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

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Мануйленко Івана Миколайовича*
(ПІБ)

академічної групи *122-18-1*
(шифр)

спеціальності *122 Комп'ютерні науки*
(код і назва спеціальності)

освітньої програми *Комп'ютерні науки*
(назва освітньої програми)

на тему: *Розробка інтелектуальної системи підбору взуття з використанням продукційної моделі бази знань*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*доц. Кабак Л.В.*			
розділів:				
спеціальний	*доц. Кабак Л.В.*			
економічний	*доц. Касьяненко Л.В.*			
Рецензент				
Нормоконтролер	*доц. Гуліна І.Г.*			

Дніпро
2022

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

_____ І.М. Удовик
(підпис) (прізвище, ініціали)

« » _____ 2022 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *122-18-1* (група) *Мануйленко Івана Миколайовича* (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка інтелектуальної системи підбору взуття з використанням продукційної моделі бази знань*

затверджена наказом ректора НТУ «ДП» від

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів виробничої практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі.*** ***Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***13.05.2022 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПО й тривалості його розробки***	***27.05.2020 р.***

Завдання видав _____ (підпис) *доц. Кабак Л.В.* (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ (підпис) *Мануйленко І.М.* (прізвище, ініціали)

Дата видачі завдання: *14.01.2022 р.*

Термін подання кваліфікаційної роботи до ЕК: *13.06.2020 р.*

РЕФЕРАТ

Пояснювальна записка: с.56, рис.3, дод.2, джерел14.

Об'єкт розробки: мобільний додаток для роботою з взуттям

Мета дипломного проекту: забезпечити зручний додаток для роботи з взуттям використовуючи гарний інтерфейс.

У вступі розглядається аналіз та сучасний стан проблеми, конкретизується мета кваліфікаційної роботи та галузь її застосування, наведено обґрунтування актуальності теми та уточнюється постановка завдання.

У першому розділі проведено аналіз предметної області, визначено актуальність завдання та призначення розробки, розроблена постановка завдання, задані вимоги до програмної реалізації, технологій та програмних засобів.

У другому розділі виконано аналіз існуючих рішень, обрано вибір платформи для розробки, виконано проектування і розробка програми, наведено опис алгоритму і структури функціонування системи, визначені вхідні і вихідні дані, наведені характеристики складу параметрів технічних засобів, описаний виклик та завантаження застосунку, описана робота програми.

В економічному розділі визначено трудомісткість розробленої інформаційної підсистеми, проведений підрахунок вартості роботи по створенню застосунку та розраховано час на його створення.

Актуальність даного програмного забезпечення визначається необхідністю користувачів купувати взуття у доступному вигляді, завдяки простому у користуванні інтерфейсу.

Список ключових слів: JAVA, DART, FLUTTER, REST, МОБІЛЬНИЙ ДОДАТОК.

ABSTRACT

Explanatory note: pages, pics, apps, sources.

Object of development: mobile application for buying shoes.

The purpose of the diploma project: to provide a user-friendly application for reviewing, evaluating, purchasing shoes through a user-friendly interface.

The introduction considers the analysis and the current state of the problem, specifies the purpose of the qualification work and the field of its application, provides a justification for the relevance of the topic and clarifies the problem.

In the first section the analysis of the subject area is carried out, the urgency of the task and purpose of development is defined, the statement of the task is developed, requirements to software realization, technologies and software are set.

The second section analyzes the existing solutions, selects the platform for development, performs design and development of the program, describes the algorithm and structure of the system, determines the input and output data, provides the characteristics of the parameters of hardware, describes the call and application load, describes the program .

In the economic section, the complexity of the developed information subsystem is determined, the cost of work on creating the application is calculated and the time for its creation is calculated.

The relevance of this software is determined by the need for users to buy shoes in an affordable form, thanks to the easy-to-use interface.

List of keywords: MOBILE APP, JAVA, DART, FLUTTER, REST.

ЗМІСТ

СПИСОК УМОВНИХ ПОЗНАЧЕНЬ

AGILE - Agile software development
ЕОМ – електронно-обчислювальна машина.
REST – Representational state transfer
CRUD – create, read, update, delete.

API – інтерфейс програмування додатків.

MVC – Model-View-Controller

ВСТУП

Інтернет – це мережа пристроїв поєднаних між собою. Є мільйони комп’ютерних пристроїв, які під’єднані до цієї мережі постійно або на деяких проміжок часу. Ці пристрої запускають мережеві програми, які спілкуються за допомогою мідних або волоконно-оптичних кабелів, радіо або супутникової передачі.

Ми живемо у такий гарний час, коли ми можемо отримати все просто натиснувшки пошук у браузері.

Якщо казати раніше, то цього не було необхідно робити, вам аби щось купити потрібно було йти в магазин, чи попросити свого знайомого щось вам купити, але зараз один клік і ви купили якійсь товар.

Тому створення мобільного додатку, де буде продаватися продукція, у моєму випадку взуття – це одна з найнеобхідніших речей що потрібно зробити початківцю бізнесмену, аби досягти успіху.

Тому темою дипломного проекту було обрано створення мобільного додатку, у якості інтернет магазину продажу з взуття.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Загальні відомості з предметної галузі

Мобільний додаток, який найчастіше називають додатком, — це тип програмного забезпечення, призначеного для запуску на мобільному пристрої, такому як смартфон або планшетний комп'ютер. Мобільні програми часто служать для надання користувачам послуг, подібних до послуг, доступних на ПК. Програми – це, як правило, невеликі окремі програмні блоки з обмеженою функцією. Таке використання програмного забезпечення було спочатку популяризовано Apple Inc. та її App Store, який пропонує тисячі програм для iPhone, iPad та iPod Touch.

Мобільний додаток також може бути відомий як додаток, веб-додаток, онлайн-додаток, додаток для iPhone або додаток для смартфона.

Мобільні програми – це відхід від інтегрованих програмних систем, які зазвичай зустрічаються на ПК. Натомість кожен додаток надає обмежену та ізольовану функціональність, наприклад гру, калькулятор або мобільний веб-перегляд. Хоча програми, можливо, уникали багатозадачності через обмежені апаратні ресурси ранніх мобільних пристроїв, їх специфічність тепер є частиною їхньої бажаності, оскільки вони дозволяють споживачам самостійно вибирати, що здатні робити їхні пристрої.

Найпростіші мобільні програми беруть програми на базі ПК і портують їх на мобільний пристрій. Оскільки мобільні додатки стають більш надійними, цієї техніки дещо бракує. Більш складний підхід передбачає розробку спеціально для мобільного середовища, використовуючи як його обмеження, так і переваги. Наприклад, програми, які використовують функції на основі місцезнаходження, за своєю суттю створені з нуля з урахуванням мобільних пристроїв, оскільки користувач не прив'язаний до місця розташування, як на ПК.

Програми поділяються на дві великі категорії: нативні та веб-програми. Нативні програми створені для певної мобільної операційної системи, як правило, iOS або Android. Нативні програми мають кращу продуктивність і більш точно налаштований користувальницький інтерфейс (Інтерфейс користувача), і зазвичай перед випуском їм потрібно пройти набагато суворіший процес розробки та забезпечення якості.

Веб-програми використовуються в HTML5 або CSS і вимагають мінімум пам’яті пристрою, оскільки вони запускаються через браузер. Користувач перенаправляється на певну веб-сторінку, а вся інформація зберігається в базі даних на сервері. Для використання веб-програм потрібно стабільне з’єднання.

Наразі доступні кілька типів програм.

Ігрові програми: еквівалент комп’ютерних відеоігор, вони є одними з найпопулярніших типів програм. На них припадає одна третина всіх завантажень додатків і три чверті всіх споживчих витрат.

