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 *Смигунова Іллі Володимировича*
(ПІБ)

академічної групи *121-17-1*
(шифр)

спеціальності *121 Інженерія програмного забезпечення*
(код і назва спеціальності)

освітньої програми *Інженерія програмного забезпечення*
(назва освітньої програми)

на тему: *Розробка програмного забезпечення підтримки та управління процесами он-лайн продажів товарів*

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

Дніпро
2021

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

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

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

« » _____ 2021 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *121-17-1* *Смигунова І.В.*
(група) (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка програмного заюезпечення підтримки та управління процесами он-лайн продажів*

затверджена наказом ректора НТУ «ДП» від *07.06.2021* № *317-с*

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***	***13.05.2021 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***27.05.2021 р.***

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

Завдання прийняв до виконання _____ *Смигунов І.В.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2021 р.*

Термін подання кваліфікаційної роботи до ЕК: *11.06.2021 р.*

РЕФЕРАТ

Пояснювальна записка: 72 с., 16 рис., 12 табл., 3 дод., 20 джерел.

Об'єкт розробки: автоматизована система управління та підтримки процесів онлайн продажів товарів електроніки.

Мета кваліфікаційної роботи проекту: розробка інтернет-магазину з продажу електроніки з гнучким функціоналом та можливістю подальшого розширення каталогу продукції.

У вступі відбувається аналіз сучасних технічних реалій, конкретизується мета створення нашої роботи на підставі цих самих реалій та пояснюється необхідність у розробці даного та подібних проектів в залежності від потреб сучасного суспільства.

У першому розділі ми проаналізуємо предметну галузь, визначимо актуальність поставленої задачі та призначення розробки нашого проекту, ознайомимося з базовими поняттями та термінами web-програмування, а також технологіями, які використовуються у наш час для вирішення подібних завдань та, зокрема, рішення конкретно поставленої перед нами задачі.

У другому розділі проаналізовані функціональні можливості об’єкту розробки, обрані та конкретизовані інструменти, за допомогою яких відбувається виконання завдання, здійснено проектування і розробка програми, а також детально описані принципи роботи програми.

У третьому розділі визначено трудомісткість розробки програмного забезпечення, підраховані витрати на створення програмного забезпечення і гаданий період розробки.

Практичне значення полягає у розробці додатку для ведення електронної комерції, створенні функціональних блоків проекту та реалізації обраних інструментів.

Актуальність виконання поставленої задачі зумовлена декількома факторами. По-перше, це комфорт. Щоб обрати якийсь товар та замовити потрібен лише доступ до мережі Інтернет. По-друге, через світову пандемію використання Інтернет магазинів замість звичайних зменшує ризик захворювання, адже це мінімізує контакт з людьми.

Список ключових слів: ІНТЕРНЕТ, ІНТЕРНЕТ-МАГАЗИН, ЕЛЕКТРОННА КОМЕРЦІЯ, КЛІЄНТ, СЕРВЕР, ВЕБ-САЙТ, БІЗНЕС, МЕНЮ, ВЕБ-ПРОГРАМУВАННЯ, МОВИ ПРОГРАМУВАННЯ, ФРЕЙМВОРК.

ABSTRACT

Explanatory note: 72 p., 16 figs., 17 tabl., 3 apps., 20 sources.

Object of development: automated system of management and support of processes of online sales of electronics.

The purpose of the qualifying work: the development of an online store for the sale of electronics with flexible functionality and the possibility of further expanding the product catalog.

The introduction analyzes modern technical realities, specifies the purpose of our work on the basis of these same realities and explains the need to develop this and similar projects depending on the needs of modern society.

In the first section we will analyze the subject area, determine the relevance of the task and purpose of our project, get acquainted with the basic concepts and terms of web-programming, as well as technologies used today to solve such problems and, in particular, the solution specifically set before us. tasks.

The second section analyzes the functionality of the object of development, selected and specified tools with which the task is performed, designed and developed the program, as well as describes in detail the principles of the program.

The third section identifies the complexity of software development, calculates the cost of creating software and the estimated period of development.

The practical significance lies in the development of an application for e-commerce, the creation of functional blocks of the project and the implementation of selected tools.

The urgency of the task is due to several factors. First, it's comfort. All you need to choose a product and order it is access to the Internet. Secondly, due to the global pandemic, the use of online stores instead of the usual ones reduces the risk of disease, because it minimizes contact with people.

Keyword list: INTERNET, ONLINE STORE, E-COMMERCE, CLIENT, SERVER, WEBSITE, BUSINESS, MENU, WEB PROGRAMMING, PROGRAMMING LANGUAGES, FRAMEWORK.

ЗМІСТ

РЕФЕРАТ 3
ABSTRACT 4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ 7
ВСТУП 8
РОЗДІЛ 1 9
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ 9
1.1.Загальні відомості з предметної галузі 9
1.2. Призначення розробки та галузь застосування 13
1.1 . Підстави для розробки 14
1.4. Постановка завдання 15
1.5.1. Вимоги до функціональних характеристик 16
1.5.2. Вимоги до інформаційної безпеки 16
1.5.3. Вимоги до складу та параметрів технічних засобів 17
1.5.4. Вимоги до інформаційної та програмної сумісності 17
РОЗДІЛ 2 18
ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ 18
2.1. Функціональне призначення програми 18
2.2. Опис застосованих математичних методів 18
2.3. Опис використаної архітектури та шаблонів проектування 18
2.4. Опис використанних технологій та мов програмування 20
2.5. Опис структури програми та алгоритмів її функціонування 29
2.6. Обґрунтування та організація вхідних та вихідних даних програми 37
2.7. Опис роботи розробленого програмного продукту 38
2.7.1. Використані технічні засоби 38
2.7.2. Використані програмні засоби 38
2.7.3. Виклик та завантаження програми 40
2.7.4. Опис інтерфейсу користувача 40
РОЗДІЛ 3 42
ЕКОНОМІЧНИЙ РОЗДІЛ 42
3.1. Розрахунок трудомісткості та вартості розробки програмного продукту 42

3.2. Розрахунок витрат на створення програми 45
ВИСНОВКИ 48
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 49
ДОДАТОК А. Код програми 51
ДОДАТОК Б. Відгук керівника економічного розділу 71
ДОДАТОК В. Перелік файлів на диску 72

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД – база даних;

WWW – World Wide Web;

ОС – Операційна система;

ПК – Персональний комп’ютер;

HTML – Hypertext Markup Language;

CSS – Cascading Style Sheet;

CMS – Content Management System;

PHP – Personal Home Page;

SQL – Structured Query Language;

XAMPP – Cross-Platform(X), Apache (A), MySQL (M), PHP (p), Perl (P).

ВСТУП

Технології завжди еволюціонували з часом. За допомогою інновацій у кожному секторі трудового життя, життя стало набагато простішим. Завдяки технологіям, що розвиваються, стало дуже легко знайти все необхідне віддалено за допомогою декількох кліків миші та перегляду через стабільне з’єднання з Інтернетом. На відміну від кожного сектору, комерційний сектор також розвивався поряд із зростаючими технологіями. Людям більше не потрібно виходити з дому, щоб збирати інформацію про те, що відбувається назовні, або навіть отримувати необхідні продукти.

Метою кваліфікаційної роботи є розробка автоматизованої системи онлайн продажів, яка допомагає зробити покупки швидкими, зручними та легкодоступними кожному з будь-якого місця за допомогою діючого Інтернет-з'єднання. Кожен, хто потребує електронних пристроїв (гаджетів), пропонованих компанією, отримує комфортні умови для покупки через Інтернет.

Інтернет-магазин об'єднує елементи прямого маркетингу з образом відвідування традиційного магазину. Відмінною рисою інтернет-магазинів в порівнянні зі звичайною формою торгівлі є те, що інтерактивний магазин може запропонувати значно більшу кількість товарів і послуг, і забезпечити споживачам значно більший обсяг інформації, необхідний для прийняття рішення, щодо покупки. Крім того, за рахунок використання комп'ютерних технологій можлива персоналізація підходу до кожного з клієнтів, виходячи з історії його відвідувань магазину і зроблених раніше покупок.

Через світову пандемію створення інструменту для здійснення онлайн продажів є надзвичайно важливим аспектом, бо, купляючи будь-який товар віддалено через Інтернет люди мінімізують контакт між собою і зменшують ризик зараження.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Загальні відомості з предметної галузі

Поглиблений розвиток технологій змінив повсякденне життя кожного в цьому світі. Інтернет став важливою частиною людського життя. Від дітей до людей похилого віку всі користуються Інтернетом із певною метою. Інтернет використовується для вільного спілкування, ділових операцій, навчання, покупок, зберігання даних, обміну новинами по всьому світу, досліджень та розробок. Інтернет подорожує через масу кабелів, маршрутизаторів, супутників та вежі Wi-Fi. Інтернет допомагає розробити короткі повідомлення між двома терміналами. Для швидкого, ефективного обміну контентом та в режимі реального часу використовуються соціальні мережі. Соціальні медіа допомагають охопити велику аудиторію в бізнесі. Соціальні мережі є важливим аспектом цифрового маркетингу, який забезпечує високі переваги, а також допомагає охопити мільйони клієнтів.

Електронна комерція стосується купівлі-продажу товарів чи послуг через Інтернет, а також передачі грошей та даних для успішного завершення транзакцій. Електронна комерція також охоплює такі заходи, як Інтернет-аукціони, Інтернет банкінг, платіжні шлюзи та онлайн-квитки. Бізнес-операції в електронній комерції можна здійснити шістьма способами: від споживача до споживача, від споживача до бізнесу, від бізнесу до бізнесу та від бізнесу до споживача, від бізнесу до адміністрації та від споживача до адміністрації.

Електронна комерція - популярний термін, що забезпечує найбільш ефективний та зручний спосіб здійснення операцій, пов’язаних із цією торгівлею. Електронна комерція (від англ. Electronic commerce) — це сфера цифрової економіки, що включає всі фінансові та торгові транзакції, які проводяться за допомогою комп'ютерних мереж, та бізнес-процеси, пов'язані з

проведенням цих транзакцій [1, 2]. Електронна комерція істотно знижує вартість транзакції. Електронна комерція допомагає економити час, сили та енергію як для покупців, так і для продавців. Електронна комерція перетворилася на велику компанію, яка допомагає отримувати величезний прибуток.

Традиційні компанії змушені змінювати технології ведення бізнесу та розробляти власні веб-сайти. У всьому світі понад 75% людей щодня використовують Інтернет для ділових та приватних цілей. Завдяки платформам електронної комерції роздрібні торговці будують Інтернет-магазини, де демонструють свої товари та послуги. Цифровий маркетинг використовує оптимізацію пошукових систем (SEO) - головний інструмент просування товарів та послуг на веб-сайті електронної комерції.

Концепція електронної комерції стрімко зростає, оскільки вона приносить нові ідеї та можливості у сфері бізнесу. Електронна комерція приваблює споживачів, надаючи послуги цілодобово з різноманітними товарами.

Технічна проблема є одним із факторів ризику веб-сайту електронної комерції, оскільки вона базується на технічній системі, а іноді може призвести до відмови всього веб-сайту електронної комерції. Електронна комерція приваблює споживачів та роздрібних торговців, незважаючи на низькі комерційні витрати.

Веб-сайт відіграє важливу роль у розвитку будь-якого бізнесу.

У сучасному суспільстві, що базується на зручності, люди, як правило, роблять покупки віддалено, а не відвідують місцеві магазини за покупками. Веб-сайт допомагає розширити сервіс, перенісши форму вашого бренду з традиційної до добре керованої віртуальної. Інтернет-магазин дозволяє клієнтам робити покупки або відвідувати ваш магазин у будь-який час доби, що робить його зручним як для покупців, так і для продавців. Охоплення - це ще одна перевага, яку можна отримати, маючи добре зроблений веб-сайт.

