо масштабованості, PostgreSQL добре справляється з великими обсягами даних. Завдяки індексації, партиціюванню таблиць та використанню кешування запитів, її можна ефективно застосовувати не тільки в маленьких навчальних проєктах, а й у масштабних продакшн-системах. Це дає змогу почати з простого рішення, проте, за потреби, масштабувати його без значних змін в архітектурі.

Також PostgreSQL має чудову інтеграцію з Java та Spring Boot – основними технологіями, які використовуються в дипломному проєкті. Існує офіційний JDBC-драйвер, який дозволяє легко підключити СУБД до застосунку. Крім того, 1 Spring Data JPA має повну підтримку PostgreSQL, що дає можливість створювати запити на рівні об’єктів, уникаючи необхідності писати багато SQL-коду вручну. 11 Це значно прискорює процес розробки та зменшує вірогідність помилок.

У дипломному проєкті PostgreSQL виступає як централізоване сховище даних. Завдяки своїй надійності, гнучкості та сумісності зі Spring Boot, вона дозволяє зручно реалізувати ключові бізнес-функції інтернет-магазину. Вся інформація про користувачів, товари, замовлення, категорії продукції та нагадування зберігається саме в цій СУБД.

Збереження користувачів – одна з головних функцій. В таблиці users містяться акаунти з такими даними, як логін, пароль (захешований), ім'я, email, телефон тощо. PostgreSQL дозволяє забезпечити унікальність логіна або електронної пошти, використовуючи унікальні обмеження. Крім того, за допомогою індексів досягається висока швидкодія при аутентифікації.

Моделювання товарів і категорій також ефективно реалізується у PostgreSQL. Таблиці products і categories мають зв'язок «багато до одного» (кожен товар належить до однієї категорії). Завдяки підтримці зовнішніх ключів (foreign key), PostgreSQL гарантує цілісність даних. Наприклад, не можливо випадково видалити категорію, якщо до неї прив’язано хоча б один товар.

Робота із замовленнями реалізована через взаємопов'язані таблиці orders і order_items, що дає можливість зберігати як загальну інформацію про замовлення (дата, статус, покупець), так і деталі кожного товару в ньому (кількість, ціна, назва товару на момент покупки). Таку структуру надзвичайно легко реалізувати в PostgreSQL, а складні SQL-запити дозволяють будувати аналітику (наприклад, загальний прибуток за місяць чи найпопулярніші товари).

Ще один аспект – зберігання нагадувань. Для реалізації функціоналу нагадувань (наприклад, про сплив терміну доставки або необхідність оновити дані) можна застосувати таблицю reminders, де зберігається текст нагадування, час його створення та час спрацювання. Тип timestamp with time zone, який підтримує PostgreSQL, дає змогу точно зберігати дату та час, враховуючи часові пояси – це особливо важливо для

нагадувань.

Переваги PostgreSQL 16 для 9 забезпечення надійності та безпеки 16 в проєкті

Однією з найважливіших вимог до системи інтернет-магазину є надійність зберігання даних. Кожна транзакція – оформлення замовлення, реєстрація користувача, оновлення товарів – повинна відбуватись без помилок і з гарантією цілісності. PostgreSQL ідеально підходить для таких завдань завдяки підтримці ACID-властивостей (атомарність, узгодженість, ізольованість, довговічність).

Атомарність гарантує, що транзакція або виконується повністю, або не відбувається взагалі. Це важливо, наприклад, при створенні замовлення – і дані про замовлення, і його позиції повинні бути збережені одночасно.

Узгодженість забезпечує перехід бази з одного коректного стану в інший – наприклад, товар, котрий відсутній, не може бути доданий до замовлення.

Ізольованість не дає змогу паралельним транзакціям впливати одна на одну – декілька користувачів можуть оформляти замовлення одночасно без конфліктів.

Довговічність гарантує, що дані не будуть втрачені навіть у разі збою – після коміту транзакції інформація зберігається назавжди.

Ще одна важлива перевага PostgreSQL – розвинені механізми резервного копіювання та відновлення даних. Вона дозволяє:

Робити повні та інкрементальні бекапи;

Використовувати WAL-файли (журнали змін), що дає змогу відновити базу до конкретного моменту;

Створювати реплікацію (основна база та одна або більше резервних), що особливо корисно для відмовостійких систем.

