Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Інформаційна технологія для підтримки інтернет торгівлі з прогнозуванням попиту на основі аналізу статистики попередніх продаж»

Виконав: студент 2 курсу, групи 2КІ-20м
напряму підготовки (спеціальності)
123 — «Комп'ютерна інженерія»
_____Каташинський Д. О.

Керівник: д.т.н., проф. каф. ОТ

_____ Ткаченко О. М.

«____» _____ 2021 р.

Опонент: д.т.н., проф., зав. каф ЗІ

_____ Кондратенко Н.Р.

«____» _____ 2021 р.

Допущено до захисту
Завідувач кафедри ОТ
д.т.н., проф. Азаров О.Д.
«____» _____ 2021 р.

Вінниця ВНТУ - 2021

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки
Рівень вищої освіти ІІ-й (магістерський)
Галузь знань 12 — Інформаційні технології
Спеціальність 123 — «Комп'ютерна інженерія»
Освітня програма — «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ
Завідувач кафедри
обчислювальної техніки
_____ проф., д.т.н. О.Д. Азаров

«___»_____2021 р.

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Каташинського Дмитра Олекссандровича

1 Тема роботи «Інформаційна технологія для підтримки інтернет торгівлі з прогнозуванням попиту на основі аналізу статистики попередніх продаж»
керівник роботи Ткаченко Олександр Миколайович, д.т.н., професор,
затверджені наказом вищого навчального закладу від 24.09.2021 р. №227

2 Строк подання студентом роботи 15.12.2021 р.

3 Вихідні дані до роботи — ключі для доступу до бази даних, дані доступних товарів, секретні ключі для доступу до Heroku.

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити: вступ, огляд і аналіз існуючих методів прогнозування попиту на основі статистики попередніх даних, розробка програми, тестування, розрахунок економічної доцільності створення інформаційної технології для підтримки інтернет торгівлі.

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): схема бази даних, структура бекенд частини.

6 Консультанти розділів роботи представлені в таблиці 1.

Таблиця 1 — Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
1,2,3	Ткаченко О. М.., д.т.н., професор		
4	Лесько О. Й., к.е.н., професор		

7 Дата видачі завдання 07.09.2021 р.

8 Календарний план наведено в таблиці 2.

Таблиця 2 — Календарний план

№ з/п	Назва етапів виконання магістерської роботи	Строк виконання етапів роботи	Примітка
1	Постановка задачі роботи	01.09.21	
2	Огляд технологій для створення серверної частини	02.09-09.09.21	
3	Огляд технологій для створення клієнтської частини	10.09-18.09.21	
4	Розробка серверної частини	19.09-01.10.21	
5	Розробка клієнської частиини	12.10-22.10.21	
6	Інтегрування бази даних для збереження даних	22.10-31.10.21	
7	Підключення сервісу для оплати товарів	01.11-10.11.21	
8	Огляд та інтегрування алгоритмів прогнозування попиту	11.11-16.11.21	
9	Розрахунок економічної частини роботи	17.11-30.11.21	
10	Оформлення пояснювальної записки та ілюстративного матеріалу	01.12-06.12.21	
11	Аналіз виконання роботи, висновки, додатки	07.12-06.12.21	
12	Перевірка якості виконання магістерської роботи та усунення недоліків	15.12.21	

Студент _____Каташинський Д. О.

Керівник роботи _____Ткаченко О. М.

АНОТАЦІЯ

УДК 004.42

Каташинський Д.О. Інформаційна технологія для підтримки інтернет торгівлі з прогнозуванням попиту на основі аналізу статистики попередніх продаж. Магістерська кваліфікаційна робота зі спеціальності 123 – комп’ютерна інженерія, освітня програма – комп’ютерна інженерія. Вінниця: ВНТУ, 2021. 135с.

На укр.мові. Бібліогр.: 52 назв; рис.: 21 ; табл. 10.

У магістерській кваліфікаційній роботі розроблено інформаційну технологію для підтримки інтернет торгівлі з прогнозуванням попиту на основі статистики попередніх данних за допомогою мови Node.js та шаблону MVC.

У роботі було створено прогнозщування попиту за допоммогою мови Phyton та технології Fast.AI.

Інформаційна технологія для підтримки інтернет торгівлі розроблена за допомогою мови Node.js та використанням середовища розробки WebStorm. Додаток підтримує дизайн для мобільних девайсів та комп’ютерів. Для створення тестових оплат було обрано платформу Stripe

В магістерській роботі були проведені економічні розрахунки для визначення доцільності та конкурентоспроможності створеного програмного продукту.

Ключові слова: попит, прогнозування попиту, штучний інтелект, javascript, nodejs, веб.

ANNOTATION

Katashinsky D. Information technology for e-commerce with demand forecasting based on statistics from previous data. Master's thesis in specialty 123 - computer engineering, educational program - computer engineering. Vinnytsia: VNTU, 2021. 135p.

In Ukrainian. Bibliogr .: 52 titles; fig .: 21; table 10.

In the master's qualification work, information technology was developed to support e-commerce with demand forecasting based on statistics of previous data using the Node.js language and the MVC template.

The paper created demand forecasting using Phyton language and Fast.AI technology.

Information technology to support e-commerce was developed using the Node.js language and using the WebStorm development environment. The application supports design for mobile devices and computers. The Stripe platform was chosen to create test payments

In the master's thesis, economic calculations were performed to determine the feasibility and competitiveness of the created software product.

Keywords: demand, demand forecasting, artificial intelligence, javascript, node js, web.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

БД — база даних,

СУБД — система утправління базою даних,

ПЗ — програмне забезпечення,

MVC — Model View Controller,

SQL — Structured Query Language,

AI — Artificial Intelligence,

ARIMA — Auto-Regressive Integrated Moving Average.

ВСТУП

Створення веб-сайтів для компаній є важливою і популярною сферою діяльності, оскільки веб-сайт компанії [1] в Інтернеті є дуже дешевим і популярним способом реклами, він дозволяє потенційним і існуючим клієнтам легко отримувати інформацію про товари та послуги [2], її бізнес-інтереси, які можуть допомагають знайти нових клієнтів і ділових партнерів і таким чином допомагають збільшити продажі та прибутковість [3].

Об’єктом дослідження є процес прогнозування попиту на основі статистики попередніх даних.

Предметом дослідження є методи і алгоритми прогнозування попиту за допомогою даних в попередніх продаж в окремих регіонах в певний проміжок часу.

Мета роботи — створити інформаційну технологію для інтернет торгівлі та прогнозувння попиту для покращення продаж.

Для досягнення цієї мети потрібно вирішити такі **задачі:**

— проаналізувати методів прогнозування попиту;

— виявити та обрати найбільш ефективні технології прогнозування попиту;

— обрати інструментарій для розробки програмного продукту;

— інтегрувати сервіс для оплати товарів;

— дослідити ефективність створеного прикладного програмного продукту.

Наукова новизна отриманих результатів магістерської роботи полягає у тому, що вдосконалено інформаційну технологію для підтримки інтернет торгівлі, який відрізняється від відомих прогнозуванням попиту на основі статистики попередніх даних, що дозволяє підвищити обсяши продажу товарів.

Практичне значення одержаних результатів магістерської роботи: розроблено програмне забезпечення для підтримки роботи інтернет магазину.

Основні результати роботи повідомлено та затверджено на Всеукраїнській науково-практичній онлайн-конференції «Молодь у науці: дослідження, проблеми, перспективи» (МН-2021) (Вінниця, 05.01.2021 — 14.05.2021) та Всеукраїнській науково-практичній онлайн-конференції «Молодь у науці: дослідження, проблеми, перспективи» (МН-2022) (Вінниця, 11.05.2022 — 13.05.2022).

За результатами дослідження опубліковано тези доповіді: Комп’ютерна система для підтримки інтернет торгівлі з прогнозуванням попиту на основі аналізу статистики попередніх продаж [Текст] Д. О. Каташинський // Молодь в науці: дослідження, проблеми, перспективи (МН-2021) Тез. доп. — Вінниця, 2021. — Режим доступу https://conferences.vntu.edu.ua index.php/mn/mn2021/paper/view/13144/11053 [1].

1 АНАЛІЗ СПОСОБІВ ПОБУДОВИ ІНФОРМАЦІЙНИХ СИСТЕМ

1.1 Мови для написання програмного забезпечення

Мова програмування — це мова для написання команд, які виконує комп'ютер. Оскільки мова програмування не зрозуміла комп’ютеру, має бути спеціальна програма, яка переводить символи цієї мови в двійкові символи машинних інструкцій. Це програма для перекладу символів під назвою translator, яка була створена на початку 1950-х років. Зауважте, що мови, близькі до числового коду процесора, називаються мовами низького рівня, а дружні для людини мови називаються мовами високого рівня. Мова найнижчого рівня — машинне кодування. Трохи вище є мова Асемблера, в якій машинні команди замінені мнемонічними скороченнями. Усі інші мови програмування є мовами вищого рівня, ніж асемблер.

Мови програмування також поділяються на процедурні (детальний опис вирішення проблем і використання процедур) і непроцедурні (використання об'єктів і декларацій). У свою чергу процедурні мови поділяються на операційні (Ассемблер, Basic, C тощо) і структурні (Pascal, Module). Серед непроцедурних мов виділяють об'єктно-орієнтовані мови (C++, Delphi, Smoltok) і декларативні мови (Prologue, Lisp).

Кожна мова програмування має три основні компоненти — синтаксис, алфавіт і семантику. C++ — мова програмування високого рівня з підтримкою кількох парадигм програмування:

— об'єктно-орієнтованої;

— узагальненої;

— процедурної.

Розроблено Б’ярне Страуструпом у AT&T Bell Laboratories (Мюррей-Хілл, Нью-Джерсі) у 1979 році та названо «C with Classes». Страуструп змінив назву C++ у 1983 році. Він заснований на мові C. У 1995 році C++ став однією з основних мов програмування загального призначення. Стандарт C ++ описує, як називати об'єкти, деякі деталі щодо обробки винятків та інші деталі реалізації, які роблять об'єктний код, створений різними компіляторами,

несумісним. Однак треті сторони встановили багато стандартів для конкретних архітектур та операційних систем [1-7].

Java — це об'єктно-орієнтована мова програмування, яка була представлена Sun Microsystems у 1995 році як ключовий компонент платформи Java. Синтаксис мови дуже схожий на C і C ++, а програми на Java складаються із визначень класів та інтерфейсу. Класи містять змінні та константи, які зберігають дані, методи, які виконують дії, і конструктори, які створюють екземпляри класів об’єктів. Дані можуть бути простого типу (байт, ціле число, символ) або ж посилання на об’єкт. Java друкується статично. Програми Java написані у Unicode та доступна лексична трансформація, яка дозволяє писати символи Unicode із керуючими кодами Unicode, використовуючи лише набір символів ASCII.

Java представляє текст у послідовностях 16-бітових одиниць коду із використанням кодування UTF-16. За винятком коментарів, ідентифікаторів і вмісту символьних та рядкових літералів, всі вхідні дані до програм Java складаються з символів ASCII чи відповідних їм керуючих кодів Unicode [8].

Java є суворо типізованою мовою, кожна змінна та вираз має тип, відомий на етапі компіляції. JavaScript — це повноцінна динамічна мова програмування, яка, у застосуванні до HTML документу, може надати динамічну інтерактивність на веб-сайтах. Вона була винайдена Бренданом Ейхом, співзасновником проекту Mozilla, the Mozilla Foundation, та Mozilla Corporation.

JavaScript має багато застосувань. Ви можете почати з бази даних створення каруселі, динамічних макетів сторінок, натискань кнопок. Завдяки цьому досвіду ви зможете створювати ігри, 2D та 3D-графіку, великі інтернет-додатки з використанням баз даних та багато іншого!

JavaScript — це компактна і гнучка мова. Творці зробили доступними велику кількість інструментів для доповнення основ мови JavaScript, які при мінімумі роботи мають ряд додаткових функцій. в тому числі:

— програмні інтерфейси (APIs) для браузерів API, що надають функціонал динамічного створення HTML та застосування CSS-стилів за допомогою веб-камери користувача;

— API третіх осіб, що використовують розробниками для інтегрувати у власні сайти функціонал інших провайдерів, таких як Twitter або Facebook;

— фреймворки та бібліотеки третіх осіб, які застосовуються до вашого HTML, щоб прискорити створення сайтів.

На основі JavaScript було створено багато фреймфорків та інших мов програмування одна з яких Nodejs, саме її ми будемо використовувати в даній роботі.

1.2 Node.js середовище виконання JavaScript

Node.js — це середовище виконання JavaScript, орієнтоване на події, для двигуна v8. Його головна мета — написати масштабовані мережеві програми. Програми, написані на JavaScript з використанням асинхронного введення-виводу, зменшують навантаження і підвищують масштабованість програми. Node.js використовує модель, орієнтовану на події, для обробки інтернет-запитів. Схематично ця модель показана на рисунку 1.1.

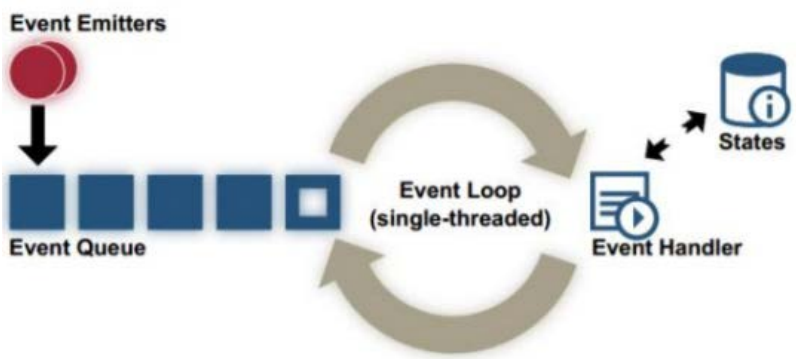

Рисунок 1.1 — Схематичне зображення подійно-орієнтованої архітектури

У цій моделі один потік виконання обробляє події в черзі один за одним, виконуючи відповідні обробники подій [10].

Середовище виконання node.js складається з наступних основних компонентів:

— рушія для мови JavaScript v8;

— бібліотеки для крос-платформенного введення;

— бібліотеки для DNS (c-ares).

Архітектура середовища виконання node.js показана на рисунку 1.2.

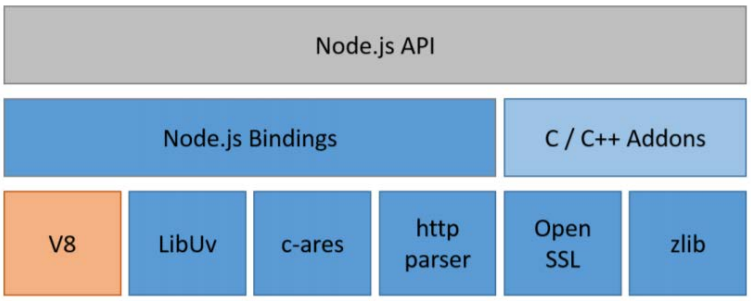

Рисунок 1.2 — Архітектура node.js

1.2.1 C++ модулі в середовищі розробки node.js

Node.js підтримує написання програм C++ за допомогою API v8 та libuv. Ці програми є динамічно зв'язаними бібліотеками, які дозволяють отримати доступ до коду C++ з JavaScript [11]. Щоб не блокувати основний потік, модулі зазвичай реалізують неблокуючий виклик, використовуючи пул потоків, наданий libuv. Типовий алгоритм такого модуля показаний на рисунку 1.3.

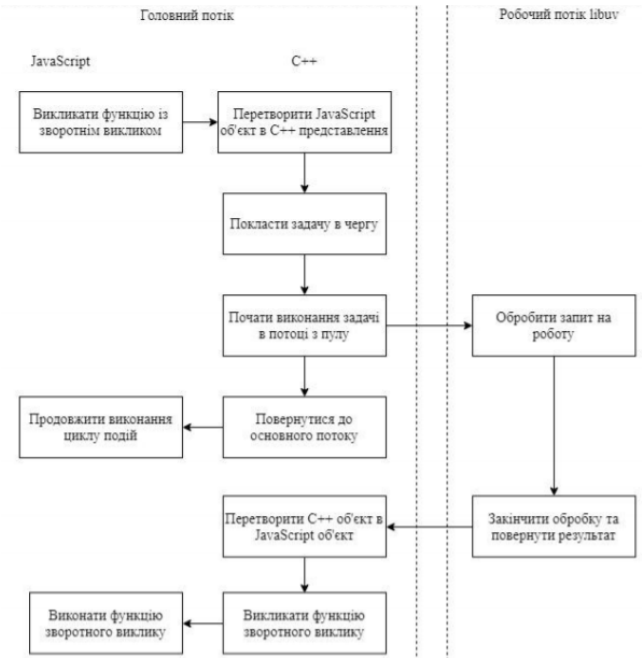

Рисунок 1.3 — Організація асинхронного виклику в модулі мовою C++

Коли викликається асинхронна функція, написана на C ++, спочатку перетворюються вхідні дані в об’єкт C ++. Завдання на виконання функції

потім поміщається в чергу завдань, надану libuv. Потім Libuv розподіляє завдання між потоками на одній із ітерацій циклу подій. Після завершення потік викликає функцію uv_async_send, яка надсилає результат у цикл подій. Наступна ітерація циклу викликає зворотний виклик із відповідним кодом JavaScript [8-12].