Інтернет дозволив переглядати веб-сайти з усього світу в будь-який момент часу.

Веб-сайт є хорошим місцем для користувачів для збору інформації про товари, які їх цікавлять, або про загальний бізнес, який їх цікавить. Довіра - ще один важливий фактор, який формує наявність належного веб-сайту. Окрім того, добре налагоджений інтернет магазин створює враження, що бізнес набагато успішніший, ніж може бути насправді.

Для реалізації серйозних бізнес проектів, які потребують підвищеної надійності і безпеки від несанкціонованого доступу, потрібні правильно підібрані інструменти - мови програмування, фреймфорки або системи управління контентом (CMS - Content Management System), що стають все більш актуальними.

Всі мови WEB-програмування можна класифікувати на клієнтські і серверні. Як випливає з назви, клієнтські мови використовуються для написання програм, які виконуються на стороні клієнта (WEB-браузер), а серверні - для програм, які виконуються на сервері.

Класифікації також підлягають і безпосередньо Інтернет-магазини. Розглянемо параметри, за якими можна класифікувати Інтернет-магазини [3].

За типом платформи:

– на власній CMS. Серед переваг можна назвати унікальність (і, як наслідок, впізнаваність) дизайну, зручне адміністрування і розширення функціоналу залежно від потреб проекту. Але при цьому варто врахувати, що доведеться витратитися на покупку ліцензії;

– на open source CMS (безкоштовної). Такий хід дозволить заощадити кошти і надасть можливість навчитися створювати інтернет-магазин своїми силами. З недоліків доведеться зіткнутися з обмеженим функціоналом, неможливістю отримати оперативну технічну підтримку, необхідністю внесення доробок і скачування додаткових модулів;

– на SaaS-платформі. Замовник отримує вже повністю готовий інтернет-магазин, функціонал якого продуманий до дрібниць. Також надається оперативна підтримка разом з автоматичними оновленнями. Оренда SaaS передбачає внесення щомісячної оплати. Якщо порівнювати цю платформу з власною CMS, то все ж функціонал комерційних сайтів, створених на SaaS, буде дещо обмежений, як і можливості вносити зміни до дизайну.

За асортиментом товару:

– спеціалізовані магазини. Здійснюється продаж тільки певних видів товарів (наприклад, аксесуари для мобільних гаджетів або нижня білизна);

– віртуальні супермаркети. У таких інтернет-магазинах вам запропонують будь-які товари. Асортимент може включати буквально все від предметів особистої гігієни і домашнього текстилю до великогабаритного обладнання і будівельних матеріалів.

За типом продажів:

– оптові. Відсутня можливість покупки штучних товарів, так як позиції реалізуються тільки великими партіями;

– роздрібні. Покупці можуть вибрати одиничні товари в необхідній кількості;

– оптово-роздрібні. Реалізація товарів здійснюється як оптом, так і в роздріб. Найчастіше оптовим покупцям пропонуються знижки та інші програми лояльності.

За типом товарів:

– нішеві. Реалізують товари, які користуються високим попитом і мають низьку вартість;

– брендові. Працюють з фірмовою продукцією, гарантійні зобов'язання на яку, як правило, надаються виробником;

– змішані. У продажу можна знайти як нішеві, так і брендові товари.

За географією продажів:

– регіональні;

– територія однієї країни;

– міжнародні.

За типом роботи:

– тільки онлайн-бізнес. Компанія здійснює свою діяльність тільки в Інтернеті. В офлайні може мати тільки склад або офіс + склад;

– Інтернет-магазин, який має стаціонарне представництво. Торгівля ведеться і в Інтернеті, і в звичайному магазині або торговому центрі.

1.2. Призначення розробки та галузь застосування

Розробка автоматизованої системи управління та підтримки процесів онлайн продажів товарів електроніки.

Розроблений продукт має використовуватися користувачами/потенційними покупцями для отримання досвіду комфортного здійснення транзакцій та одержання бажаного продукту, мінімізуючи затрачений час та зусилля.

Призначення розробки полягає наданні потенціальним користувачам наступних переваг автоматизованої системи, призначеної для реалізації онлайн продажів :

– можливість ознайомлення з асортиментом магазину цілодобово, 7 днів на тиждень;

– візуально представляє всі характеристики товару, що набагато зручніше користування послугами продавців-консультантів;

– дозволяє віддалено ознайомити покупців з акціями, розпродажами і новинками;

– дає можливість охопити максимально велику клієнтську базу без географічних обмежень;

– економить час покупців і служить додатковим засобом реклами для бізнесу.

Назвою для нашого електронного магазину було обрано *«Smigy Store»*, на основі назви також був розроблений логотип.

Функціонально в структурі нашої автоматизованої системи управління та підтримки процесів онлайн продажів товарів електроніки присутні блоки:

– віртуальний каталог з картками-описами товарів;
– оформлення замовлення;
– вибір способу оплати і доставки;
– особистий кабінет.

1.3. Підстави для розробки

Згідно з навчальним планом та відповідно до навчальної програми та графіка навчального процесу, наприкінці навчання студентами виконується кваліфікаційна робота.

Тема проекту є узгодженою з його керівником, кафедрою, а також затверджена наказом ректора

Отже, підставами для виконання кваліфікаційної роботи є:

– навчальна програма спеціальності 121 «Інженерія програмного забезпечення»;
– навчальний план та графік навчального процесу;
– наказ ректора Національного технічного університету «Дніпровська політехніка» № 317-с від 07.06.2021 р;
– завдання на кваліфікаційну роботу: «Розробка програмного забезпечення підтримки та управління процесами он-лайн продажів товарів».

1.4. Постановка завдання

Завданням є розробка автоматизованої системи управління та підтримки процесів онлайн продажів товарів електроніки. Обсяг вимог до інформаційного наповнення даного сайту дозволяє створити швидку і комфортну для користувача систему з гнучким функціоналом, зрозумілим і доступним інтерфейсом, можливістю подальшого розширення каталогу продукції. Дана система буде виконувати функції основного майданчика для ведення комерційної діяльності.

Реалізація поставленої мети передбачає вирішення наступних завдань:

Розробка системи онлайн продажів відповідно до структури бази даних, який буде володіти наступним функціоналом:

– відображення каталогу продукції компанії;

– здійснення сортування по групах товарів;

– здійснення реєстрації користувачів;

– створення зареєстрованим користувачем заявки на покупку, включаючи вибір способу оплати і доставки.

Оформлення замовлення онлайн:

– кожен покупець зможе купити товар віддалено з мінімальними затратами часу.

Публікація інформації про послуги і товари:

– каталог товарів постійно оновлюється, та, заходячи на даний веб сайт можна побачити усі актулальні товари та послуги, які можна придбати.

Власний онлайн магазин дозволить*:*

– зменшити витрати на зовнішню і медійну рекламу;

– збільшити дохід шляхом залучення клієнтів з інтернету;

– надати актуальну інформацію клієнтам і партнерам;

– налагодити оперативний зворотний зв'язок з клієнтами та

відвідувачами;

– представити власну компанію і товари або послуги в кращому ракурсі;

– сайт може бути платформою для навчання нових співробітників або сервісом для зв'язку та інше [4].

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Система повинна забезпечувати можливість виконання наступних функцій:

– ініціалізацію системи (введення списків покупців, переліків товарів тощо);

– зберігання інформації про покупців;

– отримання відомостей про поточний стан товарів на складі.

Початкові дані:

– обраний покупцем товар;

– наявність товару на складі;

– поточні відомості про можливості доставки товару.

Результати:

– запис покупця в базу даних;

– замовлення товару;

– оплата товару;

– доставка товару.

1.5.2. Вимоги до інформаційної безпеки

Основними вимогами до інформаційної безпеки є:

– забезпечення цілісності даних;

– захищеність від втручання в функціонал програмного продукту;
– конфіденційність інформації
– доступність інформації для усіх користувачів, які пройшли авторизацію.

Для того, щоб програмне забезпечення надійно функціонувало необхідно дотримуватися наступних вимог:

– використання лише ліцензійного програмного забезпечення;
– встановлення блоків безперебійного живлення;
– захищеність від зловмисних програм, які можуть завдати шкоди ПЗ.

1.5.3. Вимоги до складу та параметрів технічних засобів

Для коректного функціонування кінцевого продукту, а саме автоматизованої системи управління та підтримки процесів онлайн продажів товарів електроніки необхідними технічними умовами є:

– операційна система Windows, Linux, MacOS, Android або IOS;
– обсяг оперативної пам’яті не менш 4 ГБ;
– наявність будь-якого сучасного браузера, наприклад, Google Chrome або Opera;
– підтримка високошвидкісного Інтернету.

1.5.4. Вимоги до інформаційної та програмної сумісності

Для повної сумісності використовуваного для взаємодії з інтернет-магазином девайсу обов’язковими вимогами є встановлення актуальної версії мережевого драйвера, наявність стабільно швидкого Інтернет з’єднання та сучасного браузера, як Google Chrome, Opera, Safari тощо.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

2.1. Функціональне призначення програми

Призначення та ціль створеної автоматизованої системи управління та підтримки процесів онлайн продажів товарів електроніки - представлення компанії «Smigy Store» в мережі Інтернет, знайомство потенційних клієнтів та оптових покупців з продукцією, виробництвом компанії, автоматизація процесу заказу товарів.

У зв'язку з цим наша система онлайн продажів повинна задовольнити наступні умови:

– максимальна швидкість доступу до інформації;

– представлення інформації повинно бути максимально простим і чітким;

– швидкий доступ до основного розділу та розділів, що найбільш цікавлять потенційного покупця;

– оформлення та керування замовленнями повинно працювати безперебійно для того, щоб клієнт не зіткнувся з фінансовими втратами.

2.2. Опис застосованих математичних методів

У даному проекті математичні методи використані не були.

2.3. Опис використаної архітектури та шаблонів проектування

У програмній інженерії шаблоном проектування є загальне повторюване рішення загальнопоширеної проблеми при розробці програмного

забезпечення. Об'єктно-орієнтований шаблон зазвичай є зразком розв’язання проблеми та відображає відношення між класами та об'єктами, без чіткої вказівки на те, яким способом буде зрештою реалізоване дане відношення [5].

Для виконання завдання, поставленого перед нами умовами кваліфікаційної роботи, була обрана REST архітектура.

REST - це програмний архітектурний стиль, створений для керівництва розробкою та розробкою архітектури для Всесвітньої павутини (World Wide Web). REST визначає набір обмежень щодо того, як повинна поводити себе архітектура розподіленої гіпермедійної системи в масштабі Інтернету. Архітектурний стиль REST підкреслює масштабованість взаємодії між компонентами, єдиними інтерфейсами, незалежним розгортанням компонентів та створенням шаруватої архітектури для полегшення кешування компонентів, щоб зменшити сприйняту користувачем затримку, забезпечити безпеку та інкапсулювати застарілі системи [6]. REST застосовується в індустрії програмного забезпечення та є загальновизнаним набором настанов щодо створення надійних веб-сервісів.

Дані повинні передаватися у вигляді невеликої кількості стандартних форматів (наприклад, HTML, XML, JSON). Будь-який REST протокол (HTTP в тому числі) повинен підтримувати кешування, не повинен залежати від мережевого прошарку, не повинен зберігати інформації про стан між парами «запит-відповідь». Стверджується, що такий підхід забезпечує масштабовність системи і дозволяє їй еволюціонувати з новими вимогами.

Як відомо, web-сервіс - це додаток, що працює в World Wide Web і доступ до якого надається по HTTP-протоколу, а обмін інформацією йде за допомогою формату XML. Отже, формат даних переданих в тілі запиту буде завжди XML.

GET - отримує детальну інформацію про об'єкт.

POST - створює новий об'єкт. Дані передаються в тілі запиту без застосування кодування.

PUT - змінює дані з ідентифікатором {id}, можливо замінює їх.