В контексті дипломного проєкту це дає можливість забезпечити стабільну роботу інтернет-магазину навіть у разі збою сервера або втрати даних.

PostgreSQL також приділяє велику увагу безпеці:

Підтримка SSL-з'єднань захищає передавання даних між сервером і клієнтом;

Можна детально налаштовувати доступи до бази за допомогою системи ролей та GRANT-політик;

Реалізовано журналювання дій користувачів і запитів, що дозволяє відстежувати

потенційні загрози або некоректні дії.

Завдяки цьому PostgreSQL виступає не просто як сховище, а як інструмент безпечної обробки бізнес-логіки.

Масштабованість, гнучкість та підтримка спільноти PostgreSQL

Інтернет-магазин – це динамічний тип проєкту, котрий з часом може суттєво зростати. Кількість користувачів, товарів, замовлень, а також статистичних та аналітичних даних постійно збільшується. Саме тому важливою перевагою PostgreSQL є масштабованість. Вона може обробляти як невеликі, так і великі обсяги даних без втрати продуктивності.

PostgreSQL дозволяє:

Використовувати шардінг і реплікацію для розподілу навантаження між декількома серверами;

Забезпечувати паралельне виконання запитів для кращої продуктивності аналітичних задач;

Гнучко налаштовувати кешування, індекси, планувальник запитів – для оптимізації доступу до великої кількості записів.

Особливо варто відзначити розширюваність PostgreSQL. Це одна з небагатьох СУБД, котра дозволяє:

Створювати власні типи даних (наприклад, для зберігання координат або складних JSON-структур);

Реалізовувати власні функції на різних мовах – SQL, PL/pgSQL, Python, JavaScript, C;

Додавати розширення (наприклад, pg_stat_statements, postgis, uuid-ossp), котрі розширюють функціональність бази без втручання в її ядро.

Це надає проєкту гнучкість: можна адаптувати базу під свої потреби, не змінюючи основного коду програми.

Окрім технічних переваг, важливо також, що PostgreSQL має велику, активну та підтримуючу спільноту. В Інтернеті легко віднайти документацію, відповіді на StackOverflow, офіційні гайди, блоги та відеоуроки. Це особливо цінно для студентського та комерційного проєктів – у разі труднощів завжди є на кого спертися.

Вибір PostgreSQL для дипломного проєкту – це не випадковість, а обґрунтоване технічне рішення. База даних повинна бути:

Безпечною;

Надійною;

Швидкою;

Масштабованою;

Гнучкою.

PostgreSQL відповідає всім цим вимогам. Вона ідеально поєднується з Java та Spring Boot, які я використовую в основі проєкту, а також повністю підтримує JPA/Hibernate, що спрощує взаємодію з базою даних через об’єктно-реляційне відображення.

Для інтернет-магазину, де важлива цілісність замовлень, безпека облікових даних користувачів і стабільна робота навіть під час пікового навантаження – PostgreSQL є оптимальним вибором, котрий дозволяє будувати масштабований, надійний і сучасний вебсервіс.

3 РОЗРОБКА САЙТУ ІНТЕРНЕТ-МАГАЗИНУ

3.1 Розробка структури та опис інтерфейсу сайту

Структруру сайту «М-фурнітура» побудовано за принципами мінімалізму та чіткої функціональності. На кожній сторінці присутні три основні фіксовані блоки: шапка (header), мегаменю та підвал (footer). Така структура гарантує користувачам однорідність розташування елементів і швидке звикання до навігації. Розроблена структура сайту «М-фурнітура» показана на рисунку 3.1:

Рисунок 3.1 – Структура сайту «М-фурнітура»

У блоці «header» по центру розташовано логотип, який виконує подвійні функції: формує впізнаваний бренд і повертає на головну сторінку при кліку. Поруч — поле пошуку, оптимізоване для нечітких запитів, що дозволяє швидко знайти товар навіть за неповною назвою. Праворуч від пошуку знаходяться іконки для швидких дій: телефон для зв’язку, особистий кабінет і кошик із бейджем поточної кількості товарів. Розбиття на зони (ліворуч – логотип і навігація, центр – пошук, праворуч – дії користувача) відповідає кращим практикам UX і запобігає візуальному перевантаженню.