Програми для підвищення продуктивності: вони зосереджені на підвищенні ефективності бізнесу, полегшуючи виконання різних завдань, таких як надсилання електронних листів, відстеження ходу роботи, бронювання готелів та багато іншого.

Додатки для способу життя та розваг: стають все більш популярними, вони охоплюють багато аспектів особистого способу життя та соціалізації, наприклад знайомства, спілкування в соціальних мережах, а також обмін (і перегляд) відео. Деякі з найбільш відомих програм, таких як Netflix, Facebook або TikTok, потрапляють до цієї категорії.

Інші типи додатків включають додатки для мобільної комерції (M-commerce), які використовуються для покупки товарів в Інтернеті, наприклад

Amazon або eBay, програми для подорожей, які допомагають мандрівникові багатьма способами (бронювання турів і квитків, пошук шляху за допомогою карт і геолокації, щоденники подорожей тощо .), а також допоміжні програми, такі як програми для здоров'я та сканери штрих-кодів.

Frontend розробка відноситься до тієї області веб-розробки, яка зосереджується на тому, що користувачі бачать на своєму кінці. Це включає в себе перетворення коду, створеного розробниками бекенда, в графічний інтерфейс, забезпечуючи представлення даних у легкому для читання та зрозумілому форматі.

Без розробки інтерфейсу все, що ви побачите на веб-сайті чи веб-додатку, — це нерозшифровані коди (якщо ви теж не розробник, звичайно). Але завдяки розробникам інтерфейсу люди, які не мають досвіду програмування, можуть легко розуміти та використовувати веб-додатки та веб-сайти. Все, що ви бачите, коли ви відвідуєте Google Apps, Canva, Facebook та інші веб-програми, є продуктами спільної роботи розробників backend і frontend.

Веб-технології, залучені до розробки фронтенду

Розробники Frontend використовують кілька веб-технологій для перетворення закодованих даних у зручні для користувача інтерфейси. Серед них мова розмітки гіпертексту (HTML), каскадні таблиці стилів (CSS) і JavaScript. Нижче наведено короткий опис трьох технологій, з якими повинні бути знайомі розробники інтерфейсу.

1. HTML

HTML є будівельним блоком веб-сайтів. Це мова програмування, яка використовується для опису та позначення вмісту, тому браузер відображає його правильно. Наприклад, зображення в дописі в блозі відображатиметься як <img> у HTML-коді, тож браузери будуть знати, що їм потрібно відобразити зображення.

2. CSS

CSS більше схожий на набір інструкцій, які контролюють стиль і структуру веб-сторінки, ніж на мову програмування. Це допомагає розробникам керувати форматуванням, презентацією та макетом веб-сайту або веб-додатка. У той час як HTML визначає елементи на сторінці, CSS визначає, як користувачі бачать вміст. Наприклад, він контролює розмір, межі та вирівнювання зображення в дописі в блозі.

3. JavaScript

Розробники Frontend вже можуть створювати веб-сайти за допомогою HTML і CSS. Насправді JavaScript з'явився лише в 1995 році. Однак зараз важко уявити веб-сайти без JavaScript, оскільки він дозволяє розробникам робити сайти інтерактивними. Мова програмування може змінювати вміст веб-сайту на основі дій користувача. Наприклад, щотижневе опитування Techslang було створено за допомогою JavaScript. Якщо вибрати відповідь і натиснути «Голосувати», відобразиться загальна кількість голосів за кожен варіант.

Скільки заробляють Frontend-розробники?

Розробка фронтенду — завдання не з легких, тому фронтенд-розробники є одними з найбільш високооплачуваних в IT-індустрії. Старший розробник інтерфейсу може заробляти до 120 000 доларів США на рік. На відміну від цього, доступ до розробників середнього рівня може заробляти від 49 000 до 99 000 доларів США на рік.

Бекенд:

Бекенд виконується на сервері(зазвичай, сервер знаходиться віддалено від фронтової частини, хоча це не обов'язково, і може бути на одному пристрої) .

Сервер в основній своїй массі потрібен для або додавання динамічності до вашого додатку, або, як це є зазвичай для зберігання данних, та взаємодії з фронтовою(клієнтською) стороною, а не з самим користувачем, тобто він напряму не зв'язується з фронтовою частиною, але без нього було б не можливі багато речей.

На бекенді(серверній стороні) зазвичай пишуть API, наприклад ви можете написати серверну частину на мові програмування Java, яке буде надавати вам дані щодо вашого аккаунта на якоїсь платформі, тобто ви спочатку будете викликати API написаний вами, якій буде далі викликати API якоїсь платформи, і надавати вам якісь данні.

Back-end Development відноситься до розробки на стороні сервера. Він фокусується на базах даних, сценаріях, архітектурі веб-сайту. Він містить дії за кадром, які відбуваються під час виконання будь-якої дії на веб-сайті. Це може бути вхід в обліковий запис або покупка в інтернет-магазині. Код, написаний розробниками серверної частини, допомагає браузерам спілкуватися з інформацією бази даних.

Java: відома як король усіх мов програмування, Java використовується для розробки популярних веб-сайтів і веб-додатків, включаючи Netflix, Uber, Google Earth і Tinder.

Ruby on Rails (RoR): RoR став улюбленим серед програмістів, оскільки він робить процес веб-розробки легким. Розробники використовували цю мову для створення кількох авторитетних веб-сайтів і програм, таких як Airbnb, Slideshare, Goodreads, Groupon, Askfm і Kickstarter.

Python: Python є однією з найбільш часто використовуваних мов програмування у світі. Окрім розробки веб-сайтів, він також бере участь у мережевому програмуванні, штучному інтелекті (AI) та машинному навчанні (ML). Серед веб-програм, які використовують Python, є Spotify, Dropbox, Google, Reddit та Instagram.

PHP: Перевага PHP як мови програмування бекенда полягає в тому, що його легко вивчити. Він також підтримує різні системи керування вмістом (CMS), такі як WordPress і Joomla. PHP також використовувався для розробки веб-додатків, включаючи Facebook, Wikipedia, Tumblr, MailChimp і Flickr.

1.2 Призначення розробки та область застосування.

Область нічим не обмежується, а тому є універсальною.

1.3 Підстава для розробки

Підставою для розробки кваліфікаційної роботи на тему “Розробка інтелектуальної системи підбору взуття з використанням продукційної моделі бази знань ” є наказ по Національному технічному університету “Дніпровська політехніка” від __.__. 2022р. №_____-л.

1.4 Постановка завдання

Ціллю є створити легкий у використанні інтерфейс, аби можна було без усіляких зусиль знайти щось для вас:
А це – шукати по типу, кольору, та моделі, орієнтуючись на конкретні компоненти взуття

1.5. Вимоги до програми або програмного виробу.

1.5.1. Вимоги до функціональних характеристик

Вимоги до клієнта це мати мобільний пристрій який буде підтримувати мінімум андроїд версії 10, та мати можливість запускати якісь додатки середньої важкості.

1.5.2 Вимоги до інформаційної безпеки

Оскільки це дані зберігаються у клієнта, ніякий захист окрім валідації не буде використовуватися.

1.5.3 Вимоги до складу та параметрів технічних засобів

Мобільний пристрій повинен мати такі характеристики:

Qualcomm Snapdragon 450

- Не менше 1 GB оперативної пам’яті
- 653 МБ вільного місця на телефоні;

1.5.4 Вимоги до інформаційної та програмної сумісності

Аби додаток вірно працював андроїд система повинна бути версії 10 та вище, та мати гарне апаратне забезпечення.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ТА РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Функціональне призначення системи

Задумом цієї дипломної роботи є розробка мобільного додатку, який представляє собою магазин з продажу взуття різних видів, який включає в себе основні характеристики та поведніки магазину.