DELETE - видаляє дані з ідентифікатором {id} [7].

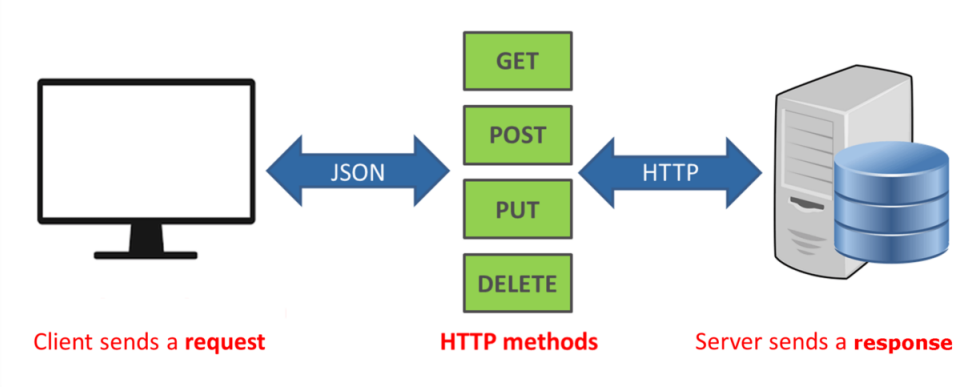


Рис. 2.1. Архітектура REST схематично

2.4. Опис використанних технологій та мов програмування

Для реалізації серйозних бізнес проектів, які потребують підвищеної надійності і безпеки від несанкціонованого доступу, потрібні правильно підібрані інструменти - мови програмування, фреймфорки або системи управління контентом (CMS - Content Management System), що стають все більш актуальними.

Всі мови WEB-програмування можна класифікувати на клієнтські і серверні. Як випливає з назви, клієнтські мови використовуються для написання програм, які виконуються на стороні клієнта (WEB-браузер), а серверні - для програм, які виконуються на сервері.

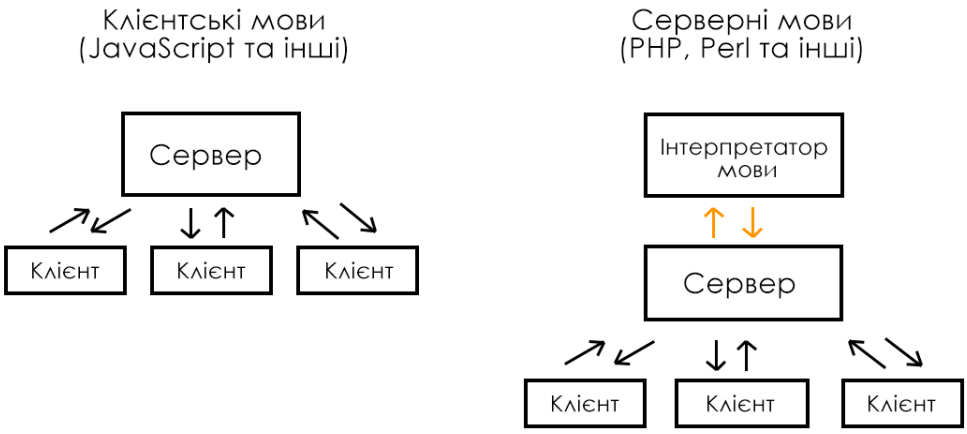


Рис 2.2. Уявлення клієнтських та серверних мов програмування

Серед клієнтських мов web-програмування треба виділити JavaScript, який, також як і HTML, CSS, лежить в основі багатьох web-технологій (наприклад, в основі популярної останнім часом технології AJAX) і вміння програмувати на ньому ставиться до базових знань web-розробника.

Фронтенд-фреймворки відповідають за зовнішній вигляд веб-додатків. На відміну від серверних, вони ніяк не пов'язані з логікою роботи. Цей тип фреймворків працює в браузері. З їх допомогою можна покращувати і впроваджувати нові інтерфейси, створювати різні анімації і односторінкові додатки. Ось деякі з них:

- Angular;
- Vue.js;
- Svelte;
- React - формально це не фреймворк, а бібліотека, але значення цього інструменту таке велике, що його постійно порівнюють з іншими веб-фреймворками [8].

Серверні мови web-програмування можуть бути умовно розділені по операційній системі, під управлінням якої вони працюють: Windows і Unix-подібними системами (* nix). Якщо говорити про ОС Windows, то тут

монопольну позицію займає технологія ASP.NET, розроблена компанією Microsoft. За допомогою ASP.NET можна створювати сайти будь-якого рівня складності - від найпростіших, що складаються з декількох сторінок, до дуже складних, обробних мільйони запитів в день (сайти Microsoft, написані на ASP.NET, є одними з найбільш відвідуваних в Інтернет). Завдяки технології .NET, розробка можлива на великій кількості мов програмування (C ++, Java, Python і т.д). Технологія ASP.NET приваблива для тих, хто непогано розбирається в ОС Windows, але незнайомий з Unix-подібними системами. Основний недолік - менша, в порівнянні з * nix, кількість дешевих хостингів або необхідність покупки серверної ліцензії, у випадку з виділеним хостингом. Однак, у порівнянні з вартістю розробки складних сайтів, а, також, вартістю трафіку, різниця витрат на Windows і * nix хостинг може бути дуже мала [9].

Найпопулярнішою мовою WEB-програмування є, безумовно, PHP - скриптова мова програмування загального призначення, яка інтенсивно застосовується для розробки web-додатків. Його основними перевагами є: безкоштовність, простий синтаксис, висока швидкодія і велика спільнота розробників. В даний час підтримується переважною більшістю хостинг-провайдерів і є одним з лідерів серед мов програмування, що застосовуються для створення динамічних web-сайтів. Проект поширюється під власною ліцензією, несумісною з GNU GPL. Синтаксис PHP подібний синтаксису мови Сі. Деякі елементи, такі як асоціативні масиви і цикл foreach, запозичені з Perl.

Окрім того, існує безліч бекенд-фреймворків - це фреймворки веб-розробки, які працюють на боці сервера. В основному вони відповідають за окремі, але критично важливі частини програми, без яких вона не зможе нормально працювати. Ось кілька найпопулярніших фреймворків, а також мови, з якими вони працюють:

- Django - Python;
- Symfony, Laravel - PHP;

– Express.js - JavaScript;

– Ruby on Rails – Ruby [10].

Правила та архітектура серверних фреймворків не дає можливості розробити веб-додаток з багатим інтерфейсом. Вони обмежені в своїй функціональності, проте ви все одно можете створювати прості сторінки і різні форми. Також вони можуть формувати вихідні дані і відповідати за безпеку в разі атак.

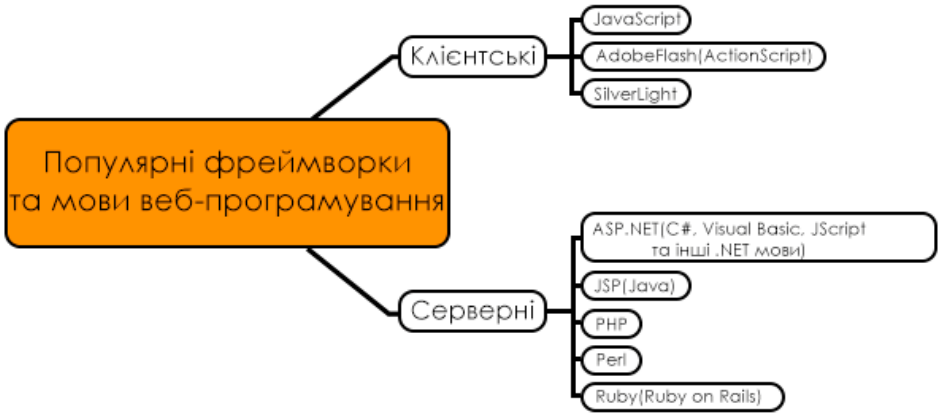


Рис 2.3. Популярні фреймворки та мови веб-програмування

Окрім перелічених вище інструментів, для реалізації веб-сайтів та інших Інтернет ресурсів також використовуються системи керування вмістом або ж CMS (Content Management System).

Система керування вмістом (СКВ; англ. Content Management System, CMS) — програмне забезпечення для організації веб-сайтів чи інших інформаційних ресурсів в Інтернеті чи окремих комп'ютерних мережах [11].

Існують сотні, а може, навіть й тисячі доступних CMS — систем. Завдяки їх функціональності ці системи можна використовувати в різних компаніях. Незважаючи на широкий вибір інструментальних та технічних засобів, наявних в CMS, існують загальні для більшості типів систем характеристики.

Серед найпопулярнішних CMS можна виділити Wordpress, Joomla!, Drupal.

Для виконання даної кваліфікаційної роботи були використані такі інструменти, як об'єктно-орієнтована мова програмування Groovy, фреймворк з відкритим кодом Spring Boot, система керування реляційними базами даних MySQL, засіб автоматизації роботи з програмними проектами Apache Maven та мова опису інтерфейсу для RESTful API Swagger.

GROOVY

Groovy — об'єктно-орієнтована динамічна мова програмування, що працює в середовищі JRE. Мова Groovy запозичила деякі корисні якості Ruby, Haskell і Python, але створена для роботи всередині віртуальної машини Java (JVM) і підтримує тісну інтеграцію з Java програмами. За роки існування Groovy навколо цієї мови сформувалася екосистема з пов'язаних проектів, таких як MVC веб-фреймворк Grails, swing-орієнтований фреймворк Griffon, системи збирання Gant і Gradle, інструментарій для інтеграції з Google App Engine - Gaelyk, система паралельного програмування Gpars, тестовий комплект Spock, інструменти для контролю якості CodeNarc і GMetrics [12].

Groovy має кілька цікавих особливостей:

– підтримка статичного та динамічного набору тексту: статично набрані мови - це мови, у яких перевірка типу здійснюється на етапі компіляції, а не під час виконання. Java є мовою статичного типу загального призначення. Синтаксиси динамічного набору, такі як Groovy - це ті, в яких перевірка виконується під час виконання. Python - ще один приклад;

– стислий, короткий, прямий синтаксис: це дозволяє розробникам, які використовують Groovy, швидше та простіше розробляти проекти;

– відносно коротка крива навчання: це порівняно проста мова, така як Python або Ruby;

– підтримка модульного тестування: Groovy - це орієнтована на тестування мова розробки. Насправді це синтаксис, який забезпечує підтримку запуску тестів в інтегрованих середовищах розробки (IDE), Ant або Maven, які є всіма засобами прикладного програмування на Java;

– власний синтаксис списків та асоціативних масивів: у програмуванні масиви зазвичай присвоюються змінним. Ці змінні часто асоціюються з даними. Розробники іноді пов'язують цю інформацію через різні елементи масиву через загальний потік, який називається index. Цей спосіб структурування інформації за допомогою програмування називається асоціативним масивом;

– власна підтримка мов розмітки, таких як XML та HTML;

– підтримка мов певних доменів: мова певного домену - це мова програмування або специфікація, призначена для вирішення конкретних проблем за допомогою певної техніки. Синтаксис загального призначення, такий як Java, C або C ++, є протилежним. Це забезпечує загальну структуру для вирішення глобальних ситуацій.

Рис 2.4. Логотип Groovy

SPRING BOOT

Spring Framework — це програмний каркас (фреймворк) з відкритим кодом та контейнери з підтримкою інверсії управління для платформи Java.

Основні особливості Spring Framework можуть бути використані будь-яким додатком Java, але є розширення для створення веб-додатків на платформі Java EE. Незважаючи на це, Spring Framework не нав'язує якоїсь конкретної моделі програмування, Spring Framework став популярним в спільноті Java як альтернатива, або навіть доповнення моделі Enterprise JavaBean (EJB).

Spring Boot - це один з численних проектів екосистеми Spring, але на відміну від більшості своїх «побратимів» він не вирішує якусь конкретну задачу, а являє собою скоріше новий етап розвитку Spring в цілому.