Блок «header» показаний на рисунку 3.2:

Рисунок 3.2 – Блок «header»

У блоці «footer» зібрана допоміжна інформація: про компанію, контакти, посилання на

політику конфіденційності та умови доставки. Контрастний фон і чітка типографіка відокремлюють цей блок від основного контенту, роблячи навігацію інтуїтивною. Тут зібрана допоміжна інформація: про компанію, контакти, посилання на політику конфіденційності та умови доставки. Контрастний фон і чітка типографіка відокремлюють цей блок від основного контенту, роблячи навігацію інтуїтивною.

Блок «footer» показаний на рисунку 3.3:

Рисунок 3.3 – Блок «footer»

Мегаменю категорій. Навігаційне мегаменю складається з чотирьох вертикальних стовпців, кожен із яких охоплює один із асортиментних напрямків:

Функціональна фурнітура

Напрямні кулькові

Фурнітура для кухонних меблів

Завіси

Складові частини меблів

Підйомні механізми

Інструмент

Механічний інструмент

Будівельні інструменти

Вимірювальні інструменти

Інструменти для обробки

Аксесуари для електроінструментівЕлектроніка

Світлодіодні стрічки

Кабелі та комплектуючі для з’єднання

Трансформатори 12 V

Світлодіодні світильники та кліпси

Профілі GLAX

Лицьова фурнітура

Гачки меблеві

Опори та каркаси для столів

Ніжки меблеві

Ролики меблеві

Ручки меблеві

Кожна категорія представлена окремою кнопкою з достатнім інтервалом, що мінімізує ризик помилкового кліку. Активний пункт позначено білим текстом на синьому фоні, неактивні — обведені, але без заливки. Такий дизайн дозволяє чітко розрізняти вибрані підкатегорії та орієнтуватися в меню миттєво.

Користувацьке тестування показало, що мегаменю прискорює пошук підкатегорії в середньому на 20 % порівняно зі звичайним дропдаун-листом, оскільки всі опції видно одразу.

Відображення підкатегорій у вигляді окремих кнопок із рівномірним відступом підсилює сприйняття контенту та знижує ймовірність випадкового кліку на сусідній пункт.

Колірна схема — основний синій фон із білим текстом для активного пункту й обведенням для неактивних.

Меню категорії «Функціональна фурнітура» показане на рисунку 3.4:

Рисунок 3.4 – Меню категорії «Функціональна фурнітура»

Меню категорії «Інструмент» показане на рисунку 3.5

Рисунок 3.5 – Меню категорії «Інструмент»

Меню категорії «Електроніка» показане на рисунку 3.6:

Рисунок 3.6 – Меню категорії «Електроніка»

Меню категорії «Лицева фурнітура» показане на рисунку 3.7:

Рисунок 3.7 – Меню категорії «Лицева фурнітура»

У користувацькому тестуванні виявлено, що такий підхід прискорює пошук потрібної

підкатегорії на 20% порівняно з простим випадаючим списком, оскільки всі опції видно одразу.

Картки товарів

Кожна картка містить:

Зображення товару з hover-ефектом (легке збільшення)

Назва товару

Ціна з відображенням

Кнопка «Додати в кошик» при наведенні

Приклад відображення товару показаний на рисунку 3.8:

Рисунок 3.8 – Приклад відображення товару

Структуру картки спроектовано таким чином, щоб вміст вміщався в однакові за розміром рамки, забезпечуючи акуратний вигляд сітки і рівномірний розподіл візуальної ваги. При наведенні на Товар відображається іконка кошику.

Аналітика відображає: користувач проводить у середньому 2,5 секунди на візуальний аналіз кожної картки, тому інформація під зображенням мінімізована до найважливіших пунктів.

Сторінка детального перегляду товару складається з двох ключових блоків:

Головне зображення займає понад 50 % ширини контентної зони та підтримує функцію zoom-in при наведенні. Під ним розташовані мініатюри з додатковими ракурсами.