1.2.2 Альтернативні середовища виконання

Існують інші середовища виконання JavaScript, крім node.js, які використовують версію 8. Найпопулярнішими є Electron.js, NW.js. Вони сумісні з C++ API node.js і призначені для створення додатків для платформ MacOS, Linux і Windows. Зауважу, що модуль C ++ можна зробити сумісним з цим типом середовища виконання.

1.2.3 Promise API

Promise є альтернативою функціям зворотного виклику для представлення асинхронних обчислень. Цей інтерфейс є контейнером для значення, яке невідоме на момент створення об’єкта [12], статус цього об’єкта можна побачити на рисунку 1.4.

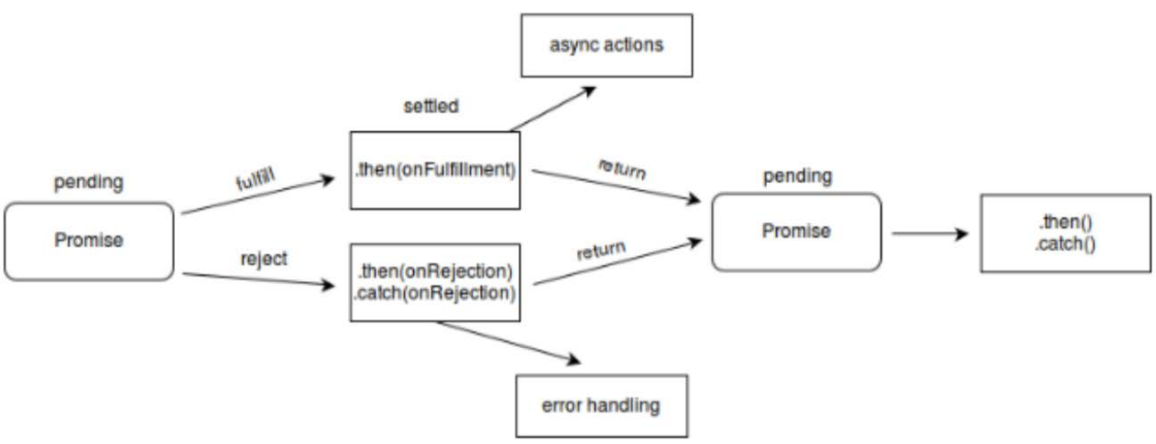

Рисунок 1.4 — Діаграма станів об'єкту Promise

Об'єкт Promise може перебувати в таких станах:

— у виконанні (розрахунки не проведені та незавершені);

— виконано (операція успішно завершена);

— відхилено (операція закінчилася помилкою).

Кожен метод об’єкта Promise повертає інший об’єкт Promise, що дозволяє створити асинхронний ланцюжок обчислень, подібний до синхронного коду.

Основними методами інтерфейсу Promise є обробка значення, коли воно стає доступним, і обробка помилки.

1.3 Огляд систем управління базами даних (СУБД)

У бізнесі вам доводиться працювати з даними з різних джерел, кожне з яких стосується певної діяльності. Для узгодження цих даних необхідні знання та навички. Електронна база даних (БД) — це структурована послідовність даних, що зберігаються на жорсткому диску комп’ютера.

Системи управління базами даних (СУБД) — це набір програмного забезпечення, необхідного для створення, використання та підтримки баз даних. База даних — це набір даних з наступними властивостями:

— дані логічно пов'язані між собою що несуть певну інформацію;

— структура баз даних відповідає тому набору даних, що вона містить;

— бази даних відображають тільки окремі аспекти реального світу.

Система управління базами даних (СУБД) об’єднує дані з різних джерел в одну реляційну базу даних. Сформовані форми, запити та звіти дозволяють швидко та ефективно оновлювати конкретні дані, отримувати відповіді на запитання, швидко знаходити потрібні дані, аналізувати дані, друкувати відповідні звіти, діаграми та наклейки.

Системи управління даними першого покоління. Бази даних першого покоління характеризуються тим, що всі групи користувачів розробили власне програмне забезпечення для управління даними. Наслідком такого сепаратизму стало надмірне зловживання кодами програм і даних.

Системи управління даними другого покоління.

Пов’язані файли даних об’єднуються в бази даних. СУБД створені для досвідчених користувачів як програмістів.

Системи управління даними третього покоління. Розширені можливості СУБД. Розроблені розширені інтерфейси, які забезпечують інтерактивний доступ для звичайних користувачів.

Переваги СУБД:

— зменшення надлишку даних;

— без баз даних неможливо уникнути зберігання великої кількості зайвих даних;

— за наявності центрального управління базою даних деякі дані можуть бути усунені;

— зайві дані не можуть бути повністю усунені, оскільки міркування щодо часу та надійності відіграють важливу роль у базі даних.

У світі існує багато СУБД. Незважаючи на те, що вони можуть використовуватися по-різному з різними об’єктами і надають користувачеві різні функції та інструменти, більшість СУБД засновані на добре встановленому наборі базових концепцій. Це дає нам можливість розглянути систему та узагальнити її концепції, прийоми та методи для всього класу СУБД. У цій статті буде використовуватися PostgreSQL, і ми обговоримо його більш детально нижче.

1.4 Класифікація баз даних

База даних (БД) — це впорядкований набір взаємопов’язаних даних, які спільно використовують і призначені для задоволення потреб користувачів. У технічному сенсі, включаючи систему управління базами даних.

Система управління базами даних (СУБД) — це набір програмних і мовних засобів для створення баз даних, їх оновлення та організації пошуку необхідної інформації. Централізований характер управління даними в базі даних вимагає існування особи, відповідальної за адміністрування даних, що зберігаються в базі даних.

Основним завданням бази даних є забезпечення того, щоб великий обсяг інформації зберігався і був доступним для користувача або прикладної програми. Тому база даних складається з двох частин — збереженої

інформації та системи управління. Для забезпечення безперебійного доступу записи даних організовані як набір фактів (елемент даних).

Існує величезна кількість баз даних, що відрізняються за критеріями (Енциклопедія технології баз даних [21] виділяє понад 50 типів баз даних). Звернимо увагу лише на основні класифікації. Класифікація бази даних відповідно до моделі даних:

— ієрархічні;
— мережеві;
— реляційні;
— об'єктні;
— об'єктно-орієнтовані;
— об'єктно-реляційні.

Класифікація баз даних за технологією фізичного зберігання:

— БД у вторинній пам'яті (традиційна);
— БД у оперативній пам'яті (in-memory databases);
— третинні бази даних.
— Класифікація баз даних за змістом:
— географічні;
— історичні;
— наукові;
— мультимедіа.

Класифікація баз даних за ступенем поширення:

— централізовані (концентровані);
— поширені.

Особливе місце в теорії і практиці займають просторові (час), часові або часові (час) і просторово-часові (простірно-часові) бази даних. Ієрархічні бази даних представлені у вигляді дерева об’єктів різного рівня. Верхній рівень займає один об'єкт, другий — об'єкти іншого рівня і т.д. Між об'єктами існують відносини, кожен об'єкт може містити кілька об'єктів нижнього рівня. Такі предмети залишаються по відношенню до предка (об’єкт ближче до

кореня) до нащадка, і можливо, що елемент-предок не має нащадків або має декілька, тоді як дочірній елемент може мати лише одного предка. Об’єкти, які мають спільного предка, називаються близнюками. Веб-бази даних подібні до ієрархічних баз даних, за винятком того, що вони мають вказівники в обох напрямках, які з’єднують пов’язану інформацію.

Основними поняттями моделі мережі бази даних є: рівень, елемент, зв'язок. Вузол — це сукупність даних, що описують об’єкт. В ієрархічній деревоподібній діаграмі вузли представлені кінцевими вершинами діаграми. У структурі мережі будь-який елемент може бути пов'язаний з будь-яким іншим елементом.

Хоча ця модель вирішує проблеми, пов’язані з ієрархічною моделлю, виконання простих запитів залишається досить складним процесом. Крім того, оскільки логіка процедури вибірки даних залежить від фізичної організації даних, модель не є повністю програмною незалежною. Іншими словами, якщо ви хочете змінити структуру даних, вам доведеться змінити програму.

Модель реляційної бази даних зосереджена на організації даних у вигляді двовимірних таблиць. Кожна реляційна таблиця є двовимірним масивом і має такі властивості:

— кожен елемент таблиці є елементом даних;

— усі клітинки в стовпці таблиці однорідні, тобто елементи стовпця однотипні (числові, символічні тощо);

— кожен стовпець має унікальну назву;

— такі ж рядки відсутні в таблиці;

— порядок рядків і стовпців може бути будь-яким.

Об’єктно-орієнтована СУБД ідеально підходить для застосування складних інтерпретацій даних, на відміну від реляційної СУБД, де додавання нового типу даних досягається ціною втрати продуктивності або значного збільшення часу та вартості розробки додатків. Об’єктна база даних, на відміну від реляційної бази даних, не потребує модифікації ядра при додаванні нового типу даних. Новий клас та його екземпляри просто входять до

зовнішньої структури бази даних. Їхня система управління залишається незмінною.

Об’єктно-орієнтована база даних (OBD) — це база даних, де дані форматуються як об’єктні моделі, що містять програми, якими керують зовнішні події. Об'єктно-орієнтовані системи управління базами даних (СУБД) є результатом поєднання всіх можливостей (особливостей) баз даних і можливостей, а також об'єктно-орієнтованих мов програмування. OOSUBD дозволяє вам працювати з об'єктами бази даних так само, як і зі звичайними об'єктами програмування в об'єктно-орієнтованих мовах програмування. OOSUBD розширює мови програмування, вводячи довгострокові дані, керування паралельністю, відновлення даних, запити тощо.

Об’єктно-орієнтовані бази даних рекомендуються в тих випадках, коли слід використовувати високопродуктивну обробку даних зі складною структурою. Система, яка забезпечує інфраструктуру об'єктів і набір реляційних розширювачів, називається «об'єктно-реляційною».

Об'єктно-реляційні системи поєднують переваги сучасних об'єктно-орієнтованих мов програмування з властивостями реляційних систем, такими як багаторазове представлення цих даних і непроцедурні мови запитів високого рівня. Завдяки технології обробки даних бази даних можна розділити на централізовані та розподілені.

Централізована база даних зберігається в пам'яті однієї комп'ютерної системи. Якщо комп’ютерна система є частиною комп’ютерної мережі, можливий розподілений доступ до такої бази даних. Цей метод використання баз даних часто використовується в локальних мережах ПК.

Розподілена база даних складається з кількох баз даних, можливо, що перетинаються або навіть дублюються частин, які зберігаються на різних комп’ютерах у комп’ютерній мережі. Робота з такою базою даних відбувається за допомогою розподіленої системи керування базами даних (СУБД) [16-21]. За способом доступу до даних бази даних поділяють на бази з локальним і віддаленим (мережевим) доступом.

Системи баз даних централізованого доступу до мережі включають різні архітектури таких систем:

— файл-сервер;

— клієнт-сервер.

Файловий сервер. Архітектура систем баз даних з доступом полягає у виділенні однієї з мережевих машин як штаб-квартири (сервера). Спільна база даних зберігається на такій машині. Усі інші машини в мережі діють як робочі станції, які підтримують системний доступ користувача до централізованої бази даних. Файли бази даних для запитів користувачів надсилаються на робочі станції, де вони в основному обробляються. При високій інтенсивності доступу до тих же даних продуктивність IT-системи знижується. Користувачі також мають можливість створювати локальні бази даних на робочих станціях, які вони використовують монополістично.

Клієнтський сервер. У цій концепції, окрім зберігання централізованої бази даних, центральна машина повинна забезпечувати реалізацію обсягу оброблених даних. Запит даних, виданий клієнтом (робочою станцією), генерує пошук і вилучення даних на сервері. Видобуті дані транспортуються по мережі від сервера до клієнта. Специфікою архітектури є використання мови запитів SOL.

1.5 Каласифікація основних СУБД

MySQL — це безкоштовна система керування базами даних. MySQL належить корпорації Oracle, яка придбала його разом із придбаною Sun Microsystems, яка розробляє та підтримує програму. Поширюється під загальною суспільною ліцензією GNU та власною комерційною ліцензією, необов’язково. Крім того, програмісти створюють функціонал для замовлення ліцензованих користувачів, завдяки такому порядку механізм реплікації з'явився практично в самих ранніх версіях.

Ця система управління базами даних з відкритим вихідним кодом була створена як альтернатива комерційним системам. MySQL з самого початку був дуже схожий на mSQL, але з часом виріс і зараз є однією з найпопулярніших

систем керування базами даних. В основному він використовується для створення динамічних веб-сайтів, оскільки має відмінну підтримку різними мовами програмування.

MySQL — це рішення для малих і середніх додатків. MySQL зазвичай використовується як сервер, до якого звертаються локальні або віддалені клієнти, але дистрибутив містить внутрішню серверну бібліотеку, яка дозволяє включати MySQL в окремі програми. Вихідний код сервера скомпільовано на кількох платформах. Найповніші можливості сервера доступні в системах UNIX, де є підтримка багатопотокової роботи, що підвищує продуктивність системи в цілому.

Гнучкість бази даних MySQL підтримується великою кількістю типів таблиць: користувачі можуть вибрати як таблиці MyISAM, які підтримують повнотекстовий пошук, так і таблиці InnoDB, які підтримують транзакції на рівні одного запису. Крім того, MySQL постачається зі спеціальним типом таблиці EXAMPLE, який демонструє правила створення нових типів таблиць. Завдяки відкритій архітектурі та ліцензіям GPL в базі даних MySQL постійно з’являються нові типи таблиць. MySQL характеризується високою швидкістю, стабільністю та простотою використання. MySQL безкоштовний для некомерційного використання. Можливості сервера:

— простота монтажу та використання;

— підтримується необмежена кількість користувачів, які одночасно працюють з базою даних;

— кількість рядків у таблицях може досягати 50 мільйонів;

— висока швидкість виконання команди;

— проста та ефективна система безпеки.

PostgreSQL — це об'єктно-реляційна система керування базами даних. Це альтернатива як комерційним базам даних (Oracle Database, Microsoft SQL Server, IBM DB2 та інші), так і відкритим базам даних (MySQL, Firebird, SQLite). У порівнянні з іншими проектами з відкритим кодом, такими як Apache, FreeBSD або MySQL, PostgreSQL не контролюється однією компанією, його можна розробити завдяки співпраці багатьох людей і

компаній, які хочуть використовувати цю базу даних і впроваджувати новітні рішення.

СУБД Oracle — це найпотужніший програмний пакет, який дозволяє створювати програми будь-якої складності. Ядром цього комплексу є база даних, що зберігає інформацію, обсяг якої, завдяки наданим інструментам масштабування, практично необмежений. Практично будь-яка кількість користувачів може одночасно працювати з цією інформацією з високою ефективністю (за наявності достатніх апаратних ресурсів), не виявляючи тенденції до зниження продуктивності системи при різкому збільшенні їх кількості.

Механізми масштабування в останній версії СУБД Oracle дозволяють нескінченно збільшувати потужність і швидкість сервера Oracle і його програм, просто додаючи нові й нові вузли до кластера. Він не вимагає зупинки запущених додатків, не вимагає переписування старих програм, розроблених для звичайної одномашинної архітектури. Крім того, вихід з ладу окремих вузлів в кластері також не зупиняє роботу програми. Включення JavaVM в Oracle, повна підтримка серверних технологій (сторінки сервера Java, сервлети Java, Enterprise JavaBeans, інтерфейси програм CORBA) зробили Oracle де-факто стандартом для Інтернету сьогодні.

Інша частина успіху Oracle полягає в тому, що він доступний практично з усіма існуючими операційними системами. Робота під Sun Solaris, Linux, Windows або будь-якою іншою операційною системою з продуктами Oracle не викличе жодних проблем. СУБД Oracle однаково добре працює на будь-якій платформі. Таким чином компанії, які починають працювати з продуктами Oracle, не повинні змінювати своє мережеве середовище. При роботі з СУБД існує лише невелика кількість відмінностей, обумовлених специфікою конкретної операційної системи. Загалом, це завжди та сама безпечна, надійна та зручна база даних Oracle.

Microsoft SQL Server — це система керування реляційними базами даних, розроблена компанією Microsoft. Основною мовою запитів є Transact-SQL, розроблена спільно Microsoft і Sybase. Transact-SQL — це реалізація

стандарту ANSI/ISO для мови структурованих запитів (SQL) з розширеннями. Використовується для роботи з базами даних різного розміру, від персональних до великих корпоративних баз даних, він конкурує з іншими базами даних у цьому сегменті ринку.

Під час взаємодії з Microsoft SQL Server і Sybase ASE вони використовують протокол на рівні програми, який називається табличний потік даних (TDS). Протокол TDS також реалізовано в проекті FreeTDS для забезпечення взаємодії різних додатків з базами даних Microsoft SQL Server і Sybase.

Для забезпечення доступу до даних Microsoft SQL Server підтримує Open Database Connectivity (ODBC) — інтерфейс для взаємодії з СУБД. SQL Server надає можливість підключення користувачів через веб-служби, які використовують SOAP. Це дозволяє клієнтам, які не належать до Windows, підключатися до SQL Server на різних платформах.