Можливості додатку:

- Створення моделей взуття та занесення їх у систему
- Фільтрація моделей взуття за обраними критеріями
- Виставлення оцінки моделі взуття та залишення коментаря

2.2. Опис застосованих математичних методів

У якосіті алгоритму використовується лінійна кореляція на основі попередніх даних.

2.3. Опис використаних технологій та мов програмування

Мобільний додаток має бути реалізован на двох мовах програмування.

Для обробки запититів клієнту та зв’язку з базою даних використовується Java 8 ревізії з допомогою фреймворку Spring-Boot, включно з модулями Spring-Boot-Data-Jpa, Spring-Boot-Web, Spring-Boot-Security, в якості ORM-фреймворку використувєтся одна з найпоширеніших реалізацій JPA, а саме Hibernate. Для створення UI використовується мова програмування Dart з допомогою фреймворку Flutter.

Опис мови програмування Java

Java — це мова програмування високого рівня загального призначення, вперше випущена Sun Microsystems у 1995 році. Вона розроблена таким чином, щоб мати якомога менше залежностей від реалізації, безкоштовна у використанні та може працювати на всіх платформах. Він є одночасним, заснований на класі та об’єктно-орієнтований. Простіше кажучи, Java — це обчислювальна платформа, на якій користувачі можуть розробляти програми.

Java подібна до C++, але спрощена, щоб усунути особливості мови, які викликають типові помилки програмування. Файли вихідного коду, тобто файли з розширенням .java, компілюються у формат, відомий як байт-код, який є файлами з розширенням .class. Потім це може бути виконано інтерпретатором Java. Байт-код може бути безпосередньо перетворений в інструкції на машинній мові компілятором «точно вчасно».

КОМПОНЕНТІ JAVA

Мова програмування Java складається з трьох основних компонентів:

Віртуальна машина Java (JVM): JVM — це механізм, який забезпечує середовище виконання для керування кодом Java або програмами. Він є центром мови програмування і виконує операцію перетворення байт-коду Java на машинну мову. Він надає численні бібліотеки, фреймворки та інструменти.

Середовище виконання Java (JRE): JRE — це середовище виконання, яке необхідне для виконання програм і додатків Java. Якщо користувач хоче запустити програму Java на своїй машині, він повинен мати JRE, встановлений на машині. Це залежить від платформи, тобто встановлений JRE має бути сумісним з операційною системою та архітектурою користувача.

Java Development Kit (JDK): JDK є основним компонентом середовища Java. Він містить JRE разом з компілятором Java, налагоджувачем Java та іншими класами. Він використовується для розробки Java, щоб забезпечити повні виконувані файли та двійкові файли, а також інструменти для компіляції та налагодження програми Java.

ПЕРЕВАГИ JAVA

Java легко навчитися. Ця мова не вимагає попереднього знання базової мови програмування. На відміну від інших мов програмування, включаючи C++, під час компіляції Java вона не компілюється в машину для певної платформи. Це означає, що програму, складену на одній машині, можна легко виконати на будь-якій іншій машині, не вносячи жодних змін.

Java є багатопоточною, що означає, що кілька завдань можна виконувати одночасно, а користувачі можуть створювати інтерактивні програми, які працюють безперебійно. Завдяки його безпечним функціям можна розробляти

системи без вірусів і несанкціонованого доступу. Методи аутентифікації засновані на шифруванні з відкритим ключем.

Опис Java фреймворку Spring

Spring Framework — це фреймворк додатків платформи Java та інверсія контрольного контейнера.

Що таке Spring?

Spring Framework — це фреймворк додатків платформи Java та інверсія контрольного контейнера. Будь-яка програма Java може використовувати основну функціональність фреймворка, однак є вдосконалення для створення веб-додатків поверх платформи Java EE (Enterprise Edition). Фреймворк набув популярності серед Java-спільноти як розширення архітектури Enterprise JavaBeans (EJB), незважаючи на те, що він не передбачає певної моделі програмування. Spring Framework — це безкоштовний фреймворк з відкритим кодом, який спочатку був випущений у 2002 році.

Spring — це простий каркас з великою кількістю функцій. Його іноді називають фреймворком фреймворків, оскільки він підтримує різноманітні фреймворки, включаючи Struts, Hibernate, Tapestry, EJB, JSF та інші. У більш широкому розумінні каркас можна розглядати як структуру, в якій ми знаходимо рішення різноманітних технічних труднощів.

У Spring Framework є різні модулі, які надають різноманітні послуги.

Spring Core Container є основним модулем Spring, який пропонує контейнери Spring (BeanFactory і ApplicationContext).

Аспектно-орієнтоване програмування (AOP) дозволяє реалізувати наскрізні проблеми.

Модуль Spring Roo забезпечує швидке рішення для розробки додатків для корпоративних додатків на основі Spring, зосереджуючись на конвенції, а не на конфігурації.

Доступ до даних: використання Java Database Connectivity (JDBC) та об’єктно-реляційного відображення для роботи з реляційними системами керування базами даних на платформі Java.

Обмін повідомленнями: реєстрація об’єкта прослуховування повідомлень, що налаштовується, для прозорого споживання повідомлень із черг повідомлень через Java Message Service (JMS) та покращеного надсилання повідомлень через стандартні API JMS

Модель–представлення–контролер (MVC) — це фреймворк на основі HTTP та сервлетів для онлайн-додатків і веб-сервісів RESTful (передача стану репрезентації), які надають гачки для розширення та налаштування.

Spring Framework є найбільш широко використовуваним фреймворком для розробки додатків Java. Фундаментальною особливістю Spring Framework є ін’єкція залежностей, також відома як інверсія керування (IoC). Ми можемо створити слабо пов’язану програму з Spring Framework. Якщо тип або атрибути програми чітко визначені, бажано використовувати їх.

Spring Boot — це модуль Spring Framework. Це дозволяє нам створювати автономну програму з мінімальною конфігурацією або без неї. Якщо ми хочемо створити просту програму на основі Spring або сервіс RESTful, нам слід скористатися ним.

Ми можемо порівняти і пружинний, і пружинний черевик з наступними характеристиками.

Spring Framework — це широко використовуваний фреймворк Java EE для створення додатків. але Spring Boot використовується для розробки REST API.

Spring прагне спростити розробку Java EE, а Spring Boot — скоротити довжину коду та забезпечити найпростіший спосіб розробки веб-додатків.

Основною особливістю Spring Boot є автоналаштування. Він автоматично налаштовує класи відповідно до вимог. але основною особливістю spring є ін'єкція залежностей.

Spring допомагає спростити роботу, дозволяючи розробляти слабо пов’язані програми, а Spring boot допомагає створювати окремі програми з меншою кількістю конфігурацій.

Порівнюючи процес тестування, для тестування програми spring нам потрібно явно налаштувати сервер. Але Spring boot пропонує вбудовані сервери, такі як Jetty і Tomcat.

Інверсія контролю (IOC) та ін'єкція залежності

Це шаблони проектування, які використовуються в програмному коді для видалення залежностей. Вони дозволяють легко тестувати та підтримувати код.

Опис Java ORM-фреймфорку Hibernate

Hibernate — це інструмент ORM на основі Java, який забезпечує структуру для відображення об’єктів домену програми з таблицями реляційної бази даних і навпаки.