Характеристики оформлено у двоколоночній таблиці з рядками: матеріал, розміри, вага, країна-виробник. Далі йде текстовий опис, розбитий на тематичні абзаци. Під описом розташовано комерційний блок з двома фіксованими кнопками: «Додати в кошик» (синя) та «Купити в один клік» (зелена). Ці кнопки залишаються на екрані під час скролу, що підвищує конверсію.

При натисканні на «Додати в кошик» з’являється спливаюче повідомлення про успішне додавання. Кнопка «Купити в один клік» викликає модальну форму для введення номера телефону та швидкого оформлення замовлення.

Приклад відображення товару показаний на рисунку 3.9:

Рисунок 3.9 – Приклад відображення товару

Комерційний блок. Під описом – кнопки взаємодії:

Додати в кошик (контрастна синя) та Купити в один клік (зеленим), обидві фіксуються при скролі вікна.

Фіксована панель підтримує конверсію навіть при перегляді детального опису.При нажатті «Додати в кошик» висвітлює повідомлення «Товар додано в кошик», показане на рисунку 3.10:

Рисунок 3.10 – Приклад відображення товару

Якщо нажати замовити в один клік відкриється форма в яку треба ввести номер телефону:

Кошик та оформлення

Сторінка кошика

У кошику товари відображаються таблично:

Стовпець із мініатюрою та назвою

Кількість із полями «+»/«–»

Ціна за одиницю та загальна ціна позиції

Кнопка видалення

Знизу – проміжна сума.

Кнопка «Оформити замовлення» виділена кольором та займає всю ширину контентної зони.

Кнопка «Оформити замовлення» показана на рисунку 3.11:

Рисунок 3.11 – Кнопка «Оформити замовлення»

Мої замовлення. Після авторизації користувач бачить, посиланнями на «Мої замовлення», Інтерфейс виконаний в одному стилі з головним сайтом, що забезпечує цілісність UX. Вікно «Оформити замовлення» показане на рисунку 3.12:

Рисунок 3.12 – Вікно «Оформити замовлення»

Список замовлень. Розділ відображає таблицю зі зведеними даними:

13 Номер замовлення

13 Дата оформлення

13 Загальна сума

13 Статус

3.2 Опис організації та налаштування бази даних для розробленого сайту

У сучасному програмуванні важлива ефективна взаємодія між застосунком та сховищем даних. Одним з найпоширеніших рішень для налагодження такої взаємодії в екосистемі Java є комбінація Spring Boot з PostgreSQL, використовуючи ORM (Object-Relational Mapping). Це дає змогу зменшити кількість типового коду, посилити захист 7 доступу до даних та запропонувати зручний рівень абстракції для взаємодії з базою.

PostgreSQL – це потужна 15 об’єктно-реляційна система управління базами даних (СУБД), 7 яка підтримує складні типи даних, транзакції, індексацію, функції, тригери й багато інших можливостей. Spring Boot, в свою чергу, – це фреймворк, що спрощує створення веб-застосунків на базі Spring, дозволяючи швидко запускати проєкти зі зручною конфігурацією. ORM – це підхід, який дозволяє з’єднати об’єктно-орієнтовану модель програми з реляційною моделлю бази даних, зменшуючи розбіжність між ними.

Object-Relational Mapping (ORM) автоматизує перетворення даних між об'єктами у коді та записами в базі даних. Завдяки ORM розробнику не потрібно власноруч писати SQL-запити для кожної операції зчитування чи запису. Це особливо корисно в Java-додатках, де переважає об’єктна модель.

Ключові принципи ORM

Абстрагування від SQL

Розробник працює з даними через об’єкти (класи, методи, поля), а ORM генерує SQL-запити автоматично. Наприклад, операції CRUD (Create, Read, Update, Delete) виконуються через методи save(), findById(), delete() без прямого використання INSERT або SELECT. Це знижує обсяг шаблонного коду та зменшує ймовірність помилок у синтаксисі SQL.

Відтворення об'єктів на таблиці. Кожен клас сутності (наприклад, User, Product) відповідає окремій таблиці в БД.

Поля класу відповідають стовпцям таблиці, а об'єкти — рядкам.