Мета Spring Boot полягає в тому, щоб спростити процес розробки додатків на основі Spring за допомогою їх створення на основі вже готових «наборів» програмних компонентів (так званих, «starter» пакетів), які вже включають достатній набір того, що необхідно для вирішення тієї чи іншої задачі та сконфігуровані відповідним чином [13].

Це позбавляє програміста не тільки від написання довгих конфігураційних файлів (особливо в XML), а й від необхідності налаштовувати різні компоненти для спільної роботи, Що дозволяє зосередитися на написанні прикладного коду.

Рис 2.5. Логотип Spring Boot

MYSQL

MySQL це система керування базами даних з відкритим вихідним кодом (РСКБД) з моделлю клієнт-сервер. РСКБД - це програмне забезпечення або служба, яка використовується для створення та управління базами даних на основі реляційної моделі.

MySQL виникла як спроба застосувати mSQL до власних розробок компанії: таблиць, для яких використовувалися ISAM — підпрограми низького рівня для індексного доступу до даних. У результаті був вироблений новий SQL-інтерфейс, але API-інтерфейс залишився в спадок від mSQL. Звідки походить назва «MySQL» — достеменно не відомо. Розробники дають два варіанти: або тому, що практично всі напрацювання компанії починалися з префікса My, або на честь дівчинки на ім'я My, дочки Майкла Монті Віденіуса, одного з розробників системи [14].

MySQL — компактний багатопотоковий сервер баз даних. Характеризується високою швидкістю, стійкістю і простотою використання.

MySQL вважається гарним рішенням для малих і середніх застосувань. Сирцеві коди сервера компілюються на багатьох платформах. Найповніше можливості сервера виявляються в UNIX-системах, де є підтримка багатопоточності, що підвищує продуктивність системи в цілому.

Можливості сервера MySQL:

- простота у встановленні та використанні;
- підтримується необмежена кількість користувачів, що одночасно працюють із БД;
- кількість рядків у таблицях може досягати 50 млн;
- висока швидкість виконання команд;
- наявність простої і ефективної системи безпеки.

Рис 2.6. Логотип MySQL

APACHE MAVEN

Apache Maven - фреймворк для автоматизації збірки проектів на основі опису їх структури в файлах на мові POM (англ. Project Object Model), що є підмножиною XML [13].

Maven забезпечує декларативну, а не імперативну (на відміну від засобу автоматизації збирання Apache Ant) збірку проекту. У файлах опису проекту міститься його специфікація, а не окремі команди виконання. Всі завдання по обробці файлів, описані в специфікації, Maven виконує за допомогою їх обробки послідовністю вбудованих і зовнішніх плагінів.

Maven використовується для побудови і управління проектами, написаними на Java, C #, Ruby, Scala, та інших мовах [15].

Одна з головних особливостей фреймворка - декларативний опис проекту. Це означає, що розробнику не потрібно приділяти увагу кожному аспекту збірки - всі необхідні параметри налаштовані за замовчуванням. Зміни потрібно вносити лише в тому обсязі, в якому програміст хоче відхилитися від стандартних налаштувань.

Ще одна перевага проекту - гнучке управління залежностями. Maven вміє довантажувати в свій локальний репозиторій сторонні бібліотеки, вибирати необхідну версію пакету, обробляти транзитивні залежності.

Рис 2.7. Логотип Maven

SWAGGER

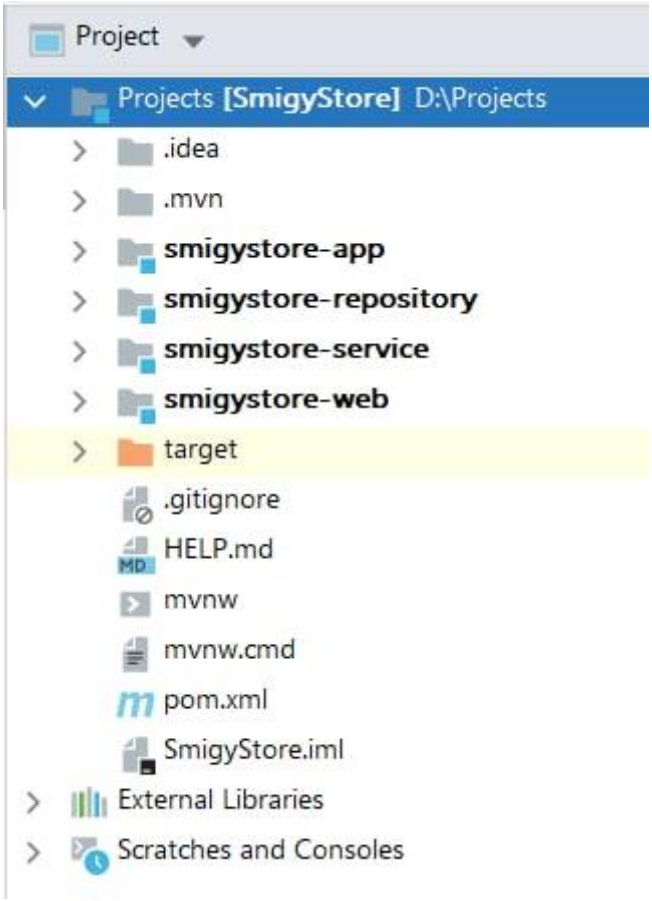
Swagger - це мова опису інтерфейсу для RESTful API, виражених за допомогою JSON. Swagger використовується разом із набором програмних засобів з відкритим кодом для проектування, побудови, документування та використання веб-служб RESTful. Swagger включає автоматизовану документацію, генерацію коду (на багатьох мовах програмування) та генерацію тестових кейсів [16].

Рис. 2.8. Логотип Swagger

2.5. Опис структури програми та алгоритмів її функціонування

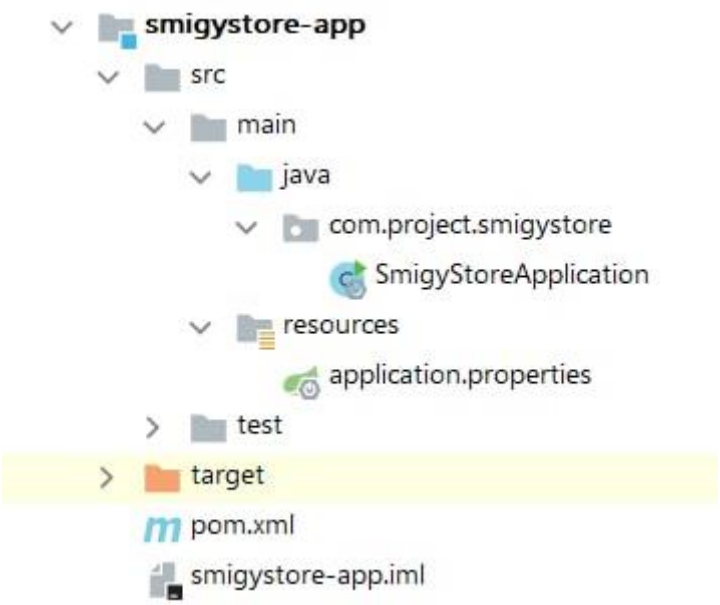
Автоматизована система онлайн продажів, розроблювана протягом виконання кваліфікаційної роботи має багатомодульну структуру. В рамках

розроблюваного додатку, модулем вважається окрема директорі, що має власну зону відповідальності.



2.9. Структура розроблюваного додатку

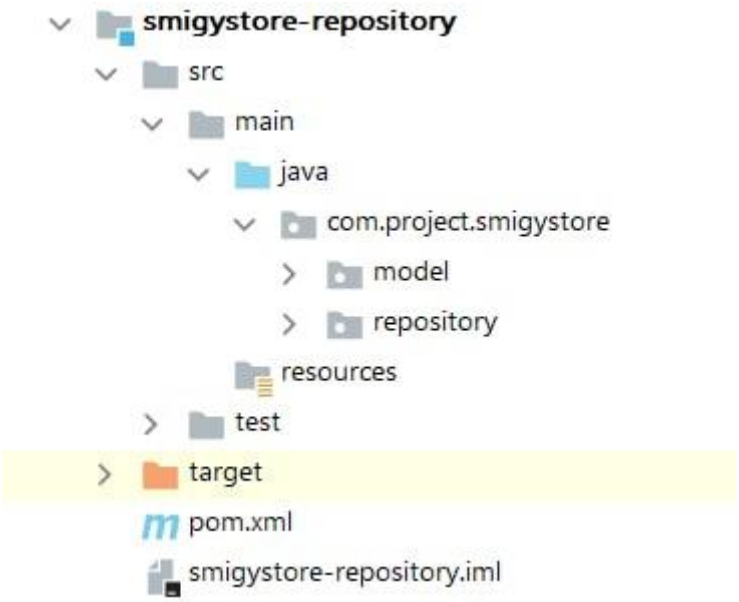
Модуль App є основними модулем розроблюваної системи. Він містить всередині класи конфігурацій, property файли (призначені, для того щоб зберігати в них якісь статичні дані, наприклад логін і пароль до БД) та головний клас нашого додатку Main.



2.10. Структура модуля App

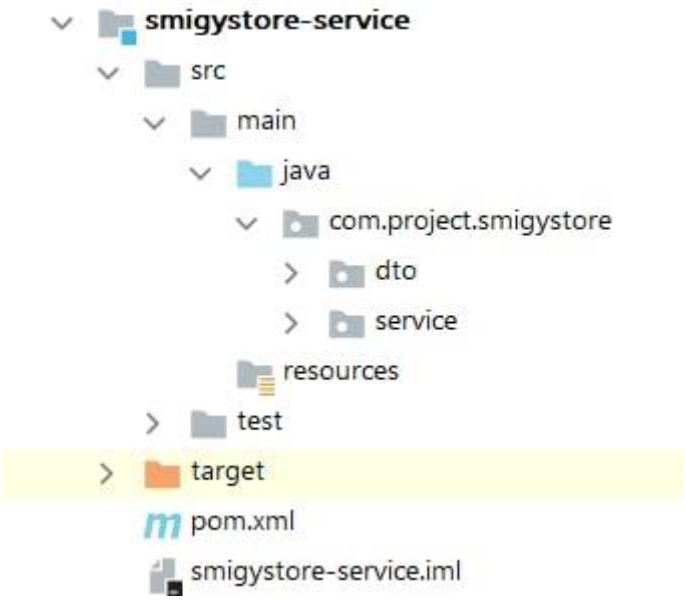
Модуль Repository відповідає за взаємодію з базою даних. Він містить всередині репозиторії, а також класи-моделі.

Репозиторій - це шар абстракції, що інкапсулює в собі все, що відноситься до способу зберігання даних. Призначення: відокремлення бізнес-логіки від деталей реалізації шару доступу до даних.



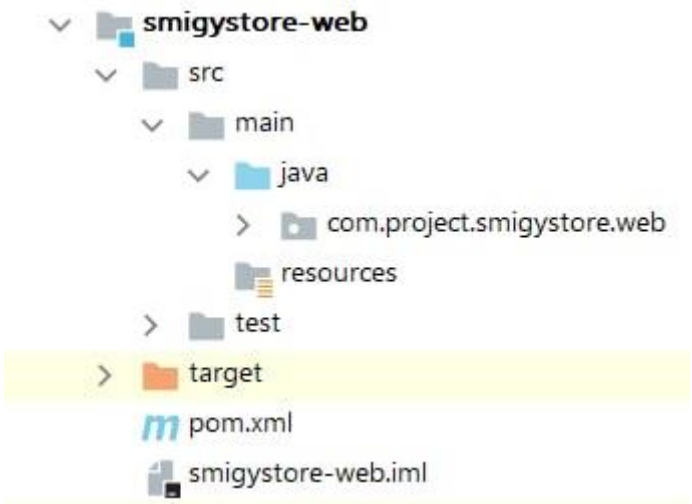
2.11. Структура модуля Repository

Модуль Service є відповідальним за бізнес-логіку нашого додатку. Бізнес-логіка - в розробці інформаційних систем - сукупність правил, принципів, залежностей поведінки об'єктів предметної галузі (сфери людської діяльності, яку система підтримує). Інакше можна сказати, що бізнес-логіка - це реалізація правил і обмежень автоматизованих операцій. Є синонімом терміна «логіка предметної галузі». Бізнес-логіка задає правила, яким підкоряються дані предметної галузі.