Hibernate, мабуть, найпопулярніша реалізація JPA і одна з найпопулярніших фреймворків Java в цілому. Hibernate діє як додатковий рівень поверх JDBC і дозволяє вам реалізувати незалежний від бази даних рівень збереження. Він забезпечує реалізацію об’єктно-реляційного відображення, яка відображає ваші записи бази даних з об’єктами Java і генерує необхідні оператори SQL для реплікації всіх операцій у базу даних.

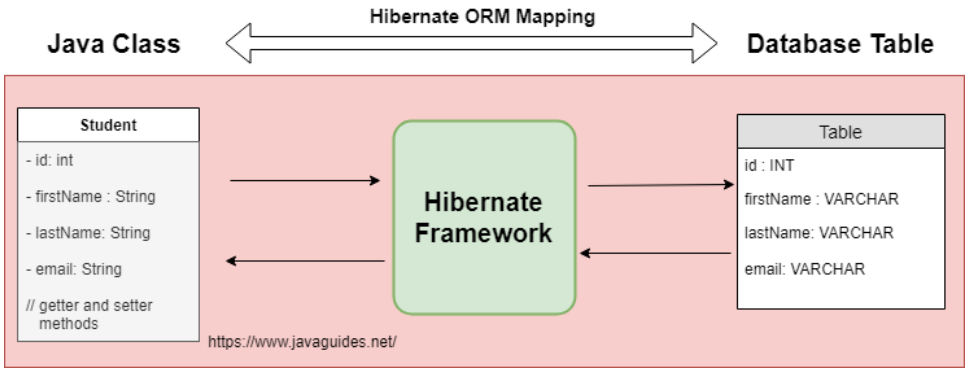


Рис. 2.1. Схема роботи Hibernate

Hibernate знаходиться між традиційними об’єктами Java і сервером бази даних, щоб виконувати всі роботи щодо збереження цих об’єктів на основі відповідних механізмів і шаблонів O/R.

Опис мови програмування Dart

Dart — це мова програмування загального призначення з відкритим кодом, розроблена Google. Він підтримує розробку додатків як на стороні клієнта, так і на стороні сервера. Але він широко використовується для розробки додатків для Android, iOS, IoT (Internet of Things) та веб-додатків за допомогою Flutter Framework.

Liquibase

Liquibase — бібліотека з відкритим кодом для відстеження змін у базі даних.

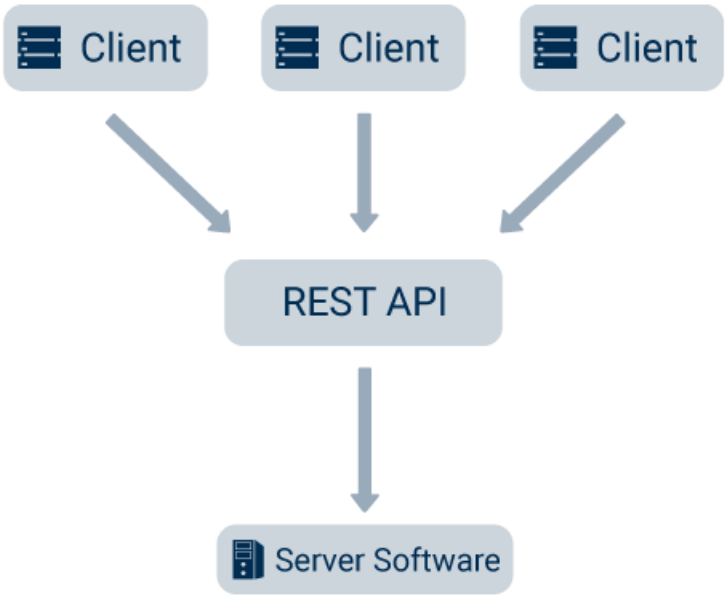
Що це означає? Ми всі працювали в проектах, де типовим процесом управління змінами в базі даних було написання DDL, DML-скриптів у sql-файлах, які контролювали джерело, а зміни в базі даних виконувались самостійно або якоюсь іншою спеціальною командою. Але не було дійсно гарного способу визначити послідовність цих змін, його дотримувались як найкращу практику в проекті. Нам завжди доводилося писати sql для відкату, а міграція до іншої бази даних була болісним завданням.

Liquibase намагається вирішити всі ці проблеми елегантним способом. Скрипти Liquibase зазвичай пишуться у форматі xml (можливо, тому, що xml є більш читабельним), хоча також підтримуються інші формати, такі як json та yaml. Зміни бази даних визначаються у файлах журналу змін як невеликі набори змін, які однозначно ідентифікуються за назвою набору змін та автором. Послідовність визначається у файлі головного журналу змін. Відкати в більшості випадків виконує сама Liquibase. І оскільки ці сценарії не залежать від бази даних, міграції до інших баз даних відбуваються відносно гладко. На даний момент Liquibase підтримує більшість популярних баз даних, включаючи MySQL, Oracle, DB2 і PostgreSQL.

REST API

Ідея «репрезентаційної передачі стану» (REST) складається з архітектури програмного забезпечення для обміну даними між програмними системами (клієнт-сервер). Особливістю порівняно з іншими архітектурами є дуже проста реалізація, концентрація на ресурсах та відсутність громадянства.

Уся комунікація через REST фокусується на ресурсі, а не на дії. Ресурс завжди адресується однозначно, і тоді можна виконувати основні дії зі створення, читання, зміни чи видалення. У цьому контексті термін без стану означає, що всі описові параметри обмінюються між клієнтом і сервером. Перевага тут у тому, що немає сесій. Без сеансів клієнтські та серверні системи можна вільно масштабувати, оскільки кожен виклик REST завершений і зрозумілий сам по собі, без попередніх або наступних.



API (інтерфейс прикладного програмування) — це відповідна реалізація архітектури REST конкретної системи. Потім його називають RESTful API. Ресурси, до яких можна звертатися через REST, визначаються в інтерфейсі.

Для відповідних ресурсів існують пов’язані параметри, які описують ресурс і змінюються за допомогою REST.

Для RESTful API системи постачальник зазвичай надає документацію API, в якій можна переглянути всі ресурси та їх параметри.

Прикладом хорошої документації API є інтерфейс REST нашої системи квитків Target Process, який можна переглянути тут: Target Process REST API.

У виробничому середовищі система з підтримкою REST може надавати такі ресурси, наприклад:

/виробництво/виробниче замовлення

Параметри: ідентифікатор, номер замовлення, кількість, продукт, термін, …

/production/production-order/{id}/component.

Параметри: ID, номер матеріалу, кількість, партія, …

2.4. Опис структури системи та алгоритмів її функціонування

Терміни «Ін’єкція залежності» (DI) та «Інверсія управління» (IoC) зазвичай використовуються як синоніми для опису одного і того ж шаблону проектування. Тому деякі люди кажуть IoC Container, а деякі люди кажуть контейнер DI, але обидва терміни вказують на те саме. Тому нехай вас термінологія не бентежить.

Контейнер IoC, який також відомий як DI Container, є основою для дуже ефективно впровадження автоматичної ін’єкції залежностей. Він керує повним створенням об’єкта та його терміном життя, а також впроваджує залежності в класи.

Щоб отримати додаткову інформацію про DI та IoC, див. Розуміння інверсії керування, ін’єкції залежностей та локатора служби та ін’єкції залежностей у ASP.NET MVC за допомогою контейнера Unity IoC.

Що таке контейнер DI
Контейнер DI — це фреймворк для створення залежностей і автоматичного введення їх, коли це потрібно. Він автоматично створює об’єкти на основі запиту та вводить їх, коли потрібно. DI Container допомагає нам керувати залежностями в програмі простим і легким способом.

Контейнер DI створює об’єкт визначеного класу, а також впроваджує всі необхідні залежності як об’єкт, конструктор, властивість або метод, який запускається під час виконання та утилізується у відповідний момент. Цей процес завершено, тому нам не доводиться постійно створювати об’єкти й керувати ними вручну.