ORM автоматично обробляє типи даних: наприклад, String в Java перетворюється на

VARCHAR в БД, а LocalDateTime – на TIMESTAMP.

Керування зв'язками між сутностями

ORM підтримує всі види взаємозв’язків між таблицями:

Один-до-одного (наприклад, User Passport).

Один-до-багатьох (наприклад, Blog Post).

Багато-до-багатьох (наприклад, Student Course).

Зв’язки описуються за допомогою анотацій або конфігураційних файлів, а ORM управляє зовнішніми ключами (foreign keys) та операціями JOIN.

Транзакційність та властивості ACID. ORM забезпечує атомарність операцій: якщо одна частина транзакції не виконається, всі зміни будуть скасовані (rollback).

Підтримка ізоляції транзакцій для уникнення колізій при одночасних запитах. У Spring Boot це реалізується за допомогою анотації @Transactional, яка автоматично контролює межі транзакцій.

Кешування та оптимізація. ORM використовує кешування першого рівня (session cache) для зменшення кількості запитів до БД. Ліниве завантаження (lazy loading): дані завантажуються лише за потреби (наприклад, колекція Order у User не завантажується, поки не викликано user.getOrders()).

Пакетна обробка (batch processing) для ефективних масових оновлень. ORM реалізовано в Spring Boot здебільшого за допомогою бібліотеки Hibernate — однієї з найпопулярніших реалізацій ORM для Java.

Архітектура взаємодії Spring Boot з PostgreSQL через ORM. Взаємодія з базою даних в Spring Boot організована через кілька компонентів:

DataSource — джерело підключення до бази даних;

Entity — Java-клас, який відповідає таблиці в базі даних;

Repository — інтерфейс, який забезпечує CRUD-операції;

JPA (Java Persistence API) — стандартна абстракція для ORM;

Hibernate — реалізація JPA, яка виконує безпосередню роботу з базою даних.

Переваги використання ORM в проєкті:

Прискорення розробки. ORM звільняє від ручного написання сотень SQL-запитів, що особливо важливо у великих проектах.

Легка інтеграція зі змінними: модифікація схеми БД через редагування класів-сутностей, а не через SQL-скрипти.

Підвищення безпеки. Заздалегідь скомпільовані запити (prepared statements) запобігають SQL-ін’єкціям.

ORM автоматично екранує спеціальні символи у введених даних.

Стандартизація коду. Код стає більш структурованим: сутності, репозиторії, сервіси чітко розділені.

Використання патернів проектування (наприклад, Unit of Work, Repository) покращує архітектуру програми.

Переносимість між СУБД. ORM дозволяє перемикатися між різними базами даних (наприклад, з MySQL на PostgreSQL) з мінімальними змінами коду.

Використання Hibernate Dialect дає змогу адаптуватися до особливостей конкретної СУБД.

Спрощення тестування. Можливість використовувати імітовані БД (наприклад, H2) для модульних тестів.

Інтеграція з тестовими фреймворками Spring Boot (наприклад, @DataJpaTest).

Ефективна робота зі складними моделями даних. ORM інтуїтивно обробляє ієрархії класів (наприклад, наслідування через стратегії SINGLE_TABLE, JOINED).

Підтримка каскадних операцій: видалення пов’язаних даних при видаленні батьківської сутності.

В проєкті структура має вигляд: контролер сервіс репозиторій база даних.

Налаштування з’єднання з PostgreSQL. Першим кроком при використанні PostgreSQL є налаштування з'єднання в конфігураційному файлі 8 application.properties або application.yml. У цьому файлі визначаються основні параметри, такі як URL, логін, пароль, драйвер та діалект Hibernate.

spring.datasource.url=jdbc:postgresql://localhost:5432/mydb

spring.datasource.username=postgres

spring.datasource.password=yourpassword

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect

Це дозволяє 8 Spring Boot автоматично підключитися до бази PostgreSQL та створити таблиці на основі визначених сутностей.

Пояснення принципу роботи JPA та Hibernate. JPA — це стандартна специфікація Java, яка визначає, як об'єкти Java мають зберігатися в базах даних. Hibernate реалізує цю специфікацію і виконує всю роботу з SQL.