Microsoft Office Access або просто Microsoft Access — це реляційна база даних від Microsoft. Він має широкий спектр функцій, включаючи пов’язані запити, зв’язок із зовнішніми таблицями та базами даних. Завдяки вбудованій мові VBA ви можете писати програми, які працюють з базами даних у самому Access. Основні компоненти MS Access:

— конструктор столів;

— конструктор екранних форм;

— Майстер SQL-запитів (мова SQL в MS Access не відповідає стандарту ANSI);

— конструктор звітів для друку.

Вони можуть виконувати сценарії у VBA, тому MS Access дозволяє створювати програми та бази даних з нуля або писати зовнішню оболонку бази даних. MS Access є файловим сервером СУБД і тому застосовний лише до невеликих програм. У багатокористувацьких базах даних не так багато механізмів, як тригери.

Він суттєво розширює можливості MS Access, написавши механізм зв’язку програми з різними зовнішніми СУБД — «пов’язані таблиці» (зв’язок

з таблицею бази даних) і «запити на сервер» (запит на діалекті SQL, який «розуміє» базу даних). MS Access також дозволяє створювати повноцінні клієнт-серверні програми на базі даних MS SQL Server. Це можна поєднати з притаманною простотою інструментів управління та розробки баз даних MS Access.

1.6 Застосування ORM(Object-relational mapping)

Об’єктне реляційне відображення (ORM) — це бібліотека мови програмування, яка відображає об’єкти реляційної моделі (зв’язки, рядки та атрибути) на об’єкти мови програмування (класи, екземпляри, методи, атрибути). Таблиця (зв'язки, включаючи віртуальні таблиці — JOIN, VIEW) зазвичай відповідає класу, рядки таблиці — екземпляри цього класу, стовпці таблиці («реляційні атрибути») відображаються в атрибутах об'єкта або викликах методів читання/запису. Це відображення зазвичай є двонаправленим: маніпулювання атрибутами об’єкта призводить до читання інформації з відповідних таблиць бази даних (запису до них).

У світі зберігання даних домінують реляційні бази даних, тоді як у світі обробки даних домінує об’єктно-орієнтований підхід до проектування та програмування. Об'єктна та реляційна моделі є ортогональними. Це означає, що вони моделюють те саме існування, але під різними кутами, під різними, я б сказав, перпендикулярними кутами. Реляційна модель орієнтується на структуру і відносини істот, об'єкт — на їх властивості та поведінку [22].

Метою реляційної моделі є моделювання інформації, вибір важливих для нас атрибутів, збереження їх значень і подальший пошук, обробка та аналіз. Мета використання об'єкта — моделювання поведінки, виділення важливих для нас функцій та їх подальше використання. Відбувається перетин моделей — структурних утворень, які по-різному відображаються в цих моделях. Для того, щоб відображати артефакти реляційної моделі в артефактах одного і того ж об'єкта в наших програмах, потрібні засоби об'єктно-реляційної проекції — ORP або загальна англійська нотація — ORM (Object Relational Mapping) показано на рисунку 1.5.

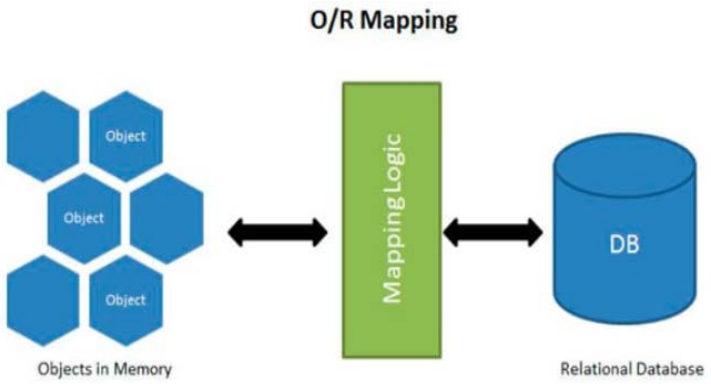

Рисунок 1.5 — ORM схема

Більш звичною мовою, об’єктно-реляційний проектор (ORP) — теоретично дозволяє розробнику працювати з таблицями, полями та зв’язками реляційної бази даних, як і з об’єктами, властивостями та колекціями (масивами), не відволікаючись на нижчі рівні. деталі, такі як порядок вибору та зберігання змінених даних, питання переносимості та властивостей діалекту SQL даної бази даних, генерування унікальних первинних ключів, заповнення полів посилання для з'єднань моделі. Вже згадані відмінності між цілями використання реляційної та об’єктно-орієнтованої моделей показують, зокрема, що якщо ваша система орієнтована на багатокритеріальний пошук та масовий пошук інформації (клас інформаційно-пошукових систем, OLAP, генерація звітів), то використання Об'єкти для доступу до даних не виправдані і просто зайві. Немає різниці між табличним представленням інформації в базі даних, у вашій програмі та на екрані користувача або у звіті, проміжна обробка зводиться до об’єднання всіх цих таблиць і простого перерахунку їх значень полів. Інша справа, якщо ваша система виконує транзакційну обробку (OLTP), складні обчислення, сповіщення про події, планування, моделювання поведінки — тут переваги використання PDO найбільші [8]. Переваги схеми БД:

— є чіткий опис схеми бази даних з точки зору мови програмування;

— вирішує принаймні наступні проблеми в нових системах, схема бази даних зазвичай уточнюється поступово в міру розвитку;

— програміст маніпулює звичайними елементами класів мови програмування, об'єктами (екземплярами класів), атрибутами та методами;

— автоматичне генерування SQL-запитів.

Об'єктно-реляційне відображення створює додатковий шар між програмою і базою даних. Цей рівень має власні API, які потрібно дослідити, що є додатковим тягарем для розробника. Цей шар створює додатковий рівень абстракції, через який іноді доводиться працювати, щоб зрозуміти, де, що і як програма працює (чи ні). Важливо розуміти, де знаходяться звичайні класи, методи та атрибути, а де ORM мають побічні ефекти (ми пишемо значення в атрибуті, і воно також зберігається в БД). Ця абстракція відображає суперечливі (у своїх діях) парадигми об'єктно-орієнтовану та реляційну. Наприклад, яка реляційна діяльність відповідає успадкуванню об’єкта? Яка об’єктно-орієнтована операція відповідає поєднанню кількох реляційних таблиць? Існують різні способи заповнити цю семантичну прогалину, але технічні рішення не усувають різницю між цими моделями. Додатковий рівень — це додатковий код, який необхідно розповсюдити разом із програмою; призводить до збільшення гучності та зниження швидкості програми.

1.7 Шаблон проектування MVC

Принцип MVC в Інтернет-програмуванні (Model View Controller) є однією з найвдаліших ідей на сьогоднішній день [7]. Принцип MVC на перший погляд інтуїтивно зрозумілий, але не дуже простий у глибині. Спочатку подумайте, для чого він призначений. Принцип MVC дозволяє розділити реалізацію логіки програми, зовнішнього вигляду (графічний інтерфейс, GUI) і взаємодію з користувачем (рисунок 1.5).

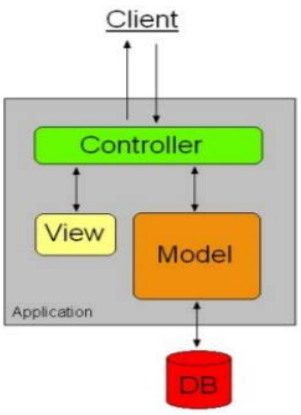

Рисунок 1.5 — Модель MVC

Це призводить до більш структурованого коду, дозволяє більш спеціалізованим людям працювати над проектом, спрощує роботу з кодом і робить його більш логічним і зрозумілим. Зміна одного компонента мінімально впливає на інші. До однієї моделі можна підключити різні типи, різні контролери. Розглянемо компоненти більш детально.

Модель — включає «бізнес-логіку» обробки та перевірки даних, доступ до баз даних, являє собою внутрішню структуру системи. Модель не повинна безпосередньо взаємодіяти з користувачем. Подання описує зовнішній вигляд програми [15]. Контролер — зв'язок між моделлю і представленням, отримує дані від користувача, відправляє їх у модель, отримує оброблений результат і передає його у представлення. Node.js framefork express надає хорошу можливість реалізувати архітектуру MVC за допомогою різних шаблонів, які підтримує ця архітектура.

1.8 Аналіз моделей прогнозування

1.8.1 Необхідність прогнозу

Чому у світі потрібне прогнозування. Відповідь полягає в тому, що всі організації працюють в умовах невизначеності, але тим не менше їх керівникам доводиться приймати рішення, які впливають на майбутнє організації. Розумні припущення щодо майбутнього більш цінні для менеджерів, ніж необґрунтовані. У цій книзі описано способи прогнозування на основі логічних методів використання даних, створених природними процесами.

Це не означає, що інтуїтивний прогноз безперечно неправильний. Навпаки, «внутрішнє» відчуття керівника підприємства часто є єдиним прийнятним прогнозом. Ми вважаємо, що ті, хто приймає рішення на основі розуміння пристроїв кількісного та якісного прогнозування та їх розумного використання, безумовно, мають перевагу перед тими, хто намагається планувати майбутнє без додаткової інформації.

Прогнозне значення оцінок суттєво змінилося за останні роки. До появи сучасного математичного механізму прогнозування та потужних комп’ютерів

оцінка менеджерів була єдиним інструментом практичного прогнозування. Завдяки роботі Макрідакіса (1986) стало зрозуміло, що оцінки з використанням лише оцінки не такі точні, як оцінки, засновані на використанні кількісних методів оцінки [8].

Оскільки світ, у якому працюють організації, завжди змінювався, завжди існувала потреба в прогнозуванні. Проте лише в останні роки зросла довіра до методів, у тому числі передових методів обробки даних. Внаслідок стрімкого розвитку нових технологій та наукових напрямків активізується діяльність органів влади на всіх рівнях, посилюється конкуренція у багатьох сферах діяльності. Рівень міжнародної торгівлі продовжує зростати майже в усіх галузях. Були створені та почали динамічно розвиватися заклади соціальної допомоги та різноманітні центри обслуговування. Інтернет став важливим джерелом фактичних даних та іншої інформації, необхідної для прийняття рішень. Все це безпрецедентно ускладнило «клімат» усіх компаній, змушуючи їх дуже швидко реагувати на зміни та залишатися конкурентоспроможними на більш високому рівні, ніж будь-коли раніше. Ті з них, хто не міг вчасно відреагувати на зміну умов і передбачити майбутнє з необхідною точністю, були приречені і перестали існувати.

Комп’ютери, поряд з кількісними методами розрахунку, які зробили їх доступними для громадськості, більше не є просто зручним інструментом для сучасних організацій, а фактично їх невід’ємною частиною. Вищезгадані проблеми сучасного життя породили величезну кількість даних, внаслідок чого виникла нагальна потреба навчитися отримувати з них різноманітну корисну інформацію. Існуючі засоби прогнозування в поєднанні з можливостями комп’ютерів стали незамінними інструментами для будь-якої організації, що працює в сучасному світі.

Кому потрібні прогнози? Практично кожна компанія, велика чи мала, приватна чи державна, використовує прогнози безпосередньо або за замовчуванням, тому що кожна компанія має планувати майбутнє, про яке вона ще нічого не знає. Крім того, потреба в прогнозах пронизує всі функціональні лінії, а також усі види організацій. Прогнози необхідні у

фінансуванні, маркетингу, підборі персоналу та різних виробничих сферах, у державних і комерційних організаціях, у невеликих соціальних клубах, у національних політичних партіях. Ось декілька прикладів запитань, які можуть вимагати відповідей на конкретні процедури прогнозування.

Які існують види прогнозів для керівника, який стикається з невизначеністю прийняття рішень? Прогнози можна розділити на довгострокові та короткострокові. Довгострокові прогнози мають важливе значення для планування основного курсу компанії в довгостроковій перспективі, і тому вони є першочерговими для вищих менеджерів. Для розробки кризових стратегій використовуються короткострокові прогнози. Найчастіше їх використовують менеджери середнього та нижчого рівня для потреб найближчого майбутнього [23].

Прогнози також можна класифікувати за їх позицією, тобто за тим, чи включають вони окремі компоненти чи узагальнені показники. Наприклад, керівник компанії може бути зацікавлений у прогнозуванні кількості працівників, які знадобляться в найближчі кілька місяців (мікропрогноз), тоді як федеральний уряд може бути зацікавлений у прогнозуванні кількості людей, зайнятих у виробництві по всій країні (макропрогноз). Крім того, менеджери різних рівнів в одній компанії зосереджуватимуть увагу на різних рівнях мікро-макроконтинууму. Наприклад, керівники вищого рівня будуть зацікавлені в прогнозуванні продажів для всієї компанії, тоді як окремі співробітники будуть набагато більше зацікавлені в прогнозуванні обсягів продажів, які вони здійснюють особисто.

Процедури прогнозування також можна розділити на кількісні та якісні. На одному полюсі є чисто якісний пристрій, який не вимагає ніяких явних операцій над математичними даними. Використовується лише «оцінка», яку надає компілятор прогнозу. Звичайно, навіть у цьому випадку «оцінка» компілятора прогнозу насправді є результатом уявного аналізу. З іншого боку, є чисто кількісний апарат, який не потребує додаткової оцінки. Це чисто механічні процедури, які дають кількісні результати. Звичайно, деякі кількісні процедури вимагають набагато більш складних методів обробки даних, ніж

інші. Ця книга зосереджена на механізмі кількісного прогнозування, оскільки широке розуміння цих дуже корисних процедур є абсолютно необхідним для ефективного управління сучасним підприємством чи організацією. Однак, як ми ще раз підкреслюємо, поряд з механічними процедурами обробки даних слід використовувати судження та здоровий глузд [8].

Усі формальні процедури прогнозування передбачають перенесення минулого досвіду в невизначене майбутнє. Тому всі вони засновані на припущенні, що умови, які призвели до отримання раніше отриманих даних, не відрізняються від майбутніх умов. Виняток становлять лише ті змінні, які точно розпізнаються прогностичною моделлю. Наприклад, якщо хтось прогнозує результативність співробітників лише на основі численних оцінок, які вони отримали в процесі найму, то, звичайно, припускається, що результативність кожного працівника залежить виключно від нього. Насправді таке припущення про нерозривність минулого і майбутнього не повністю відповідає. Отже, отриманий прогноз буде неточним, якщо він не буде модифікований на основі оцінки, зробленої автором прогнозу [46].

Розуміння того, що пристрій прогнозування працює з даними, згенерованими природними подіями, призводить до визначення наступних п’яти кроків у процесі прогнозування;

— збір даних;
— редукція або ущільнення даних;
— побудова моделі та її оцінка;
— екстраполяція обраної моделі (фактичний прогноз);
— оцінка отриманого прогнозу.

1.8.2 Метод Хельта

Експоненційні моделі згладжування та прогнозування належать до класу адаптивних прогностичних методів, головною особливістю яких є здатність постійно враховувати еволюцію динамічних характеристик досліджуваних процесів, адаптуватися до цієї динаміки, зокрема надавати більшу вагу. до поточного часу. Значення терміну полягає в тому, що адаптивне

прогнозування дозволяє оновлювати прогнози з мінімальною затримкою і з використанням відносно простих математичних процедур.

Метод експоненціального згладжування був незалежно відкритий Брауном (Brown R.G. Statistical Forecasting for Inventory Control, 1959) і Холтом (Holt C.C. Forecasting Seasonality and Trends Using Exponentially Weighted Moving Averages, 1957). Експоненціальне згладжування, як і метод ковзного середнього, використовує минулі часові ряди для прогнозування значень.

Суть методу експоненційного згладжування полягає в тому, що часові ряди згладжуються за допомогою зваженої ковзної середньої, в якій коефіцієнти ваги підпорядковуються експоненційному закону. Середньозважена змінна z. Вагові коефіцієнти з експоненційним розподілом характеризують значення процесу в кінці інтервалу згладжування, тобто є середньою характеристикою останніх рівнів ряду. Ця властивість використовується для прогнозування.

Звичайне експоненціальне згладжування використовується, коли немає даних про тенденцію чи сезонність. У цьому випадку прогнозом є середньозважене значення всіх наявних попередніх значень рядів; в той же час ваги геометрично зменшувалися в процесі просування в минулому (назад). Тому (на відміну від методу змінного середнього) немає точки, в якій ваги згортаються. Прагматично прозору модель простого експоненціального згладжування можна записати так — показуємо експоненційний характер спадних ваг значень часового ряду — від поточного до попереднього, від попереднього до передостаннього і так далі.

Якщо формула застосовується рекурсивно, кожне нове згладжене значення (яке також є прогнозом) обчислюється як середнє зважене поточного спостереження та згладженого ряду. Звичайно, результат згладжування залежить від параметра альфа-адаптації. Його можна інтерпретувати як ставку дисконту, що характеризує ступінь знецінення даних за одиницю часу. Більше того, вплив даних на прогноз експоненціально зменшується із «віком» даних.

Навіть якщо в даних спостерігається лінійна тенденція, модель Брауна може не підійти для вирішення проблеми прогнозування. Тому для врахування

впливу лінійної тенденції прийнято використовувати модель Холта [26]. цикл, тому сезонність поки не розрізняється. Якщо ряд має тенденцію до зростання або падіння, то при оцінці поточного рівня ряду (як у випадку експоненційного згладжування) тенденцію спочатку потрібно підкреслити. З цієї причини в модель було введено два коефіцієнти згладжування для контролю рівня та нахилу: ряд і коефіцієнт згладжування тренду.