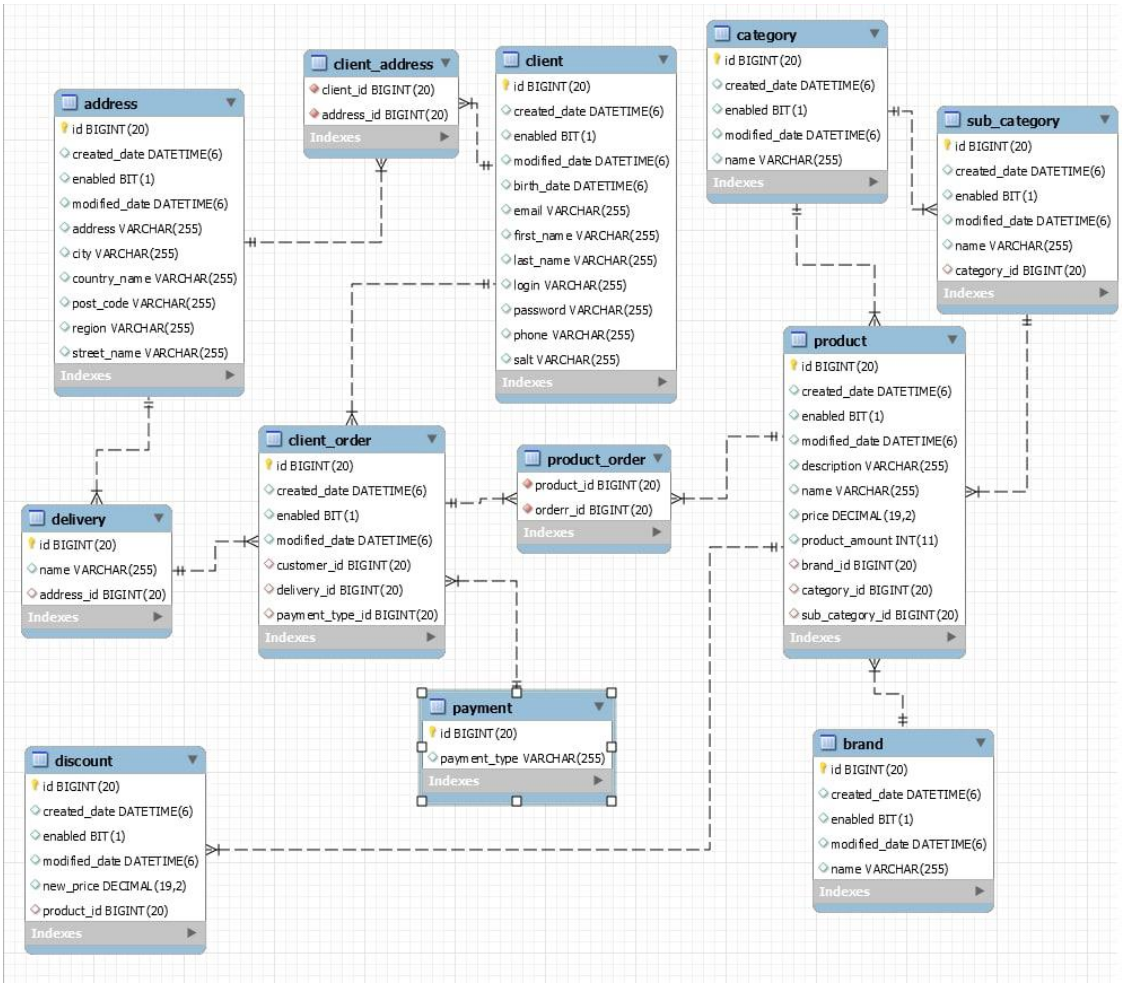
2.12. Структура модуля Service

Модуль WEB відповідає за зв’язок з клієнтською частиною. Він приймає вхідні запити та перенаправляє у відповідні сервіси за допомогою контролерів, які містяться всередині модуля.



2.13. Структура модуля WEB

База даних нашого проекту складається з 12-ти таблиць, зв’язаних між собою.



2.14. Схема бази даних

Таблиця 2.1.

Сутність «Client»

Назва	Ідентифікатор поля	Тип даних, довжина
id	id	BIGINT(20)
Дата створення	created_date	DATETIME(6)
Чи видалений/активний запис	enabled	BIT(1)
Дата редагування	modified_date	DATETIME(6)
Дата народження	birth_date	DATETIME(6)
Електронна пошта	email	VARCHAR(255)
Ім’я	first_name	VARCHAR(255)
Прізвище	last_name	VARCHAR(255)
Логін	login	VARCHAR(255)
Пароль	password	VARCHAR(255)
Телефон	phone	VARCHAR(255)
Сіль	salt	VARCHAR(255)

Таблиця 2.2.

Сутність «Client_address»

Назва	Ідентифікатор поля	Тип даних, довжина
id Клієнта	clinet_id	BIGINT(20)
id Адреси	address_id	BIGINT(20)

Таблиця 2.3.

Сутність «Address»

Назва	Ідентифікатор поля	Тип даних, довжина
id	id	BIGINT(20)
Дата створення	created_date	DATETIME(6)
Чи видалений/активний запис	enabled	BIT(1)
Дата редагування	modified_date	DATETIME(6)
Адреса	address	VARCHAR(255)
Місто	city	VARCHAR(255)
Країна	country_name	VARCHAR(255)
Поштовий індекс	post_code	VARCHAR(255)
Область	region	VARCHAR(255)
Вулиця	street_name	VARCHAR(255)

Таблиця 2.4.

Сутність «Delivery»

Назва	Ідентифікатор поля	Тип даних, довжина
id	id	BIGINT(20)
Спосіб доставки	name	VARCHAR(255)
id Адреси	address_id	BIGINT(20)

Таблиця 2.5.

Сутність «Client_order»

Назва	Ідентифікатор поля	Тип даних, довжина
id	id	BIGINT(20)
Дата створення	created_date	DATETIME(6)
Чи видалений/активний запис	enabled	BIT(1)
Дата редагування	modified_date	DATETIME(6)
id Покупця	customer_id	BIGINT(20)
id Способу доставки	delivery_id	BIGINT(20)
id Методу оплати	payment_type_id	BIGINT(20)

Таблиця 2.6.

Сутність «Payment»

Назва	Ідентифікатор поля	Тип даних, довжина
id	id	BIGINT(20)
Метод оплати	payment_type	VARCHAR(255)

Таблиця 2.7.

Сутність «Product_order»

Назва	Ідентифікатор поля	Тип даних, довжина
id Товару	product_id	BIGINT(20)
id Замовлення	order_id	BIGINT(20)

Таблиця 2.8.

Сутність «Product»

Назва	Ідентифікатор поля	Тип даних, довжина
id	id	BIGINT(20)

Закінч. табл. 2.8.

Дата створення	created_date	DATETIME(6)
Чи видалений/активний запис	enabled	BIT(1)
Дата редагування	modified_date	DATETIME(6)
Опис	description	VARCHAR(255)
Назва	name	VARCHAR(255)
Ціна	price	DECIMAL(19.2)
Кількість	product_amount	INT(11)
id Бренда	brand_id	BIGINT(20)
id Категорії	category_id	BIGINT(20)
id Субкатегорії	sub_category_id	BIGINT(20)

Таблиця 2.9.

Сутність «Category»

Назва	Ідентифікатор поля	Тип даних, довжина
id	id	BIGINT(20)
Дата створення	created_date	DATETIME(6)
Чи видалений/активний запис	enabled	BIT(1)
Дата редагування	modified_date	DATETIME(6)
Назва	name	VARCHAR(255)

Таблиця 2.10.

Сутність «Sub_category»

Назва	Ідентифікатор поля	Тип даних, довжина
id	id	BIGINT(20)
Дата створення	created_date	DATETIME(6)
Чи видалений/активний запис	enabled	BIT(1)
Дата редагування	modified_date	DATETIME(6)
Назва	name	VARCHAR(255)
id Категорії	category_id	BIGINT(20)

Таблиця 2.11.

Сутність «Brand»

Назва	Ідентифікатор поля	Тип даних, довжина
id	id	BIGINT(20)
Дата створення	created_date	DATETIME(6)
Чи видалений/активний запис	enabled	BIT(1)
Дата редагування	modified_date	DATETIME(6)
Назва	name	VARCHAR(255)

Таблиця 2.12.

Сутність «Discount»

Назва	Ідентифікатор поля	Тип даних, довжина
id	id	BIGINT(20)
Дата створення	created_date	DATETIME(6)
Чи видалений/активний запис	enabled	BIT(1)
Дата редагування	modified_date	DATETIME(6)
Нова ціна	new_price	DECIMAL(19.2)
id Товару	product_id	BIGINT(20)

2.6. Обґрунтування та організація вхідних та вихідних даних програми

Системою отримуються вхідні дані у форматі JSON, які формуються на клієнтській частині додатку та безпосередньо залежать від дій користувача. Після цього, надсилається відповідний запит у систему (в залежності від інформації, яку ми хочем отримати/операції, яку хочемо здійснити), запит оброблюється та система повертає відповідні вихідні дані назад (також у форматі JSON). JSON (англ. JavaScript Object Notation) - текстовий формат обміну даними, заснований на JavaScript. Як і багато інших текстових форматів, JSON легко читається людьми. Формат JSON був розроблений Дугласом Крокфордом [17].

2.7. Опис роботи розробленого програмного продукту

2.7.1. Використані технічні засоби

Для розробки кінцевого продукту кваліфікаційної роботи був використаний персональний комп'ютер з наступними технічними характеристиками:

- ЦП [CPU]: AMD Ryzen 5 2600 3.4 GHz;
- відеоадаптер [GPU]: nVidia GeForce 1660 ti;
- оперативна пам'ять [RAM] з обсягом: 16 ГБ;
- накопичувач постійної пам'яті з обсягом: 240 ГБ.

Для коректного функціонування даної автоматизованої системи рекомендовано використовувати обчислювальну машину з наступними мінімальними системними параметрами:

- ЦП [CPU]: Intel i3 6XXX / AMD FX 6XXX;
- відеоадаптер [GPU]: nVidia GeForce GT 640 / AMD Radeon HD 7730;
- оперативна пам'ять [RAM] з обсягом: 4 ГБ;
- стабільне Інтернет-з'єднання зі швидкістю 20 мбіт/сек.

2.7.2. Використані програмні засоби

В ході виконання кваліфікаційної роботи були використані наступні програмні засоби:

- IntelliJ IDEA 2020.3.3 x64;
- GitLab;
- Docker.

IntelliJ IDEA

IntelliJ IDEA - інтегроване середовище розробки програмного забезпечення для багатьох мов програмування, зокрема Java, JavaScript, Python, розроблена компанією JetBrains.

Починаючи з версії 9.0, середа доступна в двох редакціях: Community Edition і Ultimate Edition. Community Edition є повністю вільною версією, доступною під ліцензією Apache 2.0, в ній реалізована повна підтримка Java SE, Kotlin, Groovy, Scala, а також інтеграція з найбільш популярними системами управління версіями. В редакції Ultimate Edition, доступною під комерційною ліцензією, реалізована підтримка Java EE, UML-діаграм, підрахунок покриття коду, а також підтримка інших систем управління версіями, мов та фреймворків [18].