Ми також можемо керувати залежностями додатків без контейнера DI, але це буде як DI POOR MAN’S, і нам потрібно зробити більше роботи, щоб зробити його налаштованим і керованим.

Життєвий цикл контейнерів DI

Усі контейнери DI повинні проходити через різні набори статусів, поки не буде зроблено остаточний висновок, який включає наступні етапи життєвого циклу.

Реєстрація: контейнер слід зареєструвати або ініціювати, як тільки він отримає певний тип, і це називається етапом реєстрації
Вирішити: під час використання контейнерів DI нам не потрібно створювати об’єкти вручну. Контейнер робить це за нас. Це називається фазою розв’язання. Контейнер DI повинен включати деякі з методів для вирішення зазначеного типу, як тільки він ідентифікує. Відповідний контейнер створює об’єкт зазначеного типу разом із введенням необхідних залежностей, якщо будь-яка з них потрібна, і, нарешті, повертає об’єкт
Утилізувати: контейнери DI повинні мати можливість відповідно керувати часом життя залежних об'єктів. Більшість контейнерів DI включають різних менеджерів часу життя та їхню відповідальність за управління життєвим циклом об’єкта та його утилізацію.
Список популярних контейнерів DI для .Net
Сьогодні існує багато чудових контейнерів DI, доступних для .NET. Тут я ділюся списком найкорисніших контейнерів DI для фреймворку .Net.

2.5. Обгрунтування та організація вхідних та вихідних даних програми

Вхідні дані поступають за допомогою взаємодії користувача з елементами програми, а вихідні дані генерує програма(backend – частина). Тобто вихідні дані йдуть на вихід у вигляді компонентів.

2.6. Опис роботи розробленої системи

2.6.1. Використані технічні засоби

Оскільки вся інформація зберігається локально(на стороні клієнта) то тоді ось рекомендації щодо пристрою для використання цього додатку:

- 4 ГБ оперативної пам’яті.
- Вільного місця на диску в розмірі 800 МБ пам’яті.
- Рідкокристалічний монітор з діагоналлю не менше 17 ".
- Qualcomm Snapdragon 450 з частотою 1800 МГЦ

Вище наведені характеристики являють собою рекомендовані. Це означає, що при наявності характеристик не нижче зазначених, розроблений додаток буде функціонувати відповідно до вимог щодо надійності, безпеки та швидкості обробки даних.

2.6.2. Використані програмні засоби

Фронтова частина(візуальн) використовує Flutter, а саме його фреймворк – Dart, бекенд частина використовує Java, а саме його фреймворк – Spring.

Необхідні програмні засоби на стороні клієнта:

- Java 8
- Spring 2.5
- Hibernate 5.6

- Android SDK 30
- Liquibase
- Dart 2.17
- Flutter 3.0.2
- Операційна система Windows 7/10.
- SDK Manager

2.6.3. Виклик та завантаження програми

Для запуску програми необхідно окремо запустити файл з розширенням «*dat», це дартівський файл, та application.java для запуску бекенд(серверної частини)

2.6.4. Опис інтерфейсу користувача

Після запуску додатка, ми попадемо на сторінку входу до акканту, де треба ввести логін та пароль.

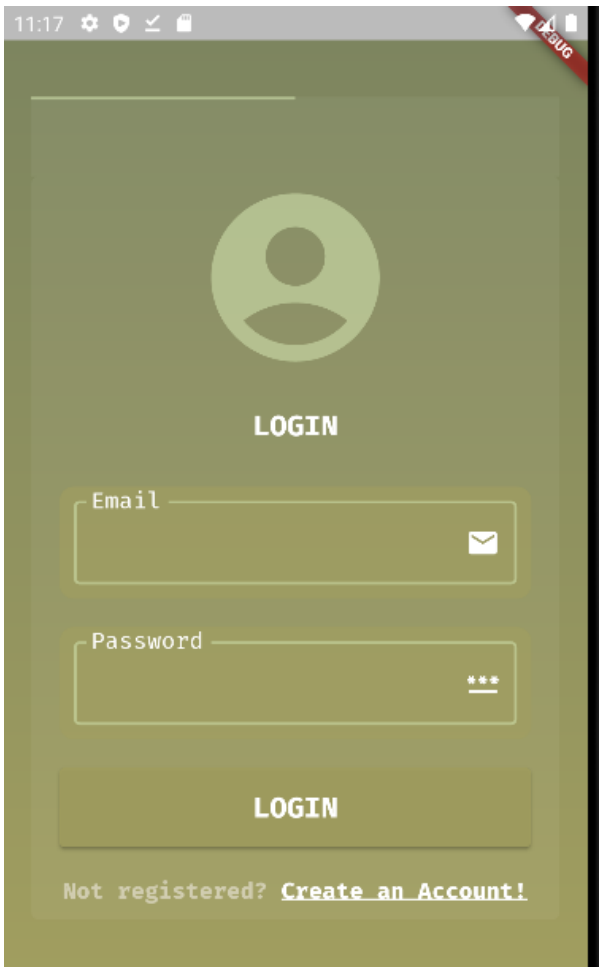


Рис. 2.3. Сторінка входу

Якщо треба зареєструватися, треба натиснути «Create an Account»

Рис. 2.4. Сторінка реєстрації

Рис. 2.5. Сторінка реєстрації продовження

Розглянемо послідовно як слід використовувати мобільний-додаток.

Після того, як ми зайдемо до нашого додатку, ми побачимо головну сторінку

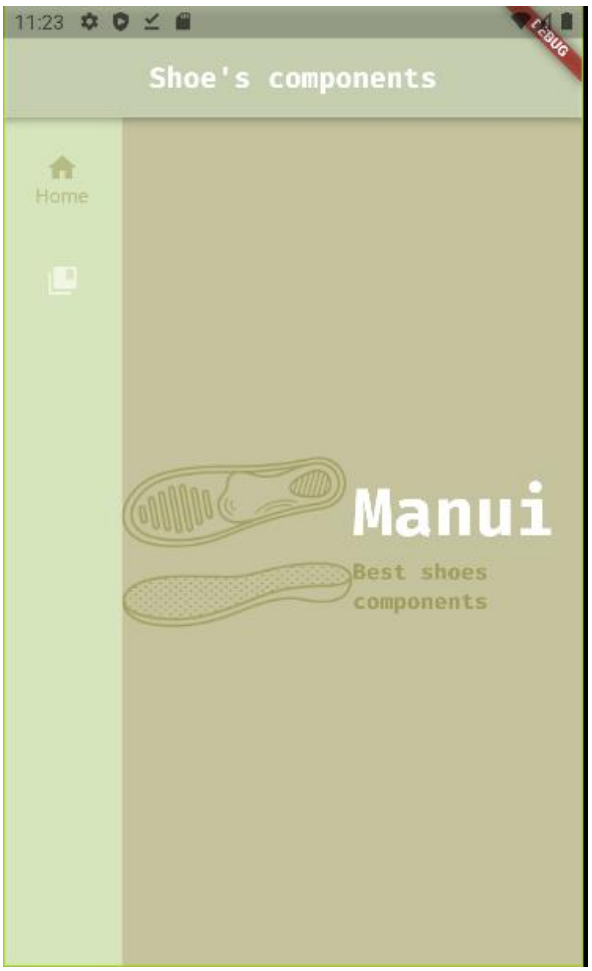


Рис. 2.5. Головна сторінка

Якщо ми натиснимо на кнопку «Categories» або свайпнимо уліво, то ми побачимо нове меню