Як саме розраховується прогноз Холта:

— обчисліть експоненціальне згладжування ряду;

— визначити значення тренду;

— прогнозуємо.

Розглянемо більш детально.

Розраховуємо експоненційно – згладжений ряд:

$$L_t = kY_t + (1-k)(L_{t-1} - T_{t-1})$$

де k — коефіцієнт згладжування ряду;

Y_t — поточне значення ряду;

L_{t-1} — згладжена величина за попередній період;

T_{t-1} — значення тренду за попередній період.

L_t = K (коефіцієнт згладжування ряду) * Y_t (поточне значення ряду) + (1-K) * L_{t-1} (згладжене значення за попередній період) — T_{t-1} (значення тенденції за попередній період).

Коефіцієнт згладжування ряду встановлюється від 0 до 1.

Для першого періоду на початку даних експоненціально згладжений ряд дорівнює першому значенню ряду.З’ясовуємо значення тренду.

$$T_t = \beta(L_t - L_{t-1}) + (1-\beta) * T_{t-1}$$

де T_t — значення тренду на поточний період;

β — коефіцієнт згладжування тренду;

L_t — експоненсійно згладжена величина за поточний період;

L_{t-1} — експоненційно згладжена величина за попередній період;

T_{t-1} — значення тренду за попередній період.

Tt (значення тренду на поточний період) = β(коефіцієнт згладжування тренду)* Lt (експоненсійно згладжена величина за поточний період) – $Lt{-}1$ (експоненсійно згладжена величина за попередній період) + (1-β) * $Tt{-}1$ (значення тренду за попередній період)

Коефіцієнт β знаходиться в діапазоні від 0 до 1. Значення тренду для першого періоду рівно 0. Робимо прогноз. Прогноз на n періодів вперед рівний:

$$Y^{*}= L_t + n * T_t,$$

де Y^{*}— прогноз за методом Хольта на n періодів;

L_t — експонеційно згладжена величина за попередній період;

n — порядковий номері періоду на який робимо прогноз;

T_t — тренд за попередній період.

Y^{*}(прогноз за методом Хольта на n періодів) = Lt (експонеційно згладжена величина за попередній період) + n (порядковий номері періоду на який робимо прогноз) * Tt(тренд за попередній період).

Після прогнозу необхідно перевірити точність моделі. Для цього потрібно:

— розрахувати значення прогнозованої моделі;

— визначити похибку моделі;

— розрахувати точність прогнозу;

— знайти оптимальні ряди та коефіцієнти згладжування тренду [10].

Для цієї моделі гарно підходять дані торгової групи дрова. Можна переглянути фактичні продажі для прогнозу (рисунок 1.6).

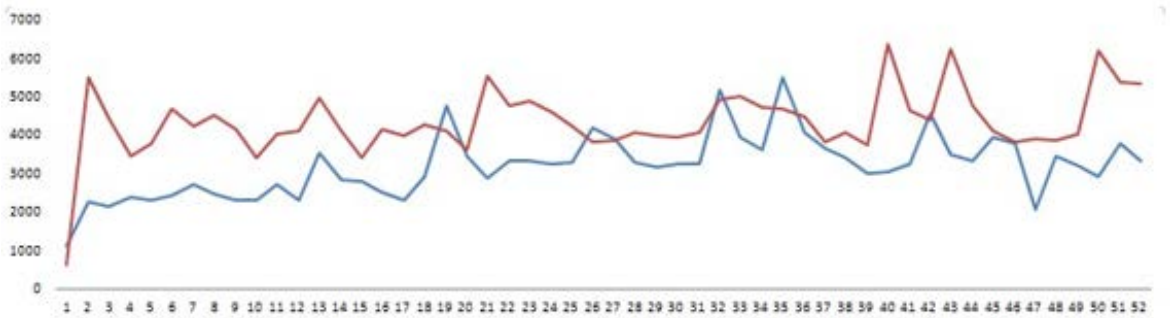

Рисунок 1.6 — Продажі вина в 2020 і 2019 роках

Ми бачимо чітку зворотну тенденцію, давайте побудуємо прогноз на основі даних, які ми маємо (рисунок 1.7).

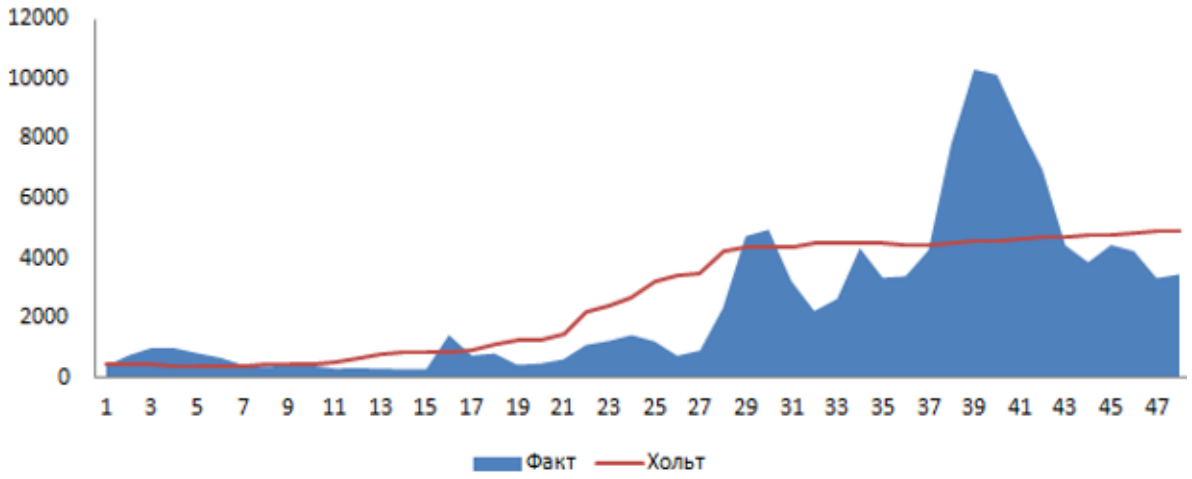

Рисунок 1.7 — Порівняння факту і прогнозу Торгової групи дрова

1.8.3 Метод Хольта-Вінтерс

Вінтерс розробив модель експоненційного згладжування з тенденцією Холта і додала до неї сезонність. Перевагою методу є можливість складання прогнозу, наприклад на 1 рік, знадобляться дані мінімум на 2 роки, а краще 3-5 років.Метод Холта-Вінтерса використовується для прогнозування часових рядів, коли в структурі даних є тенденція та сезонність. Ця модель є моделлю прогнозування з 3 параметрами, яка враховує:

— складений експоненційний ряд;

— тренд;

— сезонність.

Як розрахувати прогноз за методом Холта-Вінтерса?

Розраховуємо експоненціально згладжений ряд.

$$L_t = -t - s + (1 - \alpha)(L_{t-1} + T_{t-1})$$

Визначення значення тренду.

$$T_t = \beta(L_t - L_{t-1}) + (1 - \beta)T_{t-1}$$

Оцінюємо сезонність.

$$S_t = \gamma + (1 + \gamma)S_{t-s}$$

Робимо прогноз.

$$Y_{t+p} = (L_t + nT_t)S_{t-s+n}$$

Після прогнозу потрібно перевірити точність моделі. Для цього потрібно:

— розрахувати значення прогнозованої моделі;

— визначити похибку моделі;

— розрахувати показник точності прогнозу;

— підібрати оптимальні коефіцієнти згладжування ряду і тренду,сезонності.

Для усунення аномалій також використовується медіанний фільтр. Він вибирає в середньому 3 і залишає їх. Це необхідно, оскільки іноді трапляються аномалії, які не дозволяють нормально передбачити ряд [11]. Можна переглянути фактичні продажі для прогнозу (рисунок 1.8).

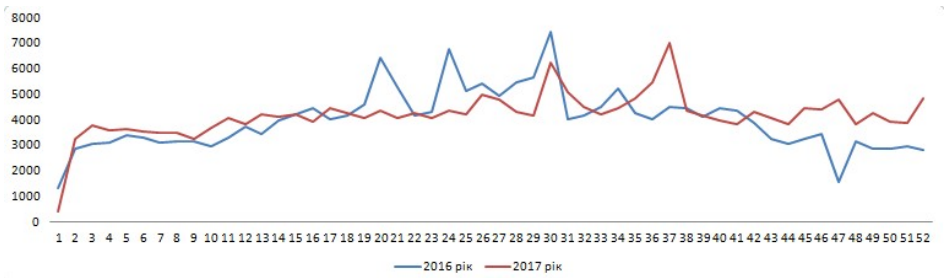

Рисунок 1.8 — Продажі за 2016 і 2017 рік консервації

Цей метод має наступні переваги. Метод Холта-Вінтерса має три компоненти — основний тренд, сезонність і експоненціально згладжений ряд. Завдяки тренду з доданим експоненційним згладжуванням можна не тільки визначити напрямок розвитку певної кількості динаміки, а й згладити невеликі коливання ряду динаміки, знайти приватні спади і стрибки. Сезонність дозволяє будувати прогноз на майбутні періоди з урахуванням цієї сезонності, що наочно показує побудований прогноз. Оскільки він враховує кілька прогностичних факторів, він буде більш точним при прогнозуванні на довгострокову перспективу. Але цей метод також має наступні обмеження.

Щоб побудувати найбільш точний прогноз, потрібні дані на тривалий час. Для побудови максимально точного прогнозу бажано використовувати дані за 4-5 періодів прогнозу. Отже, чим більш попередні дані, тим довше слід робити прогноз. Для цього методу необхідно провести додаткову оцінку точності прогнозу. Наведений вище алгоритм використовується тільки для оцінки прогнозу за допомогою цього методу, оскільки коефіцієнти a, b і q вибираються вручну і мають безпосередній вплив на поточну оцінку. Метод Холта-Вінтерса можна використовувати:

— у стратегічному плануванні: побудова основної тенденції;

— розвиток (тенденція) дозволяє врахувати зростаючу або спадну динаміку досліджуваного явища;

— в оперативно-тактичному плануванні: виділена сезонна складова дозволяє говорити про нерівномірність розподілу обсягів по місяцях щодо цієї динаміки. Це слід враховувати при плануванні, оскільки планові обсяги будуть змінюватися від місяця до місяця. Експоненціальне згладжування враховує внутрішні провали і збільшується в багатьох динаміках. Його можна використовувати для виявлення великих злетів і падінь заздалегідь і бути готовим до них. Таким чином, застосування цього методу має досить широкий діапазон. Цей метод ґрунтується на використанні великої кількості статистичних даних, які не завжди можуть бути релевантними. Метод Холта Вінтерса може бути використаний у комбінованому прогнозуванні в поєднанні з експертними методами прогнозування [12, 13], (рисунок 1.9).

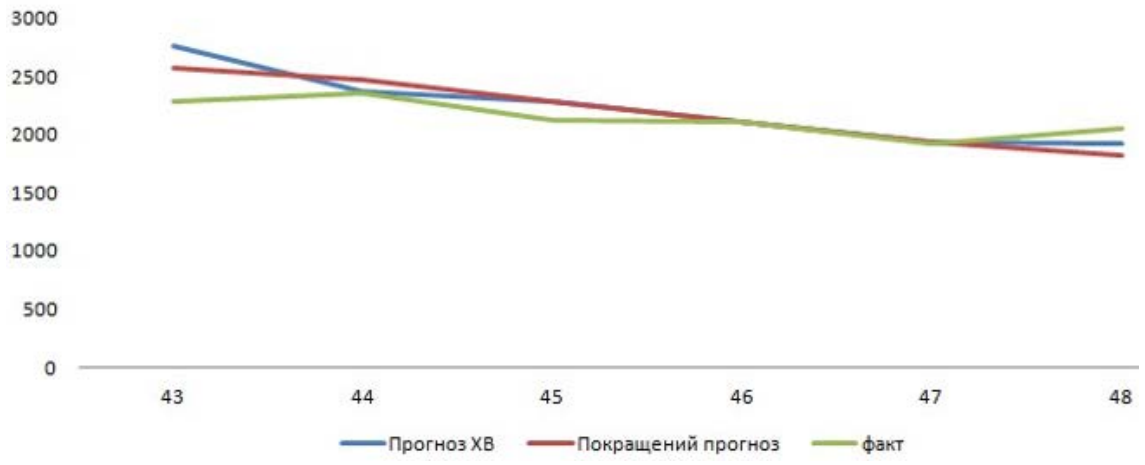

Рисунок 1.9 — Прогноз продажів разом із сезонною складовою

Покращена сезонно скоригована модель заснована на адаптації моделі до поточних продажів. Так це дозволить швидко реагувати на зміни в реалізації продукції.

1.8.4 Модель ковзаної середньої

У випадку методу простих середніх прогнозування базується на усередненні всіх наявних даних. Але іноді аналітиків більше цікавлять останні спостереження. Потім ви можете записати середню кількість точок даних і обмежитися останніми спостереженнями. Для опису такої моделі використовується термін ковзне середнє. Як тільки нове спостереження стає доступним, воно включається в усереднення, а найстаріше виключається відповідно. Щойно обчислене ковзне середнє використовується для створення прогнозу на наступний період (рисунок 1.10).

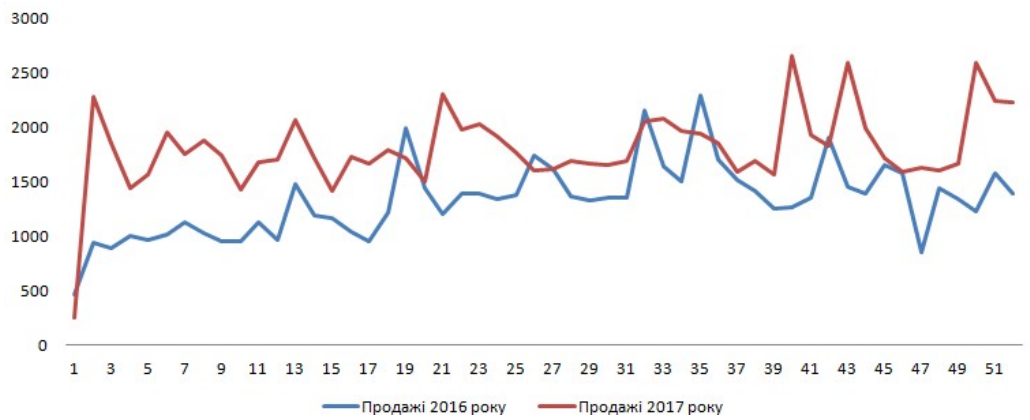

Рисунок 1.10 — Продажі 2016 і 2017 року пива темного

Методом прогнозування з використанням ковзної середньої є звичайний метод згладжування, особливо корисний для менеджерів у прогнозуванні тенденцій, коли є нерегулярні або вибіркові дані (наприклад, тенденції — сезонні або циклічні) і коли немає часу або ресурсів для розробки або застосування більш складних методів. Цей метод, як і інші методи згладжування, заснований на припущенні про наявність певних закономірностей у даних за попередні періоди. Метод «згладжує» випадкові дані, щоб відокремити шаблони від випадкового тремтіння. Хоча немає причин, чому метод ковзної середньої не слід використовувати для річних

прогнозів, він зазвичай використовується для прогнозів на набагато менший період. Частково це пояснюється тим, що ефект згладжування випадкових даних є цінним протягом тривалого періоду часу. Якщо цей метод використовується, наприклад, протягом чотирьох періодів, ефект згладжування буде недостатньо помітним, а результати відносно безплідні. Цей метод просто бере набір даних, визначає його середнє значення, а потім використовує це середнє для прогнозування наступного періоду. Це не особливо корисно для прогнозування більш ніж на один період вперед, оскільки в цьому випадку середнє значення буде взято із середнього. При використанні методу ковзної середньої бажано мати велику кількість спостережень (точок) у часі (місячні дані на рік, річні дані за багато років). Незважаючи на останню вимогу, метод ковзної середньої можна проілюструвати даними про місто Гоблер. По-перше, промоутер повинен визначити період/кількість спостережень, які будуть ковзним середнім, наприклад, 3 тижні, 3 місяці, 3 роки. Цей період залишається однаковим у всіх розрахунках, хоча вказані значення можуть змінюватися в міру додавання нових спостережень і виключення інших; звідси назва «ковзне середнє» [27]. У наведеному нижче прикладі було обрано період у 3 місяці завдяки використанню щоквартальної системи розподілу коштів у місті Гоблер. У нашому випадку середнє значення береться за 4 тижні. Метод експертної оцінки визначив, що 4 тижні є найкращим варіантом через швидку реакцію на швидкі зміни (рисунок 1.10).

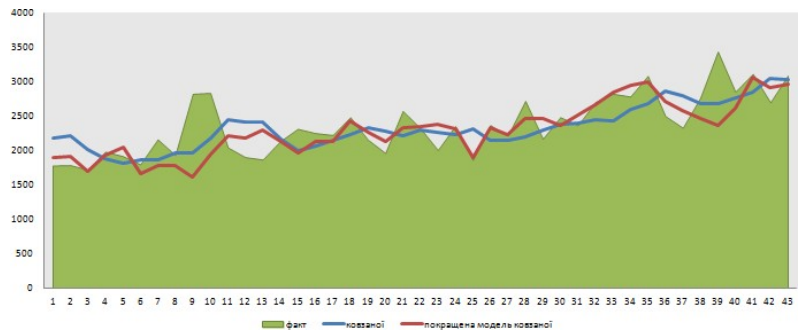

Рисунок 1.10 — Прогноз продажів пиво темне

Для оцінки прогнозу ми використовуємо стандартну помилку Маре в абсолютних значеннях.