GitLab

GitLab - веб-додаток і система управління репозиторіями програмного коду для Git.

GitLab пропонує рішення для зберігання коду та спільної розробки масштабних програмних проектів. Репозиторій включає в себе систему контролю версій для розміщення різних ланцюжків розробки та гілок, дозволяючи розробникам перевіряти код і відкочуватися до стабільної версії софта в разі непередбачених проблем.

GitLab є конкурентом GitHub, в якому серед багатьох інших проектів розміщується розробка ядра Linux Лінуса Торвальдса. Оскільки GitLab розробляється на тій же основі управління версіями (Git), принцип їх роботи однаковий. GitLab підтримує як публічні, так і необмежену кількість приватних гілок розробки [19].

Docker

Docker - програмне забезпечення для автоматизації розгортання і управління додатками в середовищах з підтримкою контейнеризації,

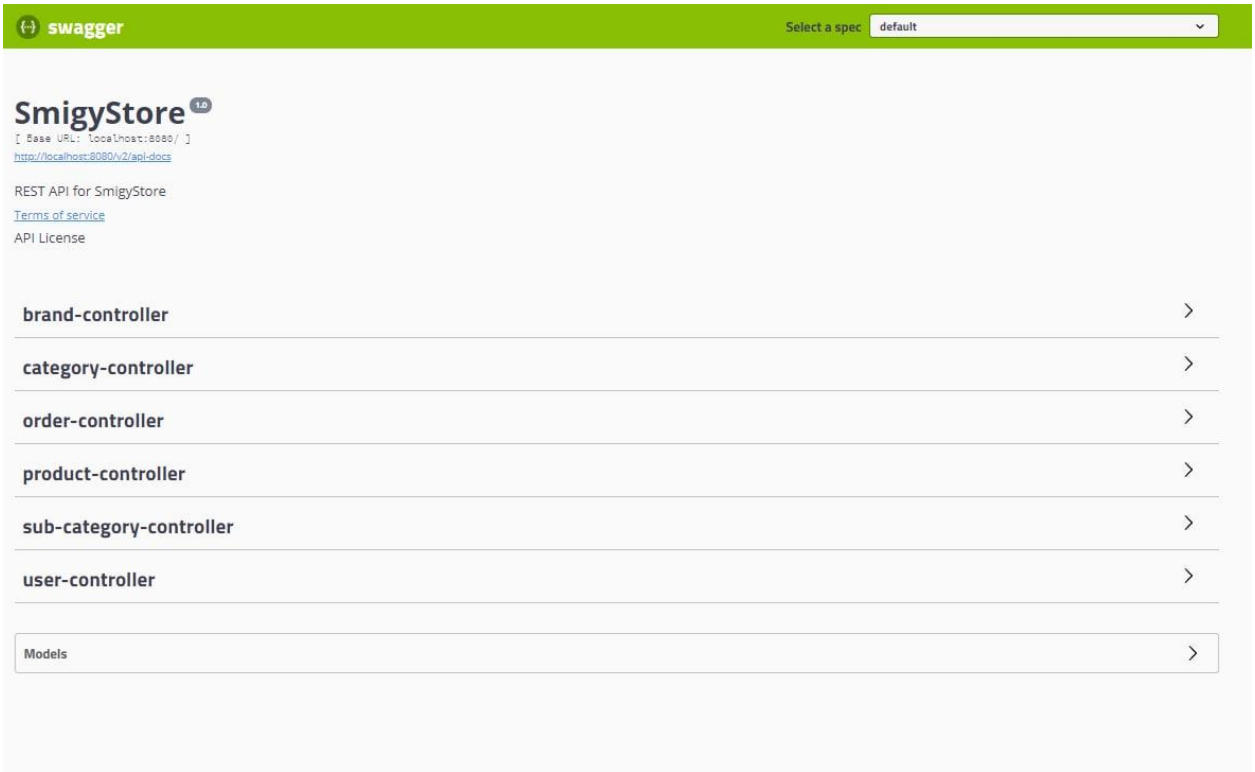
контейнерізатор додатків. Дозволяє «упакувати» додаток з усім його оточенням і залежностями в контейнер, який може бути розгорнутий на будь-який Linux-системі з підтримкою cgroups в ядрі, а також надає набір команд для управління цими контейнерами. Спочатку використовував можливості LXC, з 2015 року почав використовувати власну бібліотеку, що абстрагує віртуалізаційні можливості ядра Linux - libcontainer. З появою Open Container Initiative почався перехід від монолітної до модульної архітектури [20].

2.7.3. Виклик та завантаження програми

Для роботи програмного продукту необхідно мати встановленими на обчислювальній машині актуальні версії таких програмних інструментів, як IntelliJ IDEA, Java (11 версія), Groovy та MySQL. Щоб запустити додаток необхідно відкрити папку як проект IntelliJ IDEA та запустити його.

2.7.4. Опис інтерфейсу користувача

Сам по собі додаток немає інтерфейсу користувача, але має візуальну документацію, що описує доступні запити та відповіді. Виконати дану задачу можна за допомогою мови опису інтерфейсу для RESTful API Swagger.



2.15. Інтерфейс Swagger

2.16. Приклади запитів у Swagger

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Розрахунок трудомісткості та вартості розробки програмного продукту

Початкові дані:

1. передбачуване число операторів програми – 1038;

2. коефіцієнт складності програми – 1,27;

3. коефіцієнт корекції програми в ході її розробки – 0,06;

4. годинна заробітна плата програміста – 125 грн/год;

5. коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі – 1,2;

6. коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності – 0,9;

7. вартість машино-години ЕОМ – 14 грн/год.

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 можна розрахувати 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;

$t_п$ – 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):

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

q – передбачуване число операторів;

C – коефіцієнт складності програми;

p – коефіцієнт кореляції програми в ході її розробки.

$$Q = 1038 \cdot 1{,}27 \cdot (1 + 0{,}06) = 1326{,}17;$$

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

$$t_и = \frac{Q \cdot B}{(75 \ldots 85) \cdot K}, \text{ людино-годин,} \qquad (3.3)$$

де B – коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі;

K – коефіцієнт кваліфікації програміста, обумовлений стажем роботи з даної спеціальності;

$$t_и = \frac{1326{,}17 \cdot 1{,}2}{85 \cdot 0{,}9} = 16{,}85, \text{ людино-годин.}$$

Витрати праці на розробку алгоритму рішення задачі:

$$t_a = \frac{Q}{(20...25) \cdot K}; \qquad (3.4)$$

$t_a = \frac{1326,17}{20 \cdot 0,9} = 73,68$, людино-годин.

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20...25) \cdot K}; \qquad (3.5)$$

$t_n = \frac{1326,17}{25 \cdot 0,9} = 58,94$, людино-годин.

Витрати праці на налагодження програми на ЕОМ:

– за умови автономного налагодження одного завдання:

$$t_{отл} = \frac{Q}{(4...5) \cdot K}; \qquad (3.6)$$

$t_{отл} = \frac{1326,17}{5 \cdot 0,9} = 294,7$, людино-годин,

– за умови комплексного налагодження завдання:

$$t_{отл}^{к} = 1,2 \cdot t_{отл}; \qquad (3.7)$$

$t_{отл}^{к} = 1,2 \cdot 294,7 = 353,64$, людино-годин

Витрати праці на підготовку документації:

$$t_{\partial} = t_{\partial p} + t_{\partial o};$$ (3.8)

де $t_{\partial p}$ – трудомісткість підготовки матеріалів і рукопису

$$t_{\partial} = \frac{Q}{(15...20)\cdot K};$$ (3.9)

$$t_{\partial} = \frac{1326,17}{20 \cdot 0,9} = 73,68$$ людино-годин.

$t_{\partial o}$– трудомісткість редагування, печатки й оформлення документації

$$t_{\partial o} = 0,75 \cdot t_{\partial p};$$ (3.10)

$$t_{\partial o} = 0,75 \cdot 73,68 = 55,26,$$ людино-годин.

$$t_{\partial} = 73,68 + 55,26 = 128,94,$$ людино-годин.

Отримаємо трудомісткість розробки програмного забезпечення:

t = 50 + 16,85 + 73,68 + 58,94 + 294,7 + 128,94 = 623,11, людино-годин.

У результаті ми розрахували, що в загальній складності необхідно 623,11 людино-годин для розробки даного програмного забезпечення.

3.2. Розрахунок витрат на створення програми

Витрати на створення ПЗ Кпо включають витрати на заробітну плату виконавця програми Зз/п і витрат машинного часу, необхідного на налагодження програми на ЕОМ.

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

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

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

де t – загальна трудомісткість, людино-годин;

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

$$З_{ЗП} = 623{,}11 \cdot 125 = 77888{,}75, \text{ грн.}$$

$З_{МВ}$ – Вартість машинного часу, необхідного для налагодження програми на ЕОМ:

$$З_{МВ} = t_{отл} \cdot С_{М}, \text{ грн,} \qquad (3.13)$$

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

$С_{МЧ}$ – вартість машино-години ЕОМ, грн/год.

$$З_{МВ} = 294{,}7 \cdot 14 = 4125{,}8 \text{ грн.}$$

$$К_{ПО} = 77888{,}75 + 4125{,}8 = 82014{,}55 \text{ грн.}$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p}, \text{ мес.} \qquad (3.14)$$

де $B_к$- число виконавців;

$F_р$ – місячний фонд робочого часу (при 40 годинному робочому тижні Fр=176 годин).

$$T = \frac{623,11}{1 \cdot 176} = 3,5 \text{ міс.}$$

Висновки. На розробку даного програмного забезпечення піде 623,11 людино-годин. Тобто, ймовірна очікувана тривалість розробки складатиме 3,5 місяці при стандартному 40-годинному робочому тижні і 176-годинному робочому місяці. Очікувані витрати на створення програмного забезпечення складатимуть 82014,55 грн.

ВИСНОВКИ

Метою кваліфікаційної роботи є розробка автоматизованої системи управління та підтримки процесів онлайн продажів товарів електроніки.

Система являє собою онлайн магазин, що реалізує функції майданчика для ведення комерційної діяльності.

Система призначена для забезпечення діяльності магазину електроніки і надає можливість виконувати наступні дії:

– відображення каталогу продукції компанії, включаючи такі поля, як найменування, ціна, фільтр за категоріями і наявність на складі;

– здійснення сортування по групах товарів;

– здійснення реєстрації користувачів;

– створення зареєстрованим користувачем заявки на покупку, включаючи вибір способу оплати і доставки.

В «Економічному розділі» визначено трудомісткість розробки програмного забезпечення (623,11 чол-год), підраховані витрати на створення програмного забезпечення (82014,55 грн.) і гаданий період розробки (3.5 міс.).

Актуальність поставленої задачі обумовлюється широким попитом на такі продукти, бо кожна людина прагне до виконання повсякденних операцій з максимальним для неї комфортом, в тому числі й здійснення покупок. Інтернет-магазин як раз надає можливість користувачеві купити те, що йому потрібно не виходячи з дому у пару кліків, що дає змогу зекономити чимало сил та часу. Окрім того, в сучасних реаліях, коли світ охоплений жахливою пандемією користування онлайн сервісами набуває максимальної актуальності, адже це зменшує кількість контактів між людьми, що в свою чергу допомагає мінімізувати ризики захворювання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Алексунін В. Електронна комерція і маркетинг в Інтернеті / В. Алексунін, В. Родигина: Дашков і Ко. – 2005. – 216 с.

2. Ярова І. Інтернет як інструмент просування продукції промислових підприємств. – 2006. – С. 48-54.

3. Стефан Спенсер, Джиммі Хардінг, Дженніфер Шехан. Соціальна електронна комерція: збільшення продажів та розширення торгової марки: Санкт-Петербург, – 2015. – 310 с.

4. Botha, J., Bothma, C., Geldenhuys, P. Managing E-commerce in Business. Cape Town: Juta and Company Ltd. – 2005. – p. 3.