Рис. 2.6. Сторінка категорій продуктів

Ми того, як оберемо, ми моживо побачити більш детальний вибір

Рис. 2.7. Типи продукції

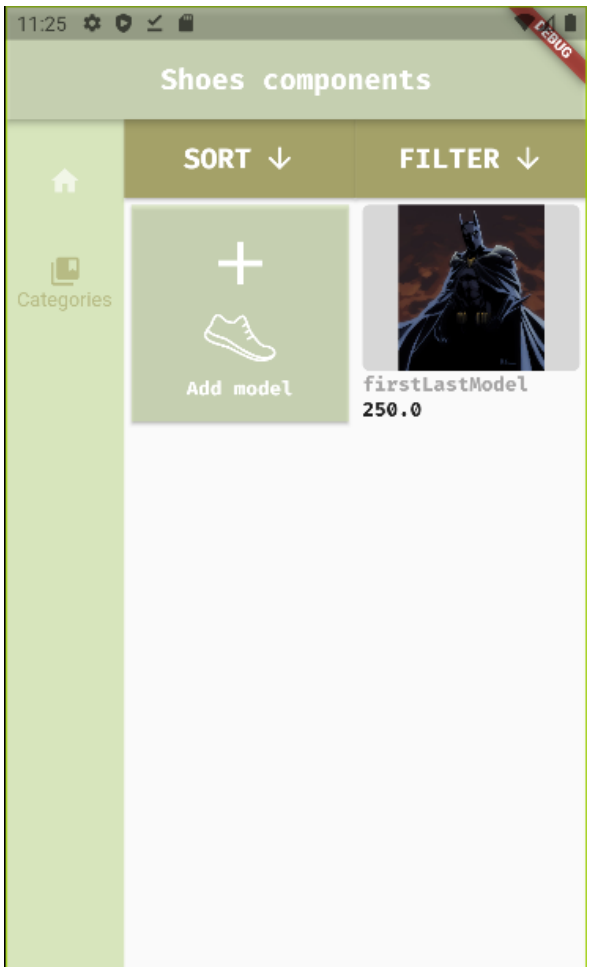


Рис. 2.8. Список взуття

Аби відфільтрувати, треба натиснути кнопку «Filter», і з’явиться можливість фільтркувати за різними критеріями

Рис. 2.10. Після натискання на фільтр

Після того, як ви натиснули на фільтр, ви повинні натиснути «View items», та подивитися відфільтровані дані.

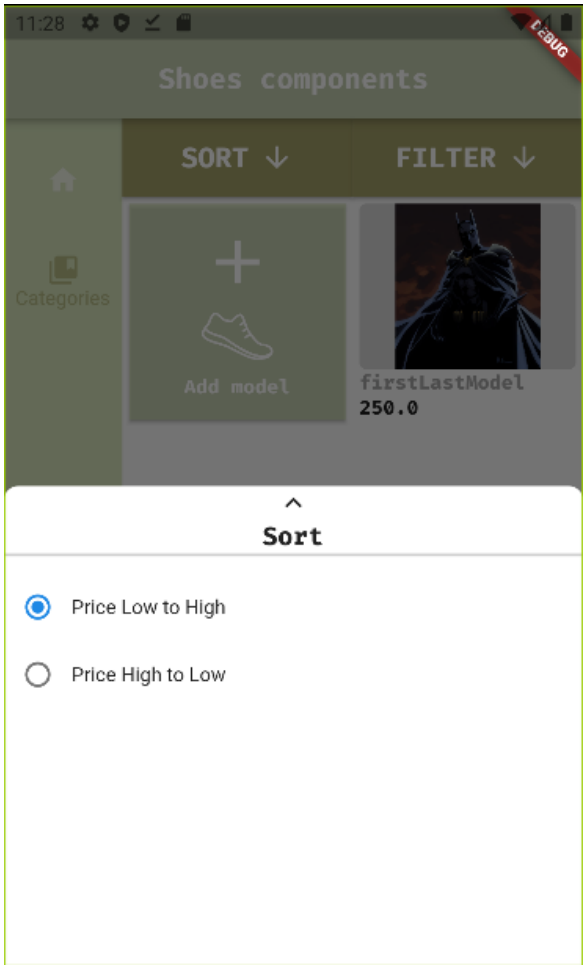


Рис. 2.11. Після натискання кнопки SORT

Задля відсортування треба натиснути кнопку SORT, та з’явиться вибір

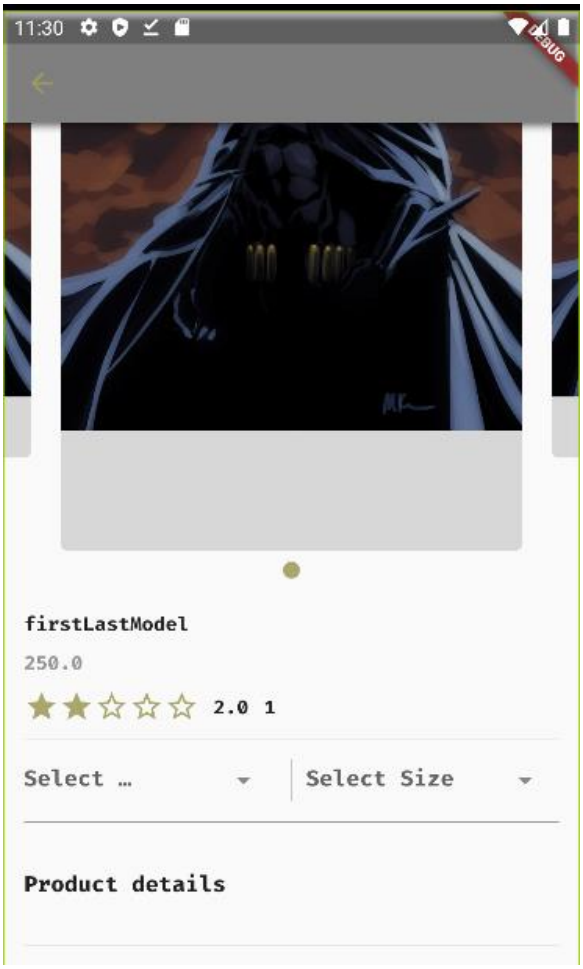


Рис. 2.13. Більш детальна інформація

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Визначення трудомісткості розробки програмного забезпечення

Початкові дані:

1. q – передбачуване число операторів програми – 1300;
2. C – коефіцієнт складності програми – 1,5;
3. p – коефіцієнт корекції програми в ході її розробки – 0,07;
4. B – коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі – 1,3;
5. $C_{ПР}$ – середня годинна заробітна плата програміста – 166 грн/год;
6. $C_{МЧ}$ – вартість машино-години ЕОМ – 15 грн/год.
7. k – коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності – 0,8;
8. – число виконавців – 1;
9. F_p – місячний фонд робочого часу (при 40 годинному робочому тижні F_p = 176 годин).

Враховуючи середню платню програміста в України, де дані були взяті з порталу «(Української спільноти програмістів - DOU)». Місячна платня в середньому становить 1000, у програміста який має досвід мовою користуванню Java менше аніж рік, Курс НБУ у червня 2022 року показував, що один американський доллар дорівнює десь 29,25 грн, якщо перекладати у гривні це становитиме 29,25 грн. Ураховуючи восьмигодинний робочий день, середня заробітна плата за годину становитиме 166 грн.