2 РОЗРОБКА АЛГОРИТМУ ПРОГНОЗУВАННЯ ПОПИТУ

2.1 Аналіз типів прогнозування попиту

Перш за все, давайте визначимося, що таке прогнозування попиту і який вплив воно має на ваш бізнес. Прогнозування попиту — це область прогнозної аналітики, яка спрямована на розуміння та прогнозування попиту споживачів, щоб оптимізувати рішення ланцюга поставок за допомогою корпоративного ланцюга поставок та управління бізнесом.

Існує кілька типів прогнозів попиту:

— короткострокові (реалізація на короткостроковий період від 3 місяців до 12 місяців);

— середньо- та довгострокові (зазвичай на 12-24 місяці наперед (36-48 місяців у деяких компаніях), довгострокове прогнозування допомагає планувати бізнес-стратегію, планування продажів і маркетингу, фінансове планування, планування ефективності, капітальні витрати тощо);

— зовнішній макрос (прогнозування має справу з більш широкими ринковими рухами, які залежать від макроекономічного середовища, також зовнішнє прогнозування виконується для оцінки стратегічних бізнес-цілей, таких як розширення асортименту продуктів, вихід на нові сегменти клієнтів, технологічні зриви, зміна парадигми поведінки споживачів і стратегії зниження ризику);

— внутрішній бізнес-рівень (тип прогнозування застосовується до внутрішніх бізнес-операцій, таких як категорія продукту, відділ збуту, фінансовий відділ та виробнича група);

— пасивний (виконується для стабільних компаній з дуже консервативними планами розвитку);

— активний (впроваджується з метою масштабування та диверсифікації бізнесу з агресивними планами розвитку у сфері маркетингової діяльності, розширення асортименту, а також з урахуванням діяльності конкуренції та зовнішньоекономічного середовища.)

Чому прогноз попиту такий важливий і який процес покращується? Прогнозування попиту є ключовим бізнес-процесом, навколо якого розробляються стратегічні та операційні плани компанії. На основі прогнозу попиту створюються стратегічні та довгострокові бізнес-плани, такі як бюджетування, фінансове планування, плани продажів і маркетингу, планування потужностей, оцінка ризиків та плани пом’якшення. Він також впливає на наступні процеси:

— управління відносинами з постачальниками

управління відносинами з клієнтами;

— виконання замовлень та логістика;

— маркетингові кампанії;

— управління виробничим потоком.

2.2 Базовий метод прогнозування попиту

У цьому розділі я хотів би пояснити та створити основні моделі підходу.

Перший — це згладжена ковзна середня. Плавне ковзне середнє (SMMA) — це модель прогнозування попиту, яку можна використовувати для вимірювання тенденцій на основі серії середніх за певний час. Наприклад, ви можете розрахувати згладжене ковзне середнє за шість місяців продажів, взявши середні продажі з січня по червень, потім середні продажі з лютого по липень, потім з березня по серпень і так далі. Цю модель називають «мобільною», оскільки середні показники постійно перераховуються в міру появи нових даних.

Згладжене ковзне середнє корисне для перегляду загальних тенденцій продажів у часі та для довгострокового планування попиту. Швидкі зміни, викликані сезонністю чи іншими коливаннями, згладжуються, щоб ви могли детальніше проаналізувати загальну картину. Модель згладженого ковзного середнього зазвичай добре працює, якщо у вас є продукт, який продовжує зростати або падати з часом. Ще одним серйозним недоліком цього підходу є те, що ми можемо створювати об’єкти без історії.

Щоб зробити це в Python, ми можемо використовувати pandas.DataFrame.shift для створення значення Lag.

full_df ['sales_lag_n'] = full_df ['sales']. зсув (періодів = n)

Тоді ми можемо використовувати pandas.DataFrame.rolling для створення середньої бази даних на основі створених значень Lag.

full_df ['sma'] = full_df ['sales_lag_n] .rolling (n) .mean ()

Наступна модель — експоненційне згладжування Холта Вінтера. Холт (1957) і Вінтерс (1960) розширили метод Холта на сезонність. Сезонний метод Холта-Вінтерса включає прогнозні рівняння та три рівняння згладжування - одне для рівня ℓtℓt, одне для тенденції bt і одне для компонента st сезону, з відповідними параметрами згладжування αα, β∗ β∗ і γγ. Ми використовуємо мм, щоб позначити частоту сезонності, яка є кількістю сезонів у році. Наприклад, для квартальних даних m = 4m = 4 і місячних даних m = 12m = 12.

Існує два різновиди цього методу, які відрізняються, а також сезонним компонентом. Адитивний метод є кращим, коли сезонні коливання приблизно постійні в ряді, тоді як мультиплікативний метод є кращим, коли сезонні коливання пропорційні рівню ряду. При адитивному методі сезонний компонент виражається в абсолютних величинах за шкалою спостережуваного ряду, а в рівнянні рівня ряд сезонно коригується шляхом віднімання сезонної складової. Щороку сезонна складова буде приблизно нульовою. У мультиплікативному методі сезонний компонент виражається у відносних показниках (відсотках), а кількість сезонно коригується шляхом ділення на сезонний компонент. У кожному році сезонна складова становитиме приблизно мм.

Цей метод ефективніший, ніж попередній, оскільки він підтримує сезонні компоненти, але має той самий недолік, що не підтримує нові елементи. Ця модель призначена для магазину однієї пари, а це означає, що ми повинні створити нову модель для кожної пари.

```
for index, row in tqdm(df_test.iterrows()):
    tmp = df_train_aggr[(df_train_aggr['shop_id'] == row['shop_id']) & (df_train_aggr['item_id'] == row['item_id'])]
```

```
model = ExponentialSmoothing(tmp.item_cnt_day)
model_fit = model.fit()
forecast = model_fit.forecast(steps=n)
```

Остання модель у моєму базовому підході — це ARIMA. ARIMA, скорочення від «Ato-regressive Integrated Moving Average», насправді є класом моделей, які «пояснюють» часові ряди на основі їхніх власних минулих значень, тобто їхніх власних лагів і помилок відкладеного прогнозування, так що рівняння можна використовувати для передбачити майбутні цінності. За допомогою моделей ARIMA ви можете моделювати будь-які несезонні тимчасові ряди, які показують закономірності, а не є випадковим білим шумом.

Модель ARIMA характеризується 3 термінами — p, d, q:

— p — порядок терміну AR;

— q — порядок терміну MA;

— d — кількість різниць, необхідних, щоб часовий ряд був стаціонарним.

Якщо часовий ряд включає сезонні моделі, вам потрібно додати сезонні терміни, і ми отримаємо SARIMA, скорочення від «Сезонна ARIMA». Детальніше про це, коли ми закінчимо ARIMA. Як і в попередній моделі, я створю окрему модель для кожної пари магазин-товар. Тому головна ідея — знайти правильні параметри для наших моделей. Не потібно писати довгий опис того, як обчислити кожен з них, але ви можете знайти його тут. У даному випадку хотів би використовувати щось на кшталт auto.arima.

Думаю, цікава реалізація — «Pmdarima».Pmdarima переносить улюблений R auto.arim до Python, що є ще більшим аргументом, чому не потрібен R для науки про дані. pmdarima — це 100% Python + Cython і не використовує жодного коду R, але реалізовано у потужному, але легкому у використанні наборі функцій і класів, які будуть знайомі користувачам.

```
import pmdarima as pm
for index, row in tqdm(df_test.iterrows()):
```

```
model = pm.auto_arima(tmp.item_cnt_day, start_p=1, start_q=1, max_p=3, max_q=3, m=12,
start_P=0, seasonal=False, d=1, D=1, trace=False,
error_action='ignore',  # don't want to know if an order does not work
suppress_warnings=True,  # don't want convergence warnings
stepwise=True)
forecast = model_fit.predict(n_periods = n, return_conf_int=False)
```

ARIMA — досить сильна модель, яка може дати хороший прогноз. ARIMA може мати обмежену здатність передбачати екстремальні значення. Хоча модель вміло моделює сезонність і тенденції, відхилення важко передбачити для ARIMA саме тому, що вони лежать за межами загальної тенденції, відображеної в моделі.

Таким чином, класичні методи прогнозування часових рядів можуть бути зосереджені на лінійних зв’язках, однак вони є складними і можуть використовуватися для вирішення широкого кола проблем за умови, що дані підготовлені належним чином і метод добре налаштований.

2.3 Метод з використанням штучного інелекту

Найпоширенішим корпоративним використанням машинного навчання в поєднанні зі статистичними методами є прогнозна аналітика. Це дозволяє не тільки оцінити попит, а й зрозуміти, що стимулює збут і як клієнти можуть поводитися за певних умов. Основне припущення використання моделей машинного навчання для прогнозування попиту полягає у створенні багатьох корисних функцій.

Розробка функцій — це використання даних про предметні знання та створення функцій, які дозволяють точніше прогнозувати моделі машинного навчання. Це дозволяє краще зрозуміти дані та отримати більш цінну інформацію. Ця функція може бути:

— характеристики товару/магазину (інформація зі словників товарів);

— внутрішня інформація про акції та будь-які зміни цін;

— цільове кодування змінних категорій на різних рівнях;

— особливості дати.

У експериментах було використовувано такі бібліотеки Python CatBoost, XGBoost і H2O AML. Почнемо з XGBoost.

XGBoost — це оптимізована бібліотека розширеного розподіленого градієнта, розроблена, щоб бути високоефективною, гнучкою та портативною. Реалізує алгоритми машинного навчання на платформі Gradient Boosting. XGBoost не може самостійно обробляти категоріальні функції, він приймає лише числові значення, подібні до Random Forest. Тому різні кодування, такі як кодування міток, проміжне кодування або одноразове кодування, повинні бути виконані перед доставкою категорійних даних до XGBoost (рисунок 2.1).

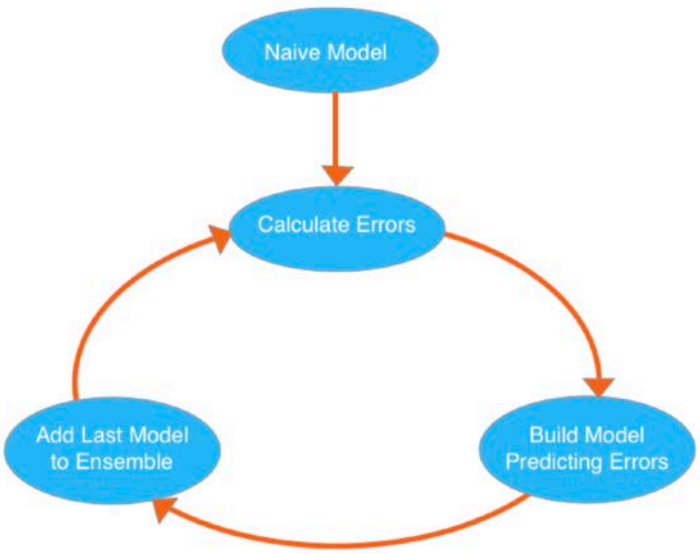

Рисунок 2.1 — Діаграма XGBoost алгоритму

XGboost розбивається до вказаного гіперпараметра max_depth, потім починає обрізати дерево назад і видаляє розриви, за межами яких немає позитивного приросту. Він використовує цей підхід, оскільки іноді за розділенням без втрат може послідувати розділ, який зменшує втрати. XGBoost також може виконувати зростання дерева на листі. Відсутні значення XGBoost будуть розміщені на стороні, яка зменшує втрати.

```
model = XGBRegressor(
   max_depth=8,
```

```
n_estimators=1000,
min_child_weight=300,
colsample_bytree=0.8,
subsample=0.8,
eta=0.3,
seed=42)
model.fit(
X_train,
Y_train,
eval_metric="rmse",
eval_set=[(X_train, Y_train), (X_valid, Y_valid)],
verbose=True,
early_stopping_rounds = 10)
```

XGBoost — це хороша і швидка реалізація алгоритму збільшення градієнта для машинного навчання, але основним недоліком, на мою думку, є те, що ви не можете використовувати категоріальні коефіцієнти, і в результаті модель може втратити частину інформації. Ще одна бібліотека — CatBoost.

Catboost росте на стійкому дереві. На кожному рівні такого дерева вибирається пара функцій розподілу, яка приносить найменші втрати (згідно з функцією штрафу), які застосовуються до всіх вузлів рівня. Його політику можна змінити за допомогою параметра grow-policy. Catboost має два режими обробки відсутніх значень — «Мін» і «Макс». У Min відсутні значення обробляються як мінімальне значення для об’єкта (вони отримують значення, менше за всі існуючі значення). Це гарантує, що розбиття вважається розділеним, відокремлюючи відсутні значення від усіх інших значень. «Макс» те саме, що і «Мін», тільки з максимальними значеннями.

Catboost використовує комбінацію одноразового кодування та розширеного кодування навпіл. Використовує одноразове кодування для функцій з невеликою кількістю категорій. Максимальною кількістю категорій для одного кодування можна керувати за допомогою параметра one_hot_max_size. Для решти стовпців категорій CatBoost використовує

ефективний метод кодування, подібний до середнього кодування, але з додатковим механізмом для зменшення перетворення. Використання категорійного кодування CatBoost має недоліки більш повільної моделі.

CatBoost надає корисні інструменти для легкої роботи з високоякісними даними. Показує тверді результати навчання за суворими категоріальними критеріями. Порівняння підходів можна переглянути на рисунку 2.2.

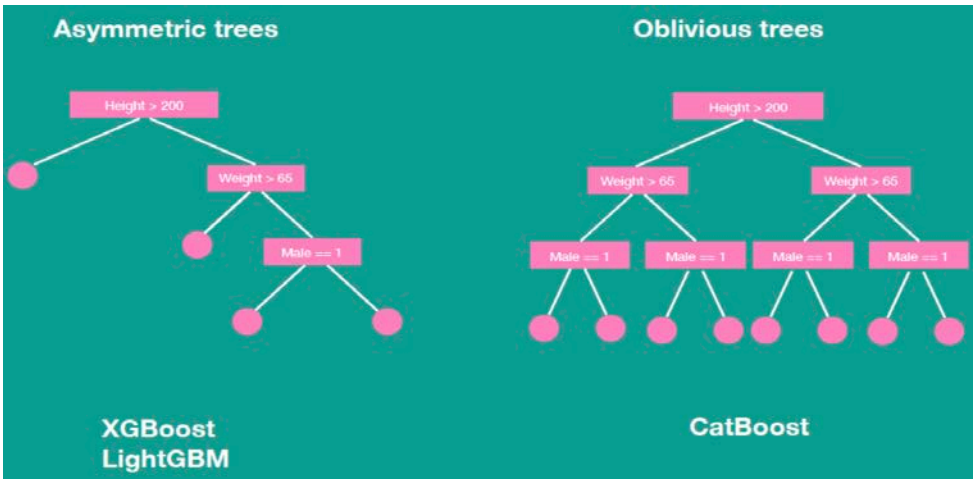

Рисунок 2.2 — Порівння XGBoost та CatBoost

H2O AutoML можна використовувати для автоматизації робочого процесу машинного навчання, який включає автоматичне навчання та конфігурацію кількох моделей у визначений користувачем час. Складні збірки — одна заснована на всіх попередньо навчених моделях і одна на основі найкращої моделі в кожній родині — автоматично навчатимуться на основі окремих колекцій моделей, щоб створювати високопередбачувані моделі складання, які в більшості випадків будуть найефективнішими моделями в таблиці лідерів AutoML.

AutoML перебирає різні моделі та параметри, щоб спробувати знайти найкращу. Потрібно вказати кілька параметрів, але в більшості випадків потрібно встановити лише максимальний час виконання в секундах або максимальну кількість моделей. Ви можете думати про AutoML як про щось на зразок GridSearch, але на рівні моделі, а не на рівні параметрів.

```
aml = H2OAutoML(max_models = 5, seed=1)
aml.train(y=y, training_frame=x_train_hf,
```

validation_frame = x_valid_hf)

AutoML призначений для автоматизації повторюваних завдань, таких як створення конвеєра та налаштування гіперпараметрів, щоб аналітики даних могли витрачати більше часу на вирішення бізнес-задач. AutoML також намагається зробити технологію доступною для всіх. AutoML і науковці з даних можуть працювати разом, щоб прискорити процес машинного навчання, щоб ви могли скористатися перевагами справжньої продуктивності машинного навчання. У даному випадку в мене найкращий результат серед попередньої моделі.

2.4 Метод з глибоким зануренням

Потужний клас алгоритмів машинного навчання, які використовують штучні нейронні мережі для розуміння та використання шаблонів у даних.

Алгоритми глибокого навчання використовують кілька шарів для поступового вилучення функцій вищого рівня із необроблених даних, що зменшує обсяг вилучення ознак, необхідних для інших методів машинного навчання (рисунок 2.3).