Сутність — це Java-клас, анотації якого описують структуру таблиці в базі. Наприклад, анотація @Entity вказує на те, що клас є сутністю. @Id вказує на первинний ключ, а @Column — на колонку таблиці.

Hibernate автоматично генерує SQL-інструкції на основі структури класу, що дозволяє розробнику працювати з об'єктами, не переймаючись SQL.

Створення та оновлення схеми бази. В налаштуваннях JPA можна вказати, як саме Hibernate має взаємодіяти зі схемою бази. Доступні опції:

none — не виконувати жодних змін в базі;

validate — перевіряти відповідність схем;

update — автоматично оновлювати структуру таблиць;

create — створювати таблиці при кожному запуску;

create-drop — створювати та видаляти таблиці після завершення сесії.

Відношення між сутностями. ORM дозволяє описувати зв'язки між сутностями безпосередньо в коді:

OneToOne — один до одного;

OneToMany — один до багатьох;

ManyToMany — багато до багатьох.

Кожен зв'язок має свої особливості реалізації, залежно від того, яка сторона є власником зв'язку (owning side) та як відбувається завантаження даних (ліниве чи

жадібне завантаження).

Репозиторії та використання Spring Data JPA. Spring Data JPA — це підпроєкт Spring, що дозволяє автоматизувати створення репозиторіїв. Достатньо оголосити інтерфейс, який розширює JpaRepository, і Spring автоматично реалізує всі основні методи: save(), findById(), delete() тощо.

Кешування та оптимізація. Hibernate підтримує два рівні кешування:

Перший рівень (вбудований) — працює в межах сесії;

Другий рівень — можливість зберігати об’єкти між сесіями, часто реалізується через сторонні бібліотеки.

Також ORM дозволяє використовувати лениве (LAZY) або жадібне (EAGER) завантаження для оптимізації запитів.

Керування транзакціями. Spring Boot забезпечує автоматичне управління транзакціями за допомогою анотації @Transactional. Це дозволяє гарантувати, що набір змін до бази даних буде або повністю застосовано, або відкочено у разі виникнення помилки.

Використання транзакцій гарантує:

Цілісність даних;

Можливість відкочування змін;

Послідовність у випадках багатьох операцій.

Міграції бази даних. Flyway та Liquibase – популярні бібліотеки для управління версіями баз даних. Вони дозволяють створювати окремі файли міграцій, які виконуються автоматично при запуску додатку, забезпечуючи контрольоване розгортання змін схеми на різні середовища.

Наприклад, Flyway використовує SQL або Java-файли, які мають префікси V1__, V2__ тощо, для версіонування змін.

Безпека при взаємодії з базою даних. ORM підвищує безпеку за допомогою параметризованих запитів, що мінімізує ризик SQL-ін’єкцій. Також Spring Security дає змогу обмежити доступ до ресурсів, зокрема репозиторіїв, і управляти правами користувачів.

Тестування взаємодії з базою. Spring Boot пропонує зручні інструменти для тестування бази даних: @DataJpaTest, H2 (вбудована база), Testcontainers. Це дозволяє перевірити

коректність логіки без підключення до реальної PostgreSQL.

Посилання

Це джерела виділених збігів у вашому документі. Кожен збіг позначено темно-зеленим числом, яке відповідає вказаному тут джерелу. Джерела впорядковані за схожістю — чим вищий бал, тим сильніше збіг.

#	Джерело	%
1	manojlakshan421.medium.com	0.2%
2	ir.nmu.org.ua	0.2%
3	web.kpi.kharkov.ua	0.2%
4	essuir.sumdu.edu.ua	0.1%
5	ela.kpi.ua	0.1%
6	metod.vntu.edu.ua	0.1%
7	dspace.znu.edu.ua	0.1%
8	docs.vntu.edu.ua	0.1%
9	iss.csc.knu.ua	0.1%
10	krs.chmnu.edu.ua	0.1%
11	docs.vntu.edu.ua	0.1%
12	ela.kpi.ua	0.1%
13	oleus.ua	0.1%
14	fishdigital.agency	0.1%
15	ua.emptyq.net	0.1%
16	repository.kpi.kharkov.ua	0.0%

Plag

Дякуємо, що перевірили
свій документ за допомогою
Plag!