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки ПЗ розраховується за формулою:

$$t = t_o + t_u + t_a + t_n + t_{отл} + t_д \text{, людино-годин} \qquad (3.1)$$

де t_o – витрати праці на підготовку й опис поставленої задачі (приймається 50);

t_u – витрати праці на дослідження алгоритму рішення задачі,

t_a – витрати праці на розробку блок-схеми алгоритму,

t_n – витрати праці на програмування по готовій блок-схемі,

$t_{отл}$ – витрати праці на налагодження програми на ЕОМ,

$t_д$ – витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у ПЗ, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q \cdot C \cdot (1 + p), \tag{3.2}$$

q – передбачуване число операторів,

C – коефіцієнт складності програми,

p – коефіцієнт корекції програми в ході її розробки.

$$Q = 1000 \cdot 1{,}5 \cdot (1 + 0{,}07) = 2086{,}5$$

Витрати праці на вивчення опису задачі t_u визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{Q \times B}{(75..85) \times k}, \text{ людино-годин.} \tag{3.3}$$

де В – коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі.

k – коефіцієнт кваліфікації програміста, обумовлений стажем роботи з даної спеціальності.

Витрати праці на розробки алгоритму рішення задачі:

$$t_a = \frac{Q}{(20..25) \times k}$$, людино-годин. (3.4)

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20..25) \times k}$$, людино-годин. (3.5)

Витрати праці на налагодження програми на ЕОМ:

- за умови автономного налагодження одного завдання:

$$t_{\text{отл}} = \frac{Q}{(4..5) \times \text{k}}$$, людино-годин. (3.6)

- за умови комплексного налагодження завдання:

$$t^{к}_{отл} = 1,4 \times t_{отл}$$, людино-годин. (3.7)

$$t^{к}_{отл} = 1,4 \times 551 = 923,4 \text{ людино-годин.}$$

Витрати праці на підготовку документації:

$$t_д = t_{др} + t_{до}$$, людино-годин, (3.8)

де $t_{\text{др}}$– трудомісткість підготовки матеріалів і рукопису

$$t_{др} = \frac{Q}{(15..20) \times k}$$, людино-годин.

(3.9)

$$t_{др} = \frac{2086}{16 \times 0,8} = 152 \text{ людино-годин,}$$

де $t_{\text{др}}$– трудомісткість редагування, печатки й оформлення документації

$$t_{до} = 0{,}75 \times t_{др}\text{ , людино-годин.} \quad (3.10)$$

$$t_{до} = 0{,}75 \times 152 = 124 \text{ людино-годин.}$$

$$t_{д} = 152 + 124 = 176 \text{ людино-годин.}$$

Підставляючи відповідні значення до формули (3.1):

$$t = 50 + 40{,}8 + 251{,}3 + 122 + 721 + 241 = 1634 \text{ людино-годин.}$$

Тобто ми розрахували, що виявилося необхідним 1267,7 людино-годин для розробки даного програмного забезпечення.

3.2. Розрахунок витрат на створення програми

Витрати на створення ПЗ $К_{ПО}$ включають витрати на заробітну плату виконавця програми $З_{ЗП}$ і витрат машинного часу, необхідного для налагодження програми на ЕОМ.

$$К_{ПО} = З_{ЗП} + З_{МВ}\text{ , грн,} \quad (3.11)$$

де $З_{ЗП}$ – заробітна плата виконавців, яка визначається за формулою:

$$З_{ЗП} = t \times С_{ПР}\text{ , грн,} \quad (3.12)$$

де t – загальна трудомісткість, людино-годин,

$С_{ПР}$ - середня годинна заробітна плата програміста, грн/година.

Підставляючи відповідні значення до формули отримуємо:

$$З_{ЗП} = 1070 \times 199 = 212930 \text{ грн.}$$

$З_{МВ}$ - вартість машинного часу, необхідного для налагодження програми на ЕОМ:

$$З_{МВ} = t_{отл} \times С_{МЧ}\text{ , грн,} \quad (3.13)$$

$t_{\text{отл}}$ – трудомісткість налагодження програми на ЕОМ, год.

$C_{\text{Мч}}$ – вартість машино-години ЕОМ, грн/год.

$$З_{МВ} = 551 \times 14 = 7714 \text{ грн.}$$

$$К_{ПО} = 212930 + 7714 = 220644 \text{ грн.}$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \times F_p}, \text{ міс,} \qquad (3.14)$$

де B_k – число виконавців,

F_p – місячний фонд робочого часу (при 40 годинному робочому тижні Fp = 176 годин).

$$\text{T} = \frac{2086}{1 \times 176} \approx 9{,}073 \text{ міс.}$$

Висновок:

Тобто очікувані витрати будуть становити 250456 гривень, та складатимуть 9,4 місяці.

ВИСНОВКИ

Під час виконання дипломного проекту був реалізований мобільний додаток, який має на меті продаж взуття, за допомогою компонентів взуття.

Власне проект був реалізований завдяки мови програмування JAVA, та інших фреймворків , такі як SPRING та HIBERNATE у JAVA та Dart у Flutter.

Вони відповідно були використані для бекенду та фронтенду відповідно.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Туторіали Фреймворку Spring / URL: https://spring.io/guides#tutorials (дата звернення 28.04.2022).
2. Огляд Dart / URL: https://dart.dev/overview (дата звернення 29.04.2020).
3. Система типів в Dart / URL: https://dart.dev/guides/language/type-system (дата звернення 30.04.2022).
4. Концепці Liquibase / URL: https://docs.liquibase.com/concepts/home.html (дата звернення 30.04.2022).
5. Вступ до JSON Web Tokens / URL: https://jwt.io/introduction (дата звернення 30.04.2022).
6. Документація MySql / URL: https://dev.mysql.com/doc/ (дата звернення 30.04.2022).
7. Реляційні бази даних / URL: https://en.wikipedia.org/wiki/Relational_database (дата звернення 02.05.2022).
8. Багато до багатьох зв’язкок в JPA/ URL: https://www.baeldung.com/jpa-many-to-many (дата звернення 02.05.2022).
9. Робота з JSON / URL: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/JSON (дата звернення 03.05.2022).
10. Вступ до XML / URL: https://developer.mozilla.org/en-US/docs/Web/XML/XML_introduction (дата звернення 03.05.2022).
11. Apache Maven 3.x – дослідження / URL: https://maven.apache.org/ref/3.8.6/ (дата звернення 03.05.2022).
12. Зберігання об’єктів сутності JPA / URL: https://www.objectdb.com/java/jpa/persistence/store (дата звернення 04.05.2020).
13. Документація Фреймворку Spring / URL: https://docs.spring.io/spring-framework/docs/current/reference/html/ (дата звернення 20.06.2022).
14. Гайди Фреймворку Spring / URL: https://spring.io/guides (дата звернення 15.06.2022).