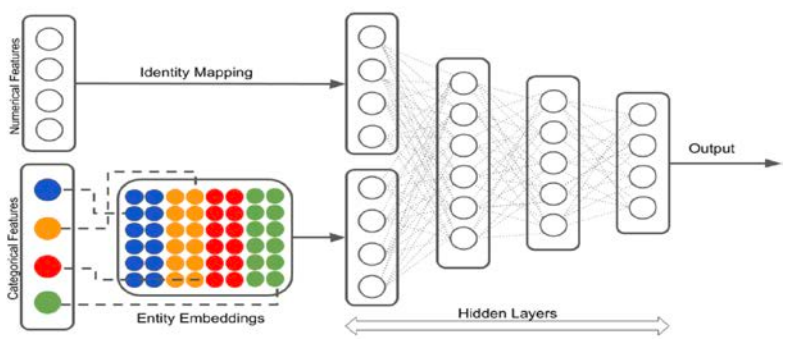

Рисунок 2.3 — Алгоритм глибокого навчання

Алгоритм глибокого навчання навчається сам, розпізнаючи шаблони на кількох рівнях обробки. Тому «глибинне» в «глибокому навчанні» відноситься до кількості шарів, через які дані перетворюються. Кілька перетворень автоматично витягують важливі функції з необроблених даних. Основним завданням у цьому завданні є обробка категоріальних змінних. Ми можемо використовувати вбудовування сутностей у глибоке навчання. Вбудовування

— це рішення для роботи з категоріальними змінними, яке дозволяє уникнути багатьох підводних каменів одноразового кодування. Вбудовування — це відображення змінної категорії в n-вимірний вектор. Отже, як виглядає архітектура нашої нейронної мережі?

Першим фреймворком Python для цього завдання є Keras. Основною проблемою тут є написання коду для вбудовування кожної категоріальної функції. Реалізація цього підходу в Kerasom досить клопітка. Усі функції машинного навчання та підхід до вбудовування одиниць працювали трохи краще, ніж попередні моделі, але на вивчення знадобилося більше часу. Перевага використання плагінів полягає в тому, що їх можна досліджувати, представляючи кожну категорію краще, ніж інші моделі. Отже, ми бачимо, що цей підхід хороший, але основним недоліком є велика кількість коду.

Fastai — наше рішення. Fast.ai — це популярне поглиблене навчання, яке надає компоненти високого рівня для досягнення найсучасніших результатів у стандартних областях поглибленого навчання. Fast.ai дозволяє практикам експериментувати, змішувати та поєднувати, щоб відкривати нові підходи. Коротше кажучи, щоб полегшити безперебійні рішення глибокого навчання. Бібліотеки використовують переваги динаміки основної мови Python і гнучкості бібліотеки PyTorch.

Глибоке навчання нейронних мереж (DNN) є складною глобальною проблемою оптимізації. Швидкість навчання (LR) є важливим гіперпараметром для налаштування під час навчання DNN. Дуже низькі швидкості навчання можуть призвести до дуже повільного навчання, тоді як дуже високі швидкості навчання можуть перешкоджати конвергенції, оскільки функція втрат коливається навколо мінімуму або навіть поширюється.

У цій структурі Fastai впровадив політику одного циклу. Суперконвергенція використовує метод CLR, але лише з одним циклом, який включає два кроки швидкості навчання, один з яких зростає, а інший зменшується, і з дуже обмеженою максимальною швидкістю навчання. Розмір циклу повинен бути меншим за загальну кількість ітерацій/епох. В кінці циклу швидкість навчання повинна знизитися ще більше для решти ітерацій/епох.

```
#TabularList for Validation
val = (TabularList.from_df(X_train.iloc[start_indx:end_indx].copy(),
path=path, cat_names=cat_feature, cont_names=con_feature))
test = (TabularList.from_df(X_test, path=path, cat_names=cat_feature,
cont_names=con_feature, procs=procs))
#TabularList for training
data = (TabularList.from_df(X_train, path=path, cat_names=cat_feature,
cont_names=con_feature, procs=procs)
.split_by_idx(list(range(start_indx,end_indx)))
.label_from_df(cols=dep_var)
.add_test(test)
.databunch())
#Initializing the network
learn = tabular_learner(data, layers=[1024,512], metrics= [rmse,r2_score])
#Exploring the learning rates
learn.lr_find()
learn.recorder.plot()
# Learn
learn.fit_one_cycle(10, 1e-02)
```

В результаті ми отримали менше коду і швидший спосіб знайти оптимальну швидкість навчання. Результат дуже схожий на попередню архітектуру нейронної мережі.

Ця архітектура працює добре, але що робити, якщо нам потрібна інформація про попередні періоди продажів без додавання функцій відставання. Тому нам потрібно додати шар LSTM або RNN до нашої архітектури. У Keras це зробить код ще більш громіздким, і для Fastai немає реалізації. Рішення цієї проблеми знайдено — прогнозування PyTorch. Прогнозування Pytorch розроблено, щоб полегшити прогнозування найсучасніших часових рядів з використанням нейронних мереж як у реальному світі, так і в тематичних дослідженнях. Мета — надати високорівневий API з максимальною гнучкістю для професіоналів і

розумними параметрами за замовчуванням для початківців. Зокрема, пакет передбачає:

— клас набору даних часового ряду, який абстрагує обробку перетворень змінних, відсутніх значень, випадкових підвибірок, кількох довжин історії тощо;

— базовий клас моделей, який забезпечує базову підготовку моделей часових рядів, а також реєстрацію тензорних таблиць і загальних візуалізацій, таких як реальні та прогнозовані графіки та графіки залежностей;

— кілька архітектур нейронних мереж для передбачення часових рядів, які були вдосконалені для реалізації в реальному світі та мають вбудовані можливості інтерпретації;

— багатогоризонтні показники часових рядів;

— оптимізатор ranger для швидшого вивчення моделі;

— налаштуйте гіперпараметри за допомогою optna;

— пакет побудовано на pytorch lightning, щоб дозволити навчатися ЦП, одному та кільком GPU з коробки.

```
early_stop_callback = EarlyStopping(monitor="val_loss", min_delta=1e-4, patience=10, verbose=False, mode="min")
lr_logger = LearningRateMonitor()  # log the learning rate
logger = TensorBoardLogger("lightning_logs")  # logging results to a tensorboard
trainer = pl.Trainer(
max_epochs=30,
gpus=1,
weights_summary="top",
gradient_clip_val=0.1,
limit_train_batches=30,
callbacks=[lr_logger, early_stop_callback],
logger=logger,)
tft = TemporalFusionTransformer.from_dataset(
training,
```

```
learning_rate=0.03,
hidden_size=16,
attention_head_size=4,
dropout=0.1,
hidden_continuous_size=8,
output_size=1,
loss=RMSE(),
log_interval=10,
reduce_on_plateau_patience=4)
print(f"Number of parameters in network: {tft.size()/1e3:.1f}k")
trainer.fit(
tft,
train_dataloader=train_dataloader,
val_dataloaders=val_dataloader
```

Стек, або узагальнення стека, — це набір алгоритмів машинного навчання. Він використовує алгоритм мета-навчання, щоб з'ясувати, як найкраще поєднувати прогнози з двох або більше основних алгоритмів машинного навчання. Ми могли б використовувати Stacking, щоб об'єднати кілька моделей і створити нові прогнози. Архітектура моделі стека включає дві або більше базових моделей, які часто називають моделями рівня 0, і мета-модель, яка поєднує передбачення базової моделі, яка називається моделями рівня 1. Моделі рівня 0 (базові моделі) — моделі відповідають навчальним даним і прогнозуються. Модель 1-го рівня (мета-модель) — модель, яка вчиться найкраще поєднувати передбачення базових моделей.

Мета-модель навчається на основі прогнозів, зроблених базовими моделями на даних поза вибіркою. Це означає, що дані, які не використовуються для навчання базових моделей, вводяться в базові моделі, робляться прогнози, і ці передбачення разом з очікуваними результатами забезпечують вхідні та вихідні пари навчального набору даних, що використовуються для відповідності мета-модель. Результатами базових моделей, які використовуються як вхідні дані для мета-моделі, можуть бути

фактичні значення регресії та значення ймовірності, значення, подібні до ймовірності, або мітки класів для класифікації. Для стекування ми можемо використовувати sklearn.ensemble.StackingRegressor.

Стекова регресія — це метод спільного навчання для поєднання кількох моделей регресії за допомогою мета-регресора. Індивідуальні регресійні моделі викладаються з повного навчального набору; потім мета-регресор визначається з необроблених даних — метафункцій — індивідуальних моделей регресії в ансамблі.

3 РОЗРОБКА ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ

3.1 Структура проекту

Розглянемо структуру проекту, цей проект був написаний за допомогою шаблону MVC. Для реалізації цього шаблону ми використовували фреймворк express.js, який покращує написання серверного коду в node.js, ми також використовували шаблон ESX, який працює з express.js для написання клієнтської частини. Структура проекту показана на рисунку 3.1.

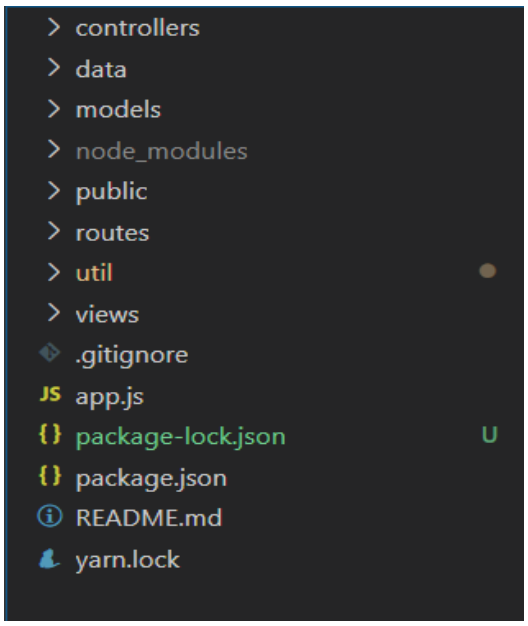

Рисунок 3.1 — Структура проекту

Основними частинами конструкції є контролери, моделі та види, ми розглянемо кожну з них:

— перегляди містять усі необхідні для клієнта елементи;

— models відповідає за створення моделей сутностей, які будуть зберігатися в базі даних;

— контролери містять основну бізнес-логіку проекту, вони реалізують логіку для кожної кінцевої точки нашого проекту;

— маршрути містять описи всіх кінцевих точок цього проекту;

— загальнодоступні магазини, що підтримують css і js код для клієнтської частини;

— utils містить код підтримки для серверної частини;

— дані містять допоміжні дані конфігурації;

— node_modules містить код допоміжних бібліотек, які використовуються в проекті (рисунок 3.2).

Рисунок 3.2 — Допоміжні залежності проекту

DevDependencies використовуються для локальної розробки проектів, вони не будуть входити в основну збірку, яка буде використовуватися для завантаження продукту на хостинг.

Залежності — використовуються для розробки проекту і входять до остаточної збірки проекту, розглянемо більш детально найосновніші з них:

— body-parser використовується для перетворення даних, які ми отримуємо від клієнтської частини, у нормалізовані дані JSON, які ми можемо використовувати в серверній частині на свій розсуд;

— шаблон ejs, який ми використовуємо для написання клієнтської частини;

— експрес-фреймворк для написання сервера node.js, завдяки цій бібліотеці обсяг серверного коду значно зменшується в порівнянні з чистим node.js;

— стиснення використовується для зменшення розміру команди проекту;

— express-session використовується для зберігання сеансів користувача;

— експрес-валідатор використовується для перевірки коректності запитів клієнта;

— nodemailer використовується для відправки інформації на пошту окремих користувачів;

— pdfkid використовується для створення pdf-документа;

— pg допоміжна бібліотека для підключення серверної частини до бази даних PostgreSQL;

— sequelize ORM, що використовується для взаємодії з базами даних SQL (у нашому випадку PostgreSQL);

— додаткова смуга послуг для створення та відстеження тестових платежів за споживчі товари.

3.2 Побудова серверної частини веб додатку

Серверну частину програми було вирішено побудувати в Node.js. Мова добре розвинена і має чудову документацію, на якій можна писати дійсно складні та стабільні програми. Широке застосування для розробки веб-додатків на корпоративному рівні став фреймворк під назвою Express, він має досить велике співтовариство і з самого початку постійно розвивався. Створення проектів на node.js зазвичай займає багато часу через деяку надмірність, тому в цьому проекті я використовую підхід експрес і MVC для прискорення початкової розробки, потім процес менеджера PM2 можна використовувати для розширення проекту, що дозволяє створювати кілька серверних об’єктів. наш продукт стане популярнішим, а навантаження на сервер зростатиме, ви також можете скористатися перевагами багатьох провідних речей, таких як Docker, Kubernetes і Nginx, щоб створити кластер і горизонтально масштабувати свій проект.

Оскільки наш проект використовує підхід MVC, нам спочатку потрібно визначити об’єкти, які з’являться в цьому проекті, щоб мати можливість написати бізнес-логіку пізніше, тобто реалізувати рівень моделі програми.

Потім нам потрібно написати контролери для отримання запитів клієнтів і надсилання відповідей, кожен з наших контролерів буде підключений до експрес-маршрутизатора, який визначає всі кінцеві точки нашого дизайну.

Основні налаштування нашого сервера будуть розташовані у файлі app.js включаючи основні залежності файлу об'єднані для налаштування структури проекту. З датою випуску нової версії JS під назвою ES6 ми можемо розбити наші файли на модульну структуру, яка використовує шаблон проекту під капотом, за допомогою цієї функції ми можемо зробити наш код чистішим і легше мати справу з іншими розробниками. проект.

Якщо необхідно додати файли з іншими розширеннями (* .sass, * .less тощо) до основної колекції проекту, ми можемо підключити Webpack, він містить додаткові плагіни, за допомогою яких ми можемо налаштувати підключення файлів з різними розширеннями до нашої основної колекції.

3.3 Визначення сутностей

Щоб обмежити доступ до сервісу, нам спочатку потрібно якось розрізнити користувачів, для цього нам потрібно визначити модель користувача. Кінцевий результат таблиці, яка буде створена в базі даних за допомогою ORM, буде виглядати як на рисунку 3.3.

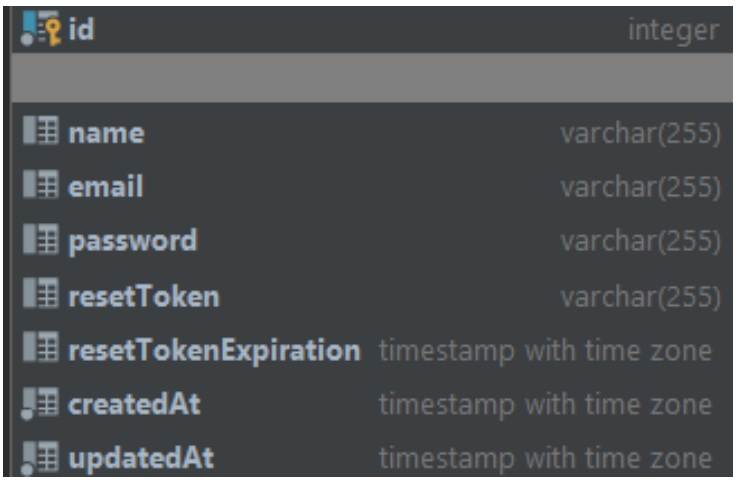

Рисунок 3.3 — Схема таблиці users

User буде мати такі основні поля:

— ім'я користувача ім'я користувача;

— електронна пошта користувача, на яку користувач буде отримувати повідомлення про товар;

— пароль унікальний пароль користувача, який використовується для аутентифікації;

— resetToken — це унікальний ключ, який користувач отримує після входу, і цей ключ використовується для авторизації користувача, оскільки цей проект має панель адміністратора, а простий користувач не має доступу до цієї панелі;

— resetTokenExpiration — час дії токена.

```
const User = sequelize.define('user', {
  id: {
    type: Sequelize.INTEGER,
    autoIncrement: true,
    allowNull: false,
    primaryKey: true
  },
  name: Sequelize.STRING,
  email: Sequelize.STRING,
  password: Sequelize.STRING,
  resetToken: Sequelize.STRING,
  resetTokenExpiration: Sequelize.DATE,
});
```

Для зображення окремого продукту ми використовуємо модель Product:

— id, унікальний ідентифікатор продукту, діє як первинний ключ, щоб пришвидшити пошук продукту в цьому полі, оскільки це поле за замовчуванням є індексом;

— назва товару;

— ціна ціна товару;

— imageUrl посилання на зображення продукту;

— опис товару.

```
const Product = sequelize.define('product', {
 id: {
  type: Sequelize.INTEGER,
  autoIncrement: true,
  allowNull: false,
  primaryKey: true
 },
 title: Sequelize.STRING,
 price: {
  type: Sequelize.DOUBLE,
  allowNull: false
 },
 imageUrl: {
  type: Sequelize.STRING(1024),
  allowNull: false
 },
 description: {
  type: Sequelize.STRING,
  allowNull: false
 }});
```

Для зображення всіх замовлень використовується модель Orders дана модель являється допоміжною для поєднання двох SQL таблиць User та Product:

— ідентифікатор унікального ідентифікатора замовлення;

— userId унікальний ідентифікатор користувача.

```
const Order = sequilize.define('order', {
   id: {
      type: Sequelize.INTEGER,
      autoIncrement: true,
      primaryKey: true,
      allowNull: false
```

```
    }
 })
Order.belongsTo(User)
User.hasMany(Order)
Order.belongsToMany(Product, { through:  OrderItem })
```

Модель OrderItem використовується для відображення одного замовлення:

— id унікальний ідентифікатор замовлення;

— унікальний ідентифікатор продукту productId;

— кількість продукції на замовлення;

— userId унікальний ідентифікатор користувача.

```
const OrderItem = sequilize.define('orderItem',
{
    id: {
        type: Sequelize.INTEGER,
        autoIncrement: true,
        primaryKey: true,
        allowNull: false
    },
        quantity: Sequelize.INTEGER
 })
Order.belongsToMany(Product, {
through:  OrderItem
})
```

Для зображення окремого товару в кошику використовується модель CardItem:

— id унікальний ідентифікатор замовлення;

— productId унікальний ідентифікатор продукту;

— кількість, кількість продукції на замовлення;

— cardId — це унікальний ідентифікатор кошика.