5. Алан Шаллоуей, Джеймс Р. Тротт. Шаблоны проектирования. Новый подход к объектно-ориентированному анализу и проектированию = Design Patterns Explained: A New Perspective on Object-Oriented Design. – М.: «Вильямс». – 2002. – 288 с.

6. Erik Wilde, Cesare Pautasso. REST: From Research to Practice. Springer Science & Business Media. – 2011. – 528 p.

7. Todd Fredrich. REST API Tutorial. – 2012. – p. 13-15.

8. Алгоритми і структура даних: Навчальний посібник / В.М.Ткачук. - ІваноФранківськ : Видавництво Прикарпатського національного університету імені Василя Стефаника. – 2016. – 286 с.

9. Алгоритми та структури даних. Навчальний посібник / Т. О. Коротєєва. Львів : Видавництво Львівської політехніки. – 2014. – 280 с.

10. Глоба Л. С. Розробка інформаційних ресурсів та систем: конспект лекцій / Л. С. Глоба, Т. М. Кот. – Київ : НТУУ "КПІ". – 2014. – 318 с.

11. Горнаков С. Г. Осваиваем популярные системы управления сайтом. – 2009. – 123 с.

12. Блинов, И.Н. Groovy. Промышленное программирование: практ. пособие / И.Н. Блинов, В.С. Романчик. – Минск : УниверсалПресс, 2007. – 704 с.

13. Санніков, Є.В. Курс практичного програмування. Об'єктно-орієнтоване програмування / Є.В. Санніков. – М .: Солон-Прес, 2013. – 188 с.

14. Яргер, Р.Дж. MySQL и mSQL: Базы данных для небольших предприятий и Интернета / Р.Дж. Яргер, Дж. Риз, Т. Кинг. – М.: СПб: Символ-Плюс, 2015. – 560 с.

15. Maven: The Definitive Guide. Sonatype Company. O'Reilly Media, Inc. – 2009. – p. 470.

16. Баженова І.Ю. Мови програмування: Підручник для студентів установ вищ. проф. освіти / І.Ю. Баженова; Під ред. В.А. Сухомлин. – М .: ВЦ Академія, 2018. – 368 с.

17. Оппель, Эндрю Дж. Запросы. Полное руководство / Оппель Эндрю Дж.. - М.: Диалектика / Вильямс, 2016. – 902 с.

18. Давидов С.Д. IntelliJ IDEA. Професійне програмування на Java / Станіслав Давидов, Олексій Єфімов. – М .: БХВ-Петербург, 2015. – 800 с.

19. Савітч, Уолтер. Курс програмування / Уолтер Савітч. – М .: Вільямс, 2010. – 928 с.

20. Хабибуллин И.С. Создание распределенных приложений на Java 2 / И.С/ Хабибуллин. – М.: БХВ-Петербург, 2017. – 704 с.

ДОДАТОК А

КОД ПРОГРАМИ

BaseEnttity.groovy

```
package com.project.smigystore.model

import javax.persistence.MappedSuperclass
import java.time.LocalDateTime

@MappedSuperclass
class BaseEntity {

    LocalDateTime createdDate

    LocalDateTime modifiedDate

    Boolean enabled = true
}
```

User.groovy

```
package com.project.smigystore.model

import com.fasterxml.jackson.annotation.JsonFormat

import javax.persistence.Entity
import javax.persistence.GeneratedValue
import javax.persistence.GenerationType
import javax.persistence.Id
import javax.persistence.JoinColumn
import javax.persistence.JoinTable
import javax.persistence.ManyToMany
import javax.persistence.OneToMany
import javax.persistence.Table

@Entity
@Table(name = "Client")
class User extends BaseEntity {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    Long id

    String firstName

    String lastName
```

```
    String password

    String salt

    String login

    String email

    @JsonFormat(shape = JsonFormat.Shape.STRING, pattern = "yyyy-MM-dd")
    Date birthDate

    String phone

    @ManyToMany
    @JoinTable(name = "client_address",
            joinColumns = @JoinColumn(name = "client_id"),
            inverseJoinColumns = @JoinColumn(name = "address_id"))
    List<Address> addresses

    @OneToMany(mappedBy = "customer")
    List<Order> orders

    Map<String, ?> toMap() {
        [
                id       : id,
                firstName: firstName,
                lastName : lastName,
                login    : login,
                email    : email,
                birthDate: birthDate,
                phone    : phone,
                addresses: addresses*.toMap()
        ]
    }

    Map<String, ?> toShortMap() {
        [
                id       : id,
                firstName: firstName,
                lastName : lastName,
                login    : login,
                email    : email
        ]
    }

}
```

Address.groovy

```
package com.project.smigystore.model
```

```
import javax.persistence.Entity
import javax.persistence.GeneratedValue
import javax.persistence.GenerationType
import javax.persistence.Id
import javax.persistence.ManyToMany
import javax.persistence.OneToMany

@Entity
class Address extends BaseEntity {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    Long id

    String countryName

    String region

    String city

    String streetName

    String postCode

    String address

    @OneToMany(mappedBy = "address")
    List<Delivery> deliveries

    @ManyToMany(mappedBy = "addresses")
    List<User> users

    Map<String, ?> toMap() {
        [
                id          : id,
                countryName: countryName,
                region      : region,
                city        : city,
                streetName : streetName,
                postCode    : postCode,
                address     : address
        ]
    }
}
```

Brand.groovy

```
package com.project.smigystore.model
```

```
import javax.persistence.Entity
import javax.persistence.GeneratedValue
import javax.persistence.GenerationType
import javax.persistence.Id
import javax.persistence.OneToMany

@Entity
class Brand extends BaseEntity {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    Long id

    String name

    @OneToMany(mappedBy = "brand")
    List<Product> products

    Map<String, ?> toMap() {
        return [
                id  : id,
                name: name
        ]
    }
}
```

Category.groovy

```
package com.project.smigystore.model

import javax.persistence.Entity
import javax.persistence.GeneratedValue
import javax.persistence.GenerationType
import javax.persistence.Id
import javax.persistence.OneToMany

@Entity
class Category extends BaseEntity {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    Long id

    String name

    @OneToMany(mappedBy = "category")
    List<SubCategory> subCategories

    @OneToMany(mappedBy = "category")
    List<Product> products
```

```
    Map<String, ?> toMap() {
        return [
                id  : id,
                name: name
        ]

    }

    Map<String, ?> toFullMap() {
        return [
                id          : id,
                name        : name,
                subCategories: subCategories*.toMap()
        ]

    }
}
```

Delivery.groovy

```
package com.project.smigystore.model

import javax.persistence.Entity
import javax.persistence.GeneratedValue
import javax.persistence.GenerationType
import javax.persistence.Id
import javax.persistence.ManyToOne
import javax.persistence.OneToMany

@Entity
class Delivery {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    Long id

    String name

    @ManyToOne
    Address address

    @OneToMany(mappedBy = "delivery")
    List<Order> orders

    Map<String, ?> toMap() {
        [
                id   : id,
                name : name
        ]
    }
```

```
}
```

Discount.groovy

```
package com.project.smigystore.model

import javax.persistence.Entity
import javax.persistence.GeneratedValue
import javax.persistence.GenerationType
import javax.persistence.Id
import javax.persistence.ManyToOne

@Entity
class Discount extends BaseEntity {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    Long id

    BigDecimal newPrice

    @ManyToOne
    Product product

}
```

Order.groovy

```
package com.project.smigystore.model

import javax.persistence.Entity
import javax.persistence.GeneratedValue
import javax.persistence.GenerationType
import javax.persistence.Id
import javax.persistence.ManyToMany
import javax.persistence.ManyToOne
import javax.persistence.Table

@Entity
@Table(name = "ClientOrder")
class Order extends BaseEntity {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    Long id

    @ManyToMany(mappedBy = "orders")
    List<Product> products

    @ManyToOne
    User customer
```

```
    @ManyToOne
    Delivery delivery

    @ManyToOne
    Payment paymentType

    Map<String, ?> toMap() {
        [
                id: id,
                products: products*.toMap(),
                customer: customer.toShortMap(),
                delivery: delivery.toMap(),
                paymentType: paymentType.toMap(),
                totalAmount: products.price.sum()
        ]

    }

}
```

Payment.groovy

```
package com.project.smigystore.model

import javax.persistence.Entity
import javax.persistence.GeneratedValue
import javax.persistence.GenerationType
import javax.persistence.Id
import javax.persistence.OneToMany

@Entity
class Payment {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    Long id

    String paymentType

    @OneToMany(mappedBy = "paymentType")
    List<Order> orders

    Map<String, ?> toMap() {
        [
                id     : id,
                payment: paymentType
        ]
    }

}
```

Product.groovy

```
package com.project.smigystore.model

import javax.persistence.Entity
import javax.persistence.GeneratedValue
import javax.persistence.GenerationType
import javax.persistence.Id
import javax.persistence.JoinColumn
import javax.persistence.JoinTable
import javax.persistence.ManyToMany
import javax.persistence.ManyToOne
import javax.persistence.OneToMany

@Entity
class Product extends BaseEntity {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    Long id

    String name

    BigDecimal price

    String description

    Integer productAmount

    @ManyToOne
    Brand brand

    @ManyToOne
    Category category

    @ManyToOne
    SubCategory subCategory

    @OneToMany(mappedBy = "product")
    List<Discount> discount

    @ManyToMany
    @JoinTable(name = "product_order",
            joinColumns = @JoinColumn(name = "product_id"),
            inverseJoinColumns = @JoinColumn(name = "orderr_id"))
    List<Order> orders

    Map<String, ?> toMap() {
        [
```

```
            id          : id,
            name        : name,
            price       : price,
            description  : description,
            productAmount: productAmount,
            brand       :
                [
                    brandId  : brand.id,
                    brandName: brand.name
                ],
            category    :
                [
                    categoryId  : category.id,
                    categoryName: category.name
                ],
            subCategory  :
                [
                    subCategoryId  : subCategory.id,
                    subCategoryName: subCategory.name
                ]

        ]
    }

}
```

SubCategory.groovy

```
package com.project.smigystore.model

import javax.persistence.Entity
import javax.persistence.GeneratedValue
import javax.persistence.GenerationType
import javax.persistence.Id
import javax.persistence.ManyToOne
import javax.persistence.OneToMany

@Entity
class SubCategory extends BaseEntity {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    Long id

    String name

    @ManyToOne
    Category category

    @OneToMany(mappedBy = "subCategory")
```

```
    List<Product> products


    Map<String, ?> toFullMap() {
        [
                category: category.name,
                id      : id,
                name    : name
        ]
    }
    Map<String, ?> toMap() {
        [
                id      : id,
                name    : name
        ]
    }
}
```

BrandService.groovy

```
package com.project.smigystore.service.impl

import com.project.smigystore.dto.brand.CreateBrandDto
import com.project.smigystore.dto.category.CreateCategoryDto
import com.project.smigystore.model.Brand
import com.project.smigystore.model.Category
import com.project.smigystore.repository.BrandRepository
import com.project.smigystore.service.CommonService
import org.springframework.beans.factory.annotation.Autowired
import org.springframework.stereotype.Service
import org.springframework.transaction.annotation.Transactional

@Service
@Transactional
class BrandService extends CommonService<Brand> {
    @Autowired
    BrandRepository brandRepository

    Brand createBrand(CreateBrandDto dto) {
        Brand brand = new Brand(
                name: dto.name
        )
        setCreatedDate(brand)
        brandRepository.save(brand)
    }

    @Override
    Collection<Brand> getAll() {
        brandRepository.findAll()
    }
```

```
    @Override
    Brand getById(Long id) {
        brandRepository.findById(id)
                .orElseThrow()
    }
}
```