ДОДАТОК А

КОД ПРОГРАМИ

```
package com.example.projectbe.core.mapper;

import com.example.projectbe.core.dto.ShoesModelTypeCreationDto;
import com.example.projectbe.core.dto.ShoesModelTypeDto;
import com.example.projectbe.core.util.ImageUtil;
import com.example.projectbe.domain.entity.ShoesModelType;
import org.springframework.stereotype.Component;

@Component
public class ShoesModelTypeMapper {

   public ShoesModelTypeDto toShoesModelTypeDto(ShoesModelType
shoesModelType) {
      ShoesModelTypeDto shoesModelTypeDto = new ShoesModelTypeDto();

shoesModelTypeDto.setTypeName(shoesModelType.getModelType().getUiRepresen
tation());

shoesModelTypeDto.setImage(ImageUtil.base64ImageFromFilePath(shoesModelTy
pe.getPath()));
      return shoesModelTypeDto;
   }

   public ShoesModelType
fromShoesModelTypeCreationDto(ShoesModelTypeCreationDto
shoesModelTypeCreationDto) {
      ShoesModelType shoesModelType = new ShoesModelType();
```

```
        shoesModelType.setId(shoesModelTypeCreationDto.getId());
        return shoesModelType;
    }
    public ShoesModelTypeCreationDto
toShoesModelTypeCreationDto(ShoesModelType ShoesModelType) {
        ShoesModelTypeCreationDto shoesModelTypeCreationDto = new
ShoesModelTypeCreationDto();
        shoesModelTypeCreationDto.setId(ShoesModelType.getId());

shoesModelTypeCreationDto.setTypeName(ShoesModelType.getModelType().getUi
Representation());
        return shoesModelTypeCreationDto;
    }
}

package com.example.projectbe.core.constant;

import java.time.format.DateTimeFormatter;

public final class CoreConstants {

    public static final DateTimeFormatter FORMATTER =
DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm");

    public static final String pathToProjectData = "D:\\projectDate\\";

    public static final Integer IMAGE_RESIZE_HEIGHT = 300;
    public static final Integer IMAGE_RESIZE_WIDTH = 300;
    public static final Float similarityEdge = 50.0F;
}
package com.example.projectbe.core.dto.generic;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.util.List;

@NoArgsConstructor
@AllArgsConstructor
@Data
```

```
public class ListResponseDto<T> {

    private List<T> elements;

}
package com.example.projectbe.core.dto;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.time.LocalDate;

@NoArgsConstructor
@AllArgsConstructor
@Data
public class BuyerRegistrationDto {

    private String email;

    private String firstName;

    private String lastName;

    private String password;

    private LocalDate birthDate;


}
package com.example.projectbe.core.dto;


import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@NoArgsConstructor
@AllArgsConstructor
@Data
public class ProductModelCategoryDto {

    private String categoryName;

    private String image;
```

```
}
package com.example.projectbe.core.dto;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.EqualsAndHashCode;
import lombok.NoArgsConstructor;

@NoArgsConstructor
@AllArgsConstructor
@Data
@EqualsAndHashCode(of = "modelName")
public class ProductModelDto {
    private Long id;

    private String modelName;

    private Double productPrice;

    private String image;
}

package com.example.projectbe.core.service;

import com.example.projectbe.core.dto.ShoesInfoDto;
import com.example.projectbe.core.dto.ShoesReviewCreateDto;
import com.example.projectbe.core.dto.ShoesReviewDto;
import com.example.projectbe.domain.entity.User;
import com.example.projectbe.domain.enums.Rating;

import java.util.List;

public interface ShoesReviewService {


    List<ShoesReviewDto> getShoesReviewByShoesId(Long shoesId);
    Long reviewCountByRatingAndShoesModelId(Rating rating, Long
shoesModelId);

    Long reviewCountByShoesModelId(Long shoesModelId);

    Double averageRatingByShoesModelId(Long shoesModelId);

    void createReview(ShoesReviewCreateDto shoesReviewCreateDto, User user);
```

```
}
package com.example.projectbe.domain.entity;

import lombok.Data;
import lombok.EqualsAndHashCode;
import lombok.NoArgsConstructor;

import javax.persistence.*;
import java.util.List;

@Entity
@NoArgsConstructor
@Data
@EqualsAndHashCode(callSuper = true)
public class ShoesModelType extends ProductModelType {

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "shoes_model_category_id")
    private ShoesModelCategory shoesModelCategory;

    @OneToMany(fetch = FetchType.LAZY, mappedBy = "shoesModelType")
    private List<ShoesModel> shoesModel;
}

package com.example.projectbe.web.controller;

import com.example.projectbe.core.dto.ProductModelCategoryDto;
import com.example.projectbe.core.dto.generic.ListResponseDto;
import com.example.projectbe.core.service.ProductModelCategoryService;
import lombok.RequiredArgsConstructor;
import org.springframework.security.access.prepost.PreAuthorize;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping("/api/v1/secure/categories")
@RequiredArgsConstructor
public class ProductModelCategoryController {

    private final ProductModelCategoryService productModelCategoryService;

    @GetMapping
    @PreAuthorize("authentication.principal.role == 'BUYER' OR
authentication.principal.role == 'ADMIN'")
```

```
    public ListResponseDto<ProductModelCategoryDto> getAllCategories() {
        ListResponseDto<ProductModelCategoryDto> listResponseDto = new
ListResponseDto<>();

listResponseDto.setElements(productModelCategoryService.findAllProductCategori
es());
        return listResponseDto;
    }

}
package com.example.projectbe.web.security.auth.ajax;

import
com.example.projectbe.core.expection.AuthMethodIsNotSupportedRuntimeExceptio
n;
import com.fasterxml.jackson.databind.ObjectMapper;
import org.springframework.http.HttpMethod;
import org.springframework.security.authentication.AuthenticationManager;
import
org.springframework.security.authentication.UsernamePasswordAuthenticationToke
n;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.AuthenticationException;
import org.springframework.security.core.context.SecurityContextHolder;
import
org.springframework.security.web.authentication.AbstractAuthenticationProcessingF
ilter;
import
org.springframework.security.web.authentication.AuthenticationFailureHandler;
import
org.springframework.security.web.authentication.AuthenticationSuccessHandler;

import javax.servlet.FilterChain;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

public class AuthenticationLoginProcessingFilter extends
AbstractAuthenticationProcessingFilter {


    private final AuthenticationSuccessHandler successHandler;
    private final AuthenticationFailureHandler failureHandler;
    private final ObjectMapper objectMapper;
```

```
    public AuthenticationLoginProcessingFilter(String defaultFilterProcessesUrl,
AuthenticationManager authenticationManager, AuthenticationSuccessHandler
successHandler, AuthenticationFailureHandler failureHandler, ObjectMapper
objectMapper) {
        super(defaultFilterProcessesUrl, authenticationManager);
        this.successHandler = successHandler;
        this.failureHandler = failureHandler;
        this.objectMapper = objectMapper;
    }


    @Override
    public Authentication attemptAuthentication(HttpServletRequest request,
HttpServletResponse response) throws AuthenticationException, IOException,
ServletException {
        if (!HttpMethod.POST.name().equals(request.getMethod())) {
            throw new AuthMethodIsNotSupportedRuntimeException("Authentication
method is not supported");
        }
        UserLoginDto userLoginDto = objectMapper.readValue(request.getReader(),
UserLoginDto.class);

        UsernamePasswordAuthenticationToken authenticationToken = new
UsernamePasswordAuthenticationToken(userLoginDto.getEmail(),
userLoginDto.getPassword());

        return this.getAuthenticationManager().authenticate(authenticationToken);
    }

    @Override
    protected void successfulAuthentication(HttpServletRequest request,
HttpServletResponse response, FilterChain chain, Authentication authResult) throws
IOException, ServletException {
        this.successHandler.onAuthenticationSuccess(request, response, authResult);
    }

    @Override
    protected void unsuccessfulAuthentication(HttpServletRequest request,
HttpServletResponse response, AuthenticationException failed) throws IOException,
ServletException {
        SecurityContextHolder.clearContext();
        this.failureHandler.onAuthenticationFailure(request, response, failed);
    }
}
```

ДОДАТОК Б

Відзив керівника економічного розділу

ДОДАТОК В

ПЕРЕЛІК ФАЙЛІВ НА ОПТИЧНОМУ НОСІЇ

Ім’я файла	**Опис**
Пояснювальні документи	
Диплом.doc	Пояснювальна записка до дипломного проекту. Документ Word.
Диплом.pdf	Пояснювальна записка до дипломного проекту в форматі PDF
Програма	
Shoes_magazine.zip	Архів. Містить коди програми і откомпільовану програму
Презентація	
Презентація.ppt	Презентація дипломного проекту