Для зображення окремого кошику використовується модель Card:

— Id унікальний ідентифікатор кошика;

— userId — це унікальний ідентифікатор користувача.

```
const Cart = sequilize.define('cart', {
    id: {
        type: Sequelize.INTEGER,
        autoIncrement: true,
        primaryKey: true,
        allowNull: false
    }
})
User.hasOne(Cart);
Cart.belongsTo(User);
```

Повна схема бази даних з зображенням всіх таблиць та зв’язків створених за допомогою ORM sequelaize зображено на рисунку 3.4

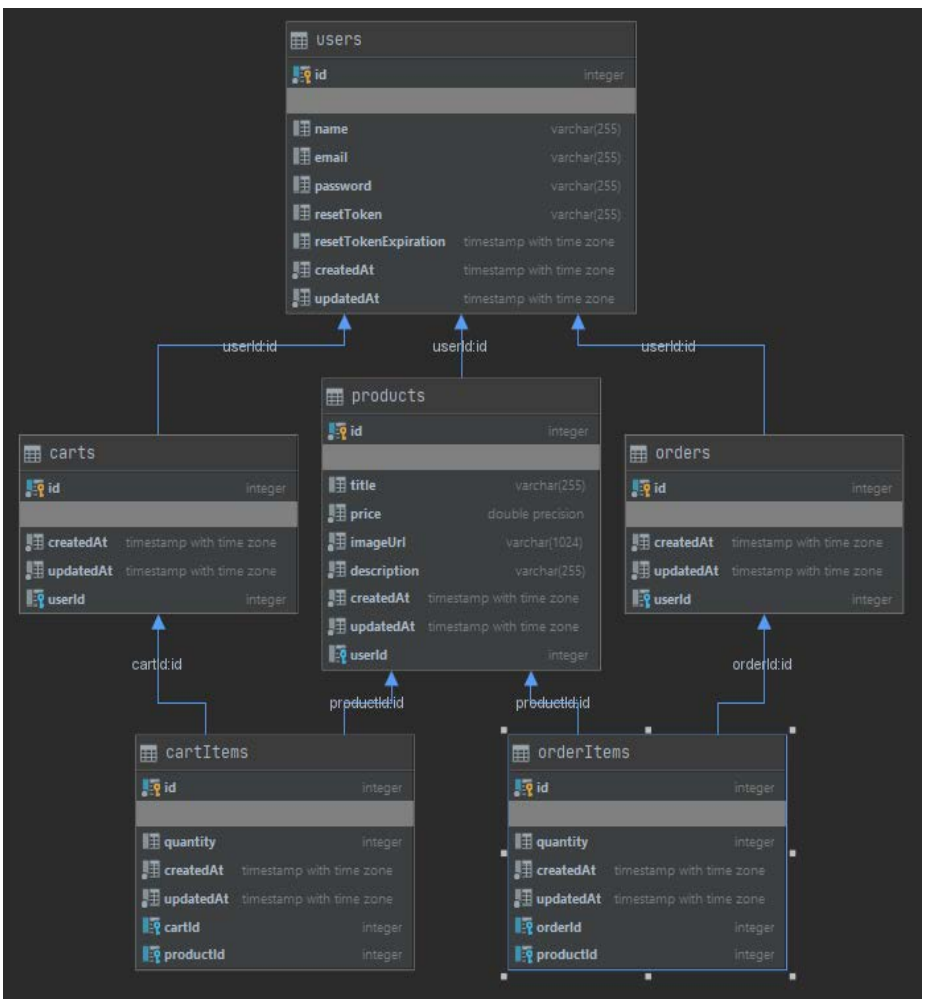

Рисунок 3.4 — Схема бази даних

3.4 Визначення кінцевих точок

Наш проект використовує express router модуль для створення кінцевих точок серверної частини. Застосування кінцевих точок в моделі MVC можна переглянути на рисунку 3.5.

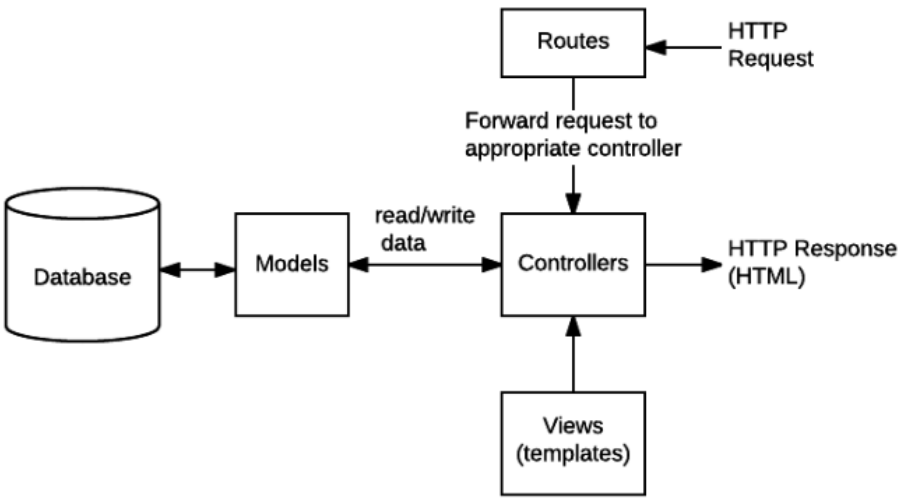

Рисунок 3.5 — Застосування кінцевих точок в моделі MVC

Кінцеві точки цього проекту розділені на дві частини, одна частина містить кінцеві точки для користувачів з правами адміністратора, інша частина для звичайних користувачів з правами доступу користувачів. Кінцеві точки для аутентифікації:

— /login (get) –- повертає сторінку HTML для аутентифікації користувача;

— /login (post) — відправляє на сервер запит з даними користувача на аутентифікацію;

— /logout (post) — надсилає на сервер запит з даними користувача на завершення сеансу користувача;

— /signup (get) — повертає сторінку HTML для створення нового користувача;

— /signup (post) — відправляє на сервер запит з даними користувача на створення нового користувача;

— /resetPassword (get) —повертає сторінку HTML для відновлення пароля користувача;

— /resetPassword (пост) — надсилає на сервер запит з даними користувача на відновлення пароля.

```
const router = express.Router()
router.get('/login', authController.getLogin)
router.post('/login',
  [
    check('email')
      .isEmail()
      .withMessage('Incorrect email')
      .normalizeEmail(),
    body('password', 'Password has to be valid.')
      .isLength({ min: 5 })
      .isAlphanumeric()
      .trim()
  ],
  authController.postLogin)
router.post('/logout', authController.postLogout)
router.get('/signup', authController.getSignUp)
router.post('/signup', authController.postSignUp)
router.get('/resetPassword', authController.getResetPassword)
router.post('/resetPassword', authController.postResetPassword)
```

Кінцеві точки для користувача що створив сесію:

— /products (get) — повертає HTML-сторінку з усіма доступними продуктами магазину;

— /products /: productId (get) — повертає HTML-сторінку з одним продуктом;

— /card (get) — повертає HTML-сторінку з кошиком користувача, який створив сеанс і додав окремі продукти в кошик;

— /card (mail) — відправляє запит на сервер створити кошик продуктів для користувача;

— /card-delete-item (post) — відправляє на сервер запит на видалення окремого товару з кошика;

— /orders (get) — повертає HTML-сторінку з усіма замовленнями користувачів;

— /orders — orderId (get) повертає HTML-сторінку з окремим порядком користувача.

```
const express = require('express');
const shopController = require('../controllers/shop');
const isAuth = require('../middlware/is-auth');
const router = express.Router();
router.get('/', shopController.getIndex);
router.get('/products', shopController.getProducts);
router.get('/products/:productId', shopController.getProduct);
router.get('/cart', isAuth, shopController.getCart);
router.post('/cart', isAuth, shopController.postCart);
router.post('/cart-delete-
item', isAuth, shopController.postCartDeleteProduct);
router.get('/orders', isAuth, shopController.getOrders);
router.get('/orders/:orderId', isAuth, shopController.getInvoice)
router.get('/checkout', isAuth, shopController.getCheckout)
module.exports = router.
```

Кінцеві точки для користувача з правами доступу admin:

— admin/products (get) — повертає HTML-сторінку з усіма доступними товарами магазину, з можливістю зміни даних про товар;

— admin/add-products (get) — повертає HTML-сторінку з можливістю додавання нового товару в базу даних магазину;

— admin/edit-product /: productId (get) — повертає HTML-сторінку з формою для зміни даних про продукт;

— admin/edit-product (post) — відправляє на сервер запит з новими даними для конкретного продукту;

— admin/delete-product (post) — надсилає запит на сервер на видалення продукту з бази даних.

```
const express = require('express');
const adminController = require('../controllers/admin');
const isAuth = require('../middlware/is-auth');
const router = express.Router();
router.get('/add-product', isAuth, adminController.getAddProduct);
router.get('/products', isAuth, adminController.getProducts);
router.post('/add-product', isAuth, adminController.postAddProduct);
router.get('/edit-
product/:productId', isAuth, adminController.getEditProduct);
router.post('/edit-product',isAuth, adminController.postEditProduct);
router.post('/delete-product', isAuth, adminController.postDeleteProduct);
router.delete('/product/:productId', isAuth, adminController.deleteProduct);
module.exports = router.
```

3.5 Визначення контролерів

Контролери — це деякі функції, які визначені для відповіді на запити клієнта, кожен контролер підключений до відповідної кінцевої точки, тому клієнт повинен запитати конкретну URL-адресу за допомогою певного методу http, і коли сервер отримає такий запит, він перевірить, чи є маршрут для обробки цієї URL-адреси, якщо такий маршрут існує, і в цьому випадку експрес викликає зазначений контролер, приєднаний до даної кінцевої точки. Показано структуру наших контролерів проекту на рисунку 3.6.

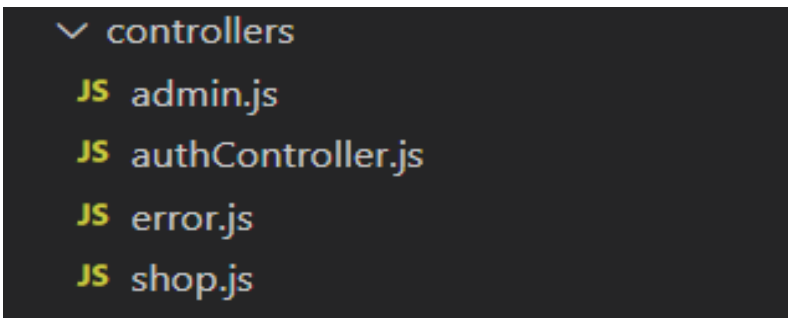

Рисунок 3.6 — Структура контролерів

Функція контролера приймає три головних параметра:

— req — дані про запит, створений з клієнтської частини, значення, що містять цей параметр, залежать від типу запиту;

— res — містить всі необхідні дані для відправки відповіді клієнту, у нашому випадку ми відправляємо готову HTML-сторінку, яку обробляємо за допомогою шаблону EJS;

— next використовується лише у функціях непрямого експрес-виклику фреймворка, щоб передати виконання запиту функції або контролеру наступного непрямого виклику.

AuthController контролер — відповідає за всі кінцеві точки що стосуються аутунтифікації та авторизації. Основні функції даного контролера:

— getLogin () — повертає HTML-сторінку для входу користувача;

— PostLogin () — приймає запит на пошту, в якому перевіряється, чи є користувач з відповідною адресою електронної пошти, при її відсутності повертається помилка, що користувач із зазначеною адресою електронної пошти не зареєстрований, якщо користувач існує, а потім паролі користувачів порівнюються за допомогою bcryptjs, який хешував пароль користувача в базі даних;

— getSignup () — повертає HTML-сторінку для реєстрації користувача;

— postSignup () — приймає запит допису, який визначає вміст з усіма необхідними полями для створення нового користувача, поля перевіряються пакетом експрес-валідатора, якщо будь-яке з полів не відповідає нашим умовам перевірки, користувач не буде створений, інакше користувач буде створений з хешованим паролем і надішле на пошту повідомлення про те, що користувач успішно зареєструвався.

— getResetPassword () — повертає HTML-сторінку з формою відновлення пароля;

— postResetPassword () — приймає запит на допис, в якому користувач надсилає свою адресу електронної пошти, на яку потрібно скинути пароль, контролер перевіряє, чи є користувач з цією адресою, якщо користувач

існує, сервер відправляє на пошту користувача і посилання з секретним маркером для скидання пароля для цього користувача.

```
exports.getSignUp = (req, res, next) => {
  res.render('auth/signup', {
    path: 'signup',
    pageTitle: 'Sign Up',
    errorMessage: req.flash('error')[0],
    oldInput: undefined,
    validationsError: []
  })
```

Контролер магазину — відповідає за всі кінцеві точки для звичайного користувача, який не має прав доступу, наприклад, адміністратора. Основні функції цього контролера:

— getProducts () — повертає HTML-сторінку з усіма доступними продуктами, на сторінці встановлюється певна кількість продуктів, щоб не обтяжувати сервер;

— getProduc t () — повертає HTML-сторінку з вибраним продуктом.

```
exports.getProduct = (req, res, next) => {
  const prodId = req.params.productId;
  Product.findById(prodId)
    .then(product => {
      res.render('shop/product-detail', {
        product: product,
        pageTitle: product.title,
        path: '/products',
      });
    })
    .catch(err => {
      const error = new Error(err)
      error.httpStatusCode = 500
```

```
      return next(error)
    })}
```

— getCard () — повертає HTML-сторінку з усіма доступними кошиками користувача;

— листівка () — використовується для додавання товару в кошик, якщо цей товар є в кошику, кількість товарів у кошику збільшується;

— postCardDeleteProduct () очищає продукти в кошику.

```
exports.postCartDeleteProduct = (req, res, next) => {
  const prodId = req.body.productId;
  req.user.deleteCartProduct(prodId)
    .then(result => {
      res.redirect('/cart')
    })
    .catch(err => {
      const error = new Error(err)
      error.httpStatusCode = 500
      return next(error)
    })
}
```

PostCreateOrder () — використовується для створення замовлення, для замовлення використовуються всі товари з кошика, перейдіть на іншу вкладку, щоб користувач міг перевірити дані.

Admin контролер — відповідає за всі кінцеві точки для користувача який немає права доступу admin Основні функції даного контролера.

getAddProduct () — повертає HTML сторінку з формою для додавання нового продукту.

```
exports.getAddProduct = (req, res, next) => {
  res.render('admin/edit-product', {
    pageTitle: 'Add Product',
    path: '/admin/add-product',
```

```
    editing: false
  });
}/
```

PostAddProduct () — використовується для створення нового продукту та вигрузки фотографію даного продукту на сервер.

```
exports.postAddProduct = (req, res, next) => {
  const title = req.body.title;
  const imageUrl = req.body.imageUrl;
  const price = req.body.price;
  const description = req.body.description;
  req.user.createProduct({
    title: title,
    price: price,
    imageUrl: imageUrl,
    description: description
  })
    .then
(result => {
      console.log('Created Product');
      res.redirect('/admin/products');
    })
};
```

GetEditProduct () — повертає HTML сторінку з формою для редагування продукту.

PostEditProduct () — використовується для редагування продукту та вигрузки нової інформації на сервер.

```
exports.postEditProduct =
(req, res, next) => {
  const prodId = req.body.productId;
  const updatedTitle = req.body.title;
```

```
  const updatedPrice = req.body.price;
  const updatedImageUrl = req.body.imageUrl;
  const updatedDesc = req.body.description;
  Product.findByPk(prodId)
    .then(product => {
      product.title = updatedTitle;
      product.price = updatedPrice;
      product.description = updatedDesc;
      product.imageUrl = updatedImageUrl;
      return product.save();
    })
    .then(result => {
      console.log('UPDATED PRODUCT!');
      res.redirect('/admin/products');
    })
    .catch(err => console.log(err));
};
```

GetProducts () — повертає HTML сторінку з усіма доступними продуктами з можливістю редагування та видалення продуктів.

```
exports.getProducts =
(req, res, next) => {
  req.user.getProducts()
    .then
(products => {
      res.render('admin/products', {
        prods: products,
        pageTitle: 'Admin Products',
        path: '/admin/products'
      });
    })
    .catch(err => console.log(err));
```

};

PostDeleteProduct () — використовується для видалення продукту.

Error контролер — несе відповідальність за обробку помилок, що виникли під час звернення до сервера або у випадку, коли на сервері виникли фатальні помилки і сервер не міг належним чином обробити запит.