CategoryService.groovy

```
package com.project.smigystore.service.impl

import com.project.smigystore.dto.category.CreateCategoryDto
import com.project.smigystore.dto.category.UpdateCategoryDto
import com.project.smigystore.dto.product.CreateProductDto
import com.project.smigystore.dto.user.UpdateUserDto
import com.project.smigystore.model.Category
import com.project.smigystore.model.Product
import com.project.smigystore.model.User
import com.project.smigystore.repository.CategoryRepository
import com.project.smigystore.service.CommonService
import org.springframework.beans.factory.annotation.Autowired
import org.springframework.stereotype.Service
import org.springframework.transaction.annotation.Transactional

import javax.persistence.EntityNotFoundException

@Service
@Transactional
class CategoryService extends CommonService<Category> {

    @Autowired
    CategoryRepository categoryRepository

    @Autowired
    SubCategoryService subCategoryService

    Category createCategory(CreateCategoryDto dto) {
        Category category = new Category(
                name: dto.name
        )
        setCreatedDate(category)
        categoryRepository.save(category)
    }

    Category updateCategory(Long id, UpdateCategoryDto dto) {
        Category category = categoryRepository.findById(id)
                .orElseThrow()
        category.name = dto.name
        category.subCategories = subCategoryService.getAll().findAll {
            subCategory -> dto.subCategoriesIds.each { id == subCategory.id }
```

```
        }
        updateModifiedDate(category)
        categoryRepository.save(category)
    }

    @Override
    Collection<Category> getAll() {
        categoryRepository.findAll()
    }

    @Override
    Category getById(Long id) {
        categoryRepository.findById(id)
                .orElseThrow()
    }
}
```

DeliveryService.groovy

```
package com.project.smigystore.service.impl

import com.project.smigystore.model.Delivery
import com.project.smigystore.repository.DeliveryRepository
import com.project.smigystore.service.CommonService
import org.springframework.beans.factory.annotation.Autowired
import org.springframework.stereotype.Service
import org.springframework.transaction.annotation.Transactional

@Service
@Transactional
class DeliveryService extends CommonService<Delivery> {

    @Autowired
    DeliveryRepository deliveryRepository

    @Override
    Collection<Delivery> getAll() {
        deliveryRepository.findAll()
    }

    @Override
    Delivery getById(Long id) {
        deliveryRepository.findById(id)
                .orElseThrow()
    }
}
```

OrderService.groovy

```
package com.project.smigystore.service.impl
```

```
import com.project.smigystore.dto.order.CreateOrderDto
import com.project.smigystore.model.Order
import com.project.smigystore.model.User
import com.project.smigystore.repository.OrderRepository
import com.project.smigystore.repository.ProductRepository
import com.project.smigystore.service.CommonService
import org.springframework.beans.factory.annotation.Autowired
import org.springframework.stereotype.Service
import org.springframework.transaction.annotation.Transactional

@Service
@Transactional
class OrderService extends CommonService<Order> {

    @Autowired
    OrderRepository orderRepository

    @Autowired
    UserService userService

    @Autowired
    ProductRepository productRepository

    @Autowired
    DeliveryService deliveryService

    @Autowired
    PaymentService paymentService

    Order createOrder(CreateOrderDto dto) {

        Order order = new Order(
                customer: userService.getById(dto.customerId),
                products: productRepository.findAllById(dto.productIds),
                paymentType: paymentService.getById(dto.paymentTypeId),
                delivery: deliveryService.getById(dto.deliveryId)
        )
        setCreatedDate(order)
        orderRepository.save(order)
    }

    Collection<Order> getAllByUser(Long userId) {
        User customer = userService.getById(userId)
        orderRepository.findAllByCustomer(customer)
    }

    @Override
    Collection<Order> getAll() {
        orderRepository.findAll()
```

```
    }

    @Override
    Order getById(Long id) {
        orderRepository.findById(id)
                .orElseThrow()
    }

    void cancelOrder(Long id) {
        Order order = getById(id)
        order.setEnabled(false)
        updateModifiedDate(order)
    }
}
```

PasswordService.groovy

```
package com.project.smigystore.service.impl

import com.google.common.hash.Hashing;
import org.apache.commons.lang3.RandomStringUtils;
import org.springframework.stereotype.Service

import java.nio.charset.StandardCharsets

@Service
class PasswordService {

    String generateSalt(){
        return RandomStringUtils.random(15,
"abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ-1234567890")
    }

    static String hashPassword(String password, String salt) {
        return Hashing.sha512()
                .hashString(password.concat(salt), StandardCharsets.UTF_8)
                .toString()
    }

}
```

PaymentService.groovy

```
package com.project.smigystore.service.impl

import com.project.smigystore.model.Payment
import com.project.smigystore.repository.PaymentRepository
import com.project.smigystore.service.CommonService
import org.springframework.beans.factory.annotation.Autowired
import org.springframework.stereotype.Service
import org.springframework.transaction.annotation.Transactional
```

```
@Service
@Transactional
class PaymentService extends CommonService<Payment> {

    @Autowired
    PaymentRepository paymentRepository

    @Override
    Collection<Payment> getAll() {
        paymentRepository.findAll()
    }

    @Override
    Payment getById(Long id) {
        paymentRepository.findById(id)
                .orElseThrow()
    }
}
```

ProductService.groovy

```
package com.project.smigystore.service.impl

import com.project.smigystore.dto.product.CreateProductDto
import com.project.smigystore.model.Product
import com.project.smigystore.repository.ProductRepository
import com.project.smigystore.service.CommonService
import org.springframework.beans.factory.annotation.Autowired
import org.springframework.stereotype.Service
import org.springframework.transaction.annotation.Transactional

@Service
@Transactional
class ProductService extends CommonService<Product> {

    @Autowired
    ProductRepository productRepository

    @Autowired
    CategoryService categoryService

    @Autowired
    SubCategoryService subCategoryService

    @Autowired
    BrandService brandService

    Product createProduct(CreateProductDto createProductDto) {
        Product product = new Product(
```

```
            name: createProductDto.name,
            price:  createProductDto.price,
            productAmount: createProductDto.productAmount,
            description: createProductDto.description,
            category: categoryService.getById(createProductDto.categoryId),
            subCategory: subCategoryService.getById(createProductDto.subCategoryId),
            brand: brandService.getById(createProductDto.brandId)
        )
        productRepository.save(product)
    }


    @Override
    Collection<Product> getAll() {
        productRepository.findAll()
    }

    @Override
    Product getById(Long id) {
        productRepository.findById(id)
                .orElseThrow()
    }

}
```

SubCategoryService.groovy

```
package com.project.smigystore.service.impl

import com.project.smigystore.dto.category.CreateCategoryDto
import com.project.smigystore.dto.category.UpdateCategoryDto
import com.project.smigystore.dto.subcategory.CreateSubCategoryDto
import com.project.smigystore.dto.subcategory.UpdateSubCategoryDto
import com.project.smigystore.model.Category
import com.project.smigystore.model.SubCategory
import com.project.smigystore.repository.SubCategoryRepository
import com.project.smigystore.service.CommonService
import org.springframework.beans.factory.annotation.Autowired
import org.springframework.stereotype.Service
import org.springframework.transaction.annotation.Transactional

@Service
@Transactional
class SubCategoryService extends CommonService<SubCategory> {

    @Autowired
    SubCategoryRepository subCategoryRepository

    @Autowired
```

```
    CategoryService categoryService

    SubCategory createSubCategory(CreateSubCategoryDto dto) {
        SubCategory subCategory = new SubCategory(
                name: dto.name,
                category: categoryService.getById(dto.categoryId)
        )
        setCreatedDate(subCategory)
        subCategoryRepository.save(subCategory)
    }

    SubCategory updateSubCategory(Long id, UpdateSubCategoryDto dto) {
        SubCategory subCategory = subCategoryRepository.findById(id)
                .orElseThrow()
        subCategory.name = dto.name
        subCategory.category = categoryService.getById(dto.categoryId)
        updateModifiedDate(subCategory)
        subCategoryRepository.save(subCategory)
    }

    @Override
    Collection<SubCategory> getAll() {
        subCategoryRepository.findAll()
    }

    @Override
    SubCategory getById(Long id) {
        subCategoryRepository.findById(id)
                .orElseThrow()
    }

    void delete(Long id) {
        subCategoryRepository.deleteById(id)
    }
}
```

UserService.groovy

```
package com.project.smigystore.service.impl

import com.project.smigystore.dto.product.CreateProductDto
import com.project.smigystore.dto.user.CreateUserDto
import com.project.smigystore.dto.user.UpdateUserDto
import com.project.smigystore.model.Product
import com.project.smigystore.model.User
import com.project.smigystore.repository.UserRepository
import com.project.smigystore.service.CommonService
import org.springframework.beans.factory.annotation.Autowired
import org.springframework.stereotype.Service
import org.springframework.transaction.annotation.Transactional
```

```
import javax.persistence.EntityExistsException
import javax.persistence.EntityNotFoundException

@Service
@Transactional
class UserService extends CommonService<User> {

    @Autowired
    UserRepository userRepository

    @Autowired
    PasswordService passwordService

    User createUser(CreateUserDto dto) {
        validateUniqueness(dto)
        String salt = passwordService.generateSalt();
        String password = PasswordService.hashPassword(dto.password, salt)
        User user = new User(
                firstName: dto.firstName,
                lastName: dto.lastName,
                login: dto.login,
                email: dto.email,
                password: password,
                salt: salt,
                birthDate: dto.birthDate,
                phone: dto.phone
        )
        setCreatedDate(user)
        userRepository.save(user)
    }

    User updateUser(Long id, UpdateUserDto dto) {
        User user = userRepository.findById(id)
                .orElseThrow(() -> new EntityNotFoundException("Cannot find user with given id: " +
id))
        user = dto.updateFromDto(user, dto)
        updateModifiedDate(user)
        user
    }



    @Override
    Collection<User> getAll() {
        return userRepository.findAll()
    }

    @Override
```

```
    User getById(Long id) {
        return userRepository.findById(id)
                .orElseThrow()
    }

    private void validateUniqueness(CreateUserDto createUserDto) {
        if (userRepository.findByLogin(createUserDto.login)
                || userRepository.findByEmail(createUserDto.email)) {
            throw new EntityExistsException()
        }
    }

    void delete(Long id) {
        User currentUser = userRepository.findById(id)
                .orElseThrow(() -> new EntityNotFoundException("Cannot find user with given id: " +
id))
        currentUser.setEnabled(false)
        updateModifiedDate(currentUser)
    }
}
```

CommonService.groovy

```
package com.project.smigystore.service

import com.project.smigystore.model.BaseEntity
import org.springframework.beans.factory.annotation.Autowired
import org.springframework.core.convert.ConversionService

import java.time.LocalDateTime

abstract class CommonService<Domain extends BaseEntity> {

    @Autowired
    protected ConversionService conversionService

    abstract Collection<Domain> getAll()

    abstract Domain getById(Long id)

    void setCreatedDate(Domain domain) {
        domain.setCreatedDate(LocalDateTime.now())
    }

    void updateModifiedDate(Domain domain) {
        domain.setModifiedDate(LocalDateTime.now())
    }

    void updateModifiedDate(Iterable<Domain> domains) {
        domains.forEach(this::updateModifiedDate)
```

```
    }
}
```

ДОДАТОК Б

Відгук керівника економічного розділу
на кваліфікаційну роботу бакалавра
на тему: «Розробка програмного забезпечення підтримки та управління процесами он-лайн продажів товарів»
студента групи 121-17-1 Смигунова Іллі Володимировича

ДОДАТОК В

Перелік файлів на диску

Ім’я файлу	Опис
Пояснювальні документи	
Кваліфікаційна робота Смигунов.docx	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Кваліфікаційна робота Смигунов.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF.
Програма	
Смигунов.rar	Архів. Містить коди програми і скомпільовану програму.
Презентація	
Смигунов.ppt	Презентація кваліфікаційної роботи.