3.6 Структура клієнтської частини

Модель MVC, яка широко використовується в Express для створення інтерфейсних шаблонів. Тому є деякі макети, схожі на звичайний html з додаванням спеціальних позначок, куди і як вставляти дані, зазвичай сервер вставляє дані після заміни даних, сторінка повністю відправляється клієнту. При такому підході є багато додатків і він відповідає моделі MVC, показана структура клієнтської частини на рисунку 3.7.

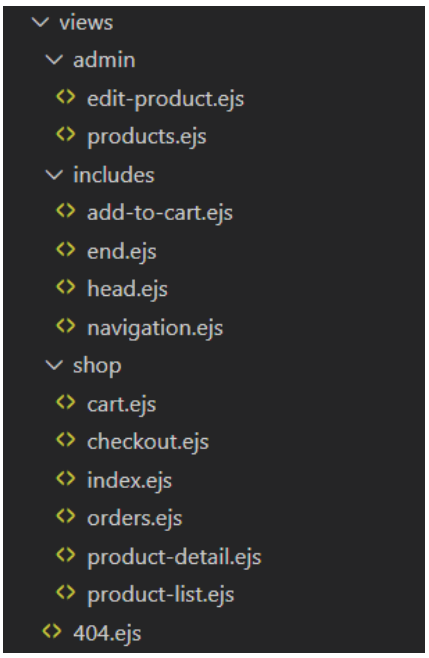

Рисунок 3.7 — Структура клієнтської частини

Однією з переваг шаблонів є те, що з їх допомогою ми можемо створювати код, який буде використовуватися багаторазово, що відповідає сучасному компонентному підходу, який використовується в найпопулярніших фреймворках, таких як Angular, Vue.js або React.js.

3.7 Інструкція з розгортання додатку

Для підготовки та розгортання додатку на сервер потрібно виконати певні кроки.

Установлення змінної середовища Node.js 12.16.3 спеціально для Windows — це PATH. Після встановлення ви можете перевірити версію Node в терміналі, командному рядку або Power Shell, виконавши команду node v результат показаний на рисунку 3.8.

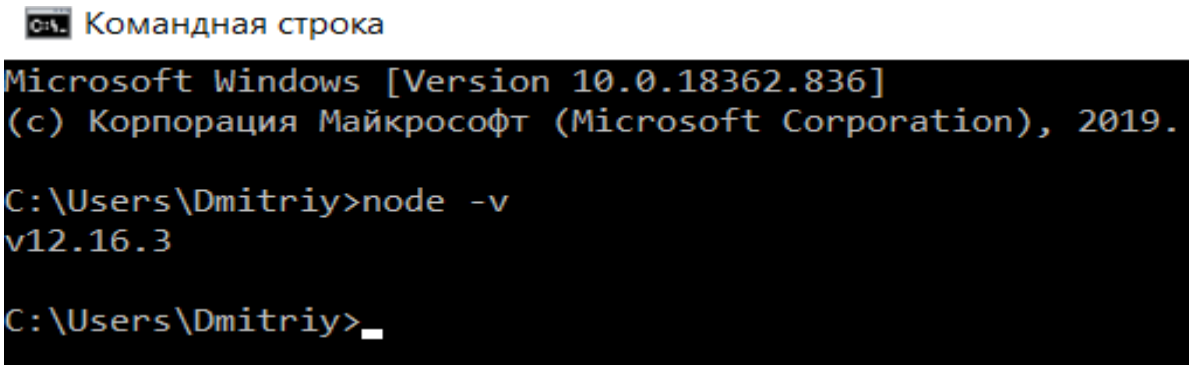

Рисунок 3.8 — Результат перевірки версії Node.js

Наступним кроком є встановлення PostgreSQL версії 9.4.5 і створення бази даних. Використовано провідну IDE під назвою DataGrip для роботи з PostgreSQL. Щоб перевірити версію PostgreSQL, потрібно запустити SELECT version () (рисунок 3.9).

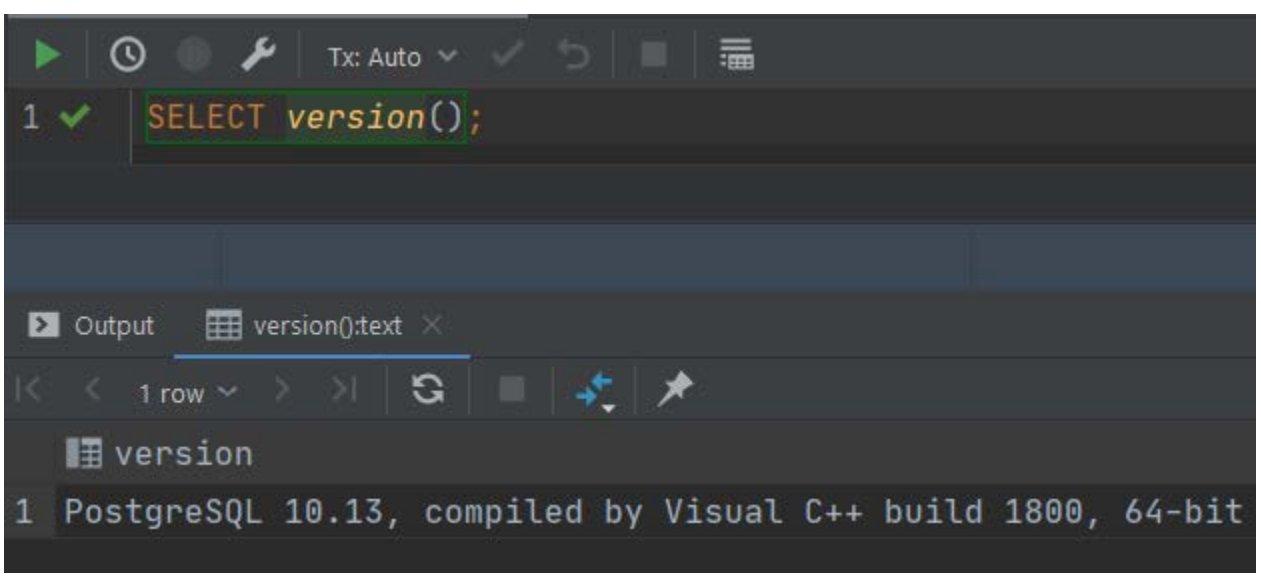

Рисунок 3.8 — Результат перевірки версії PostgreSQL

Після успішної інсталяції Node.js і PostgreSQL нам потрібно встановити всі необхідні залежності для нашого проекту, для цього ми можемо використовувати менеджер пакетів npm, який автоматично встановлюється разом з Node.js, і запустити команду npm install у каталозі проекту.

3.8 Інструкція користувача та тестування додатку

Реєстрація та авторизація. Для використання додаткових функцій веб-сайту користувачу необхідно зареєструватися та увійти під логіном та паролем, у разі використання неправильного пароля та логіна користувач отримає інструкцію, щоб зрозуміти, що сталося. Сторінка авторизації зображена на рисунку 3.10.

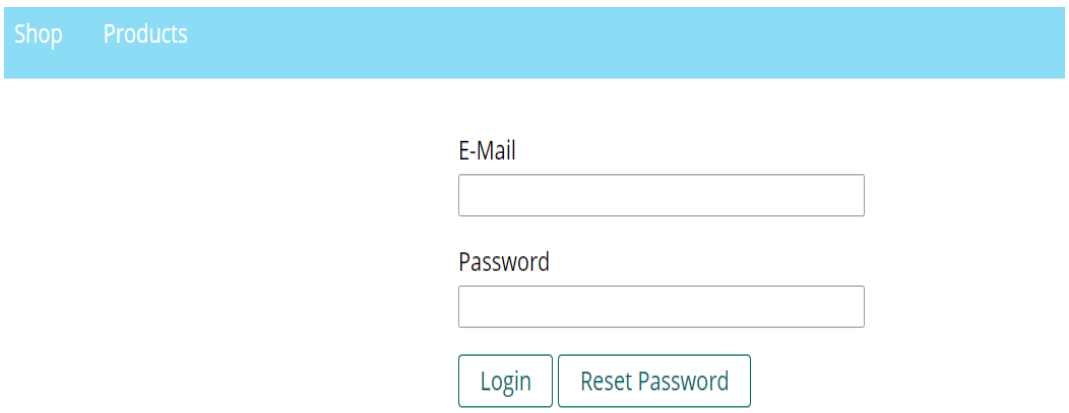

Рисунок 3.10 — Сторінка входу в систему

Якщо користувач не має попередньо створеного облікового запису, для використання сервісу натисніть кнопку реєстрації та перейдіть на сторінку реєстрації (рисунок 3.11) та зареєструйтеся.

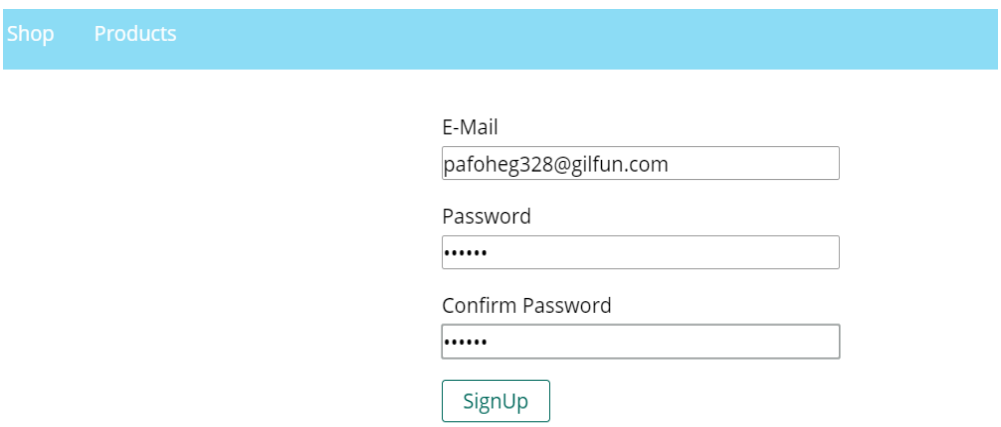

Рисунок 3.11 — Сторінка реєстрації в систему

Сторінка реєстрації також обробляє всі необхідні перевірки для перевірки правильності введених даних, після успішної реєстрації користувач отримає повідомлення на вказану адресу електронної пошти. Відображається зміст повідомлення на рисунку 3.12.

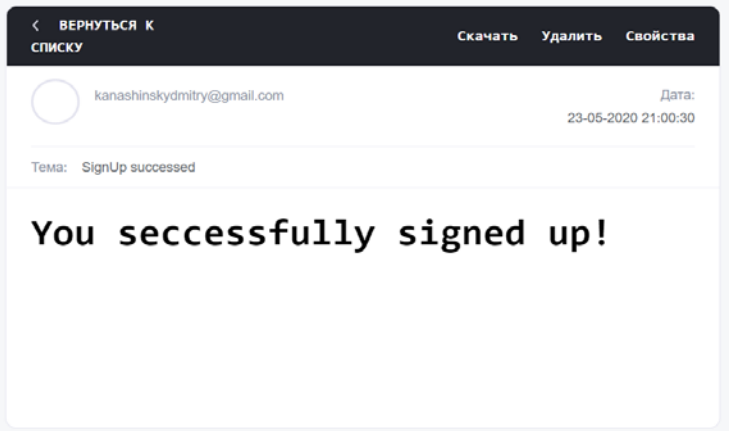

Рисунок 3.12 — Повідомлення після успішної реєстрації

На домашній сторінці звичайного користувача відображаються наступні сторінки на домашній сторінці. Магазин — відображає всі доступні продукти в магазинах на цій сторінці використовує сторінку вимірювання для вимірювання для вимірювання для вимірювання для вимірювання до виміру для зміщення (рисунок 3.13).

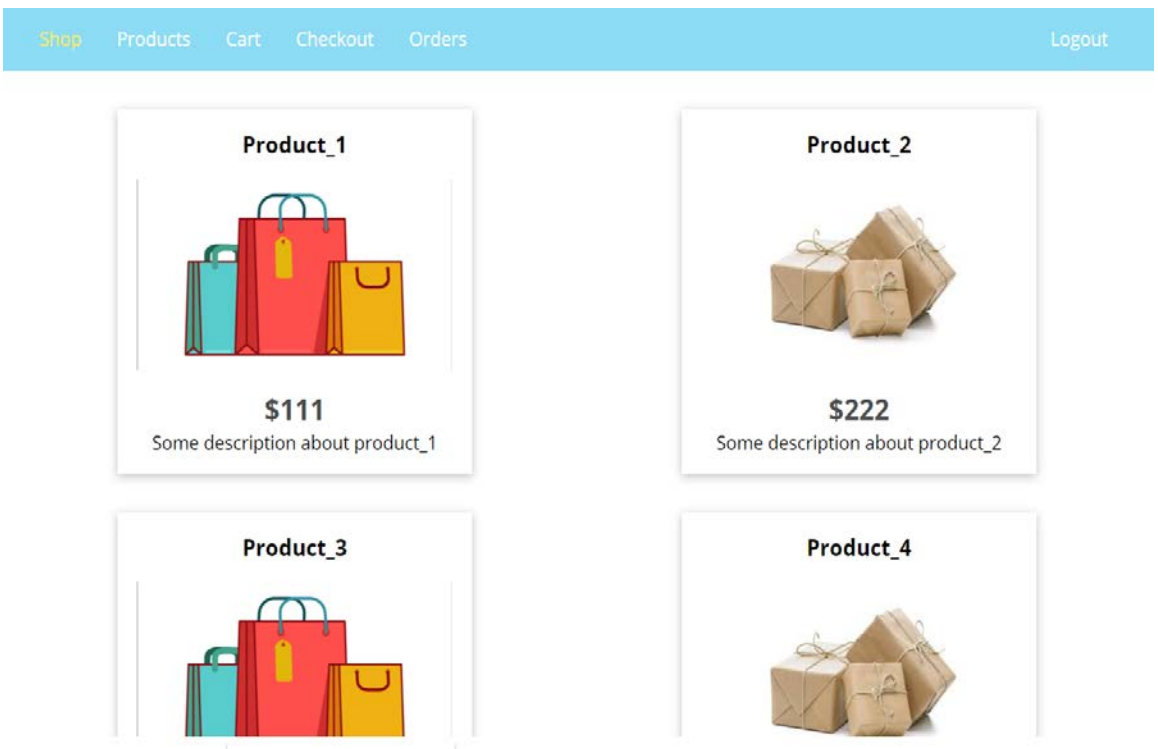

Рисунок 3.13 — Відображення всіх доступних продуктів

Products — відображення доступних продуктів з можливістю детального перегляду кожного з них та добавлення потрібних до корзини (рисунок 3.14).

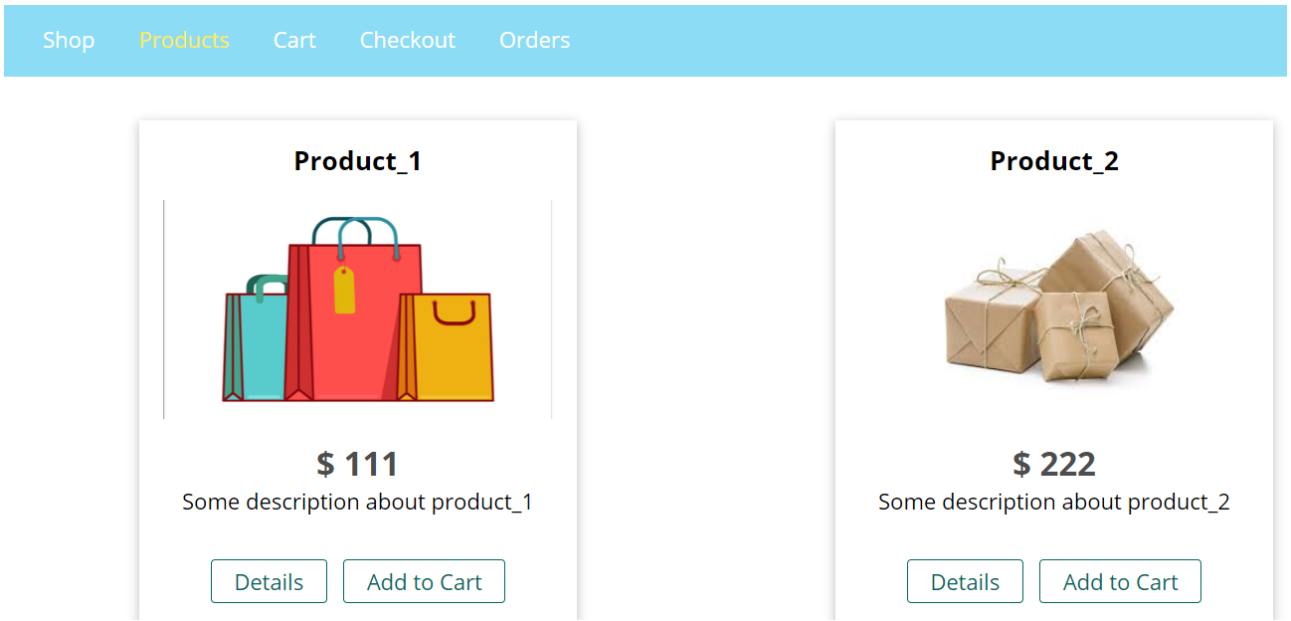

Рисунок 3.14 — Відображення вкладки Products

Cards — відображення продуктів що були додані до корзини, біля кожного продукту вказано кількість та загальну ціну продуктів, користувач має можливість видаляти продукти з корзини чи замовити їх (рисунок 3.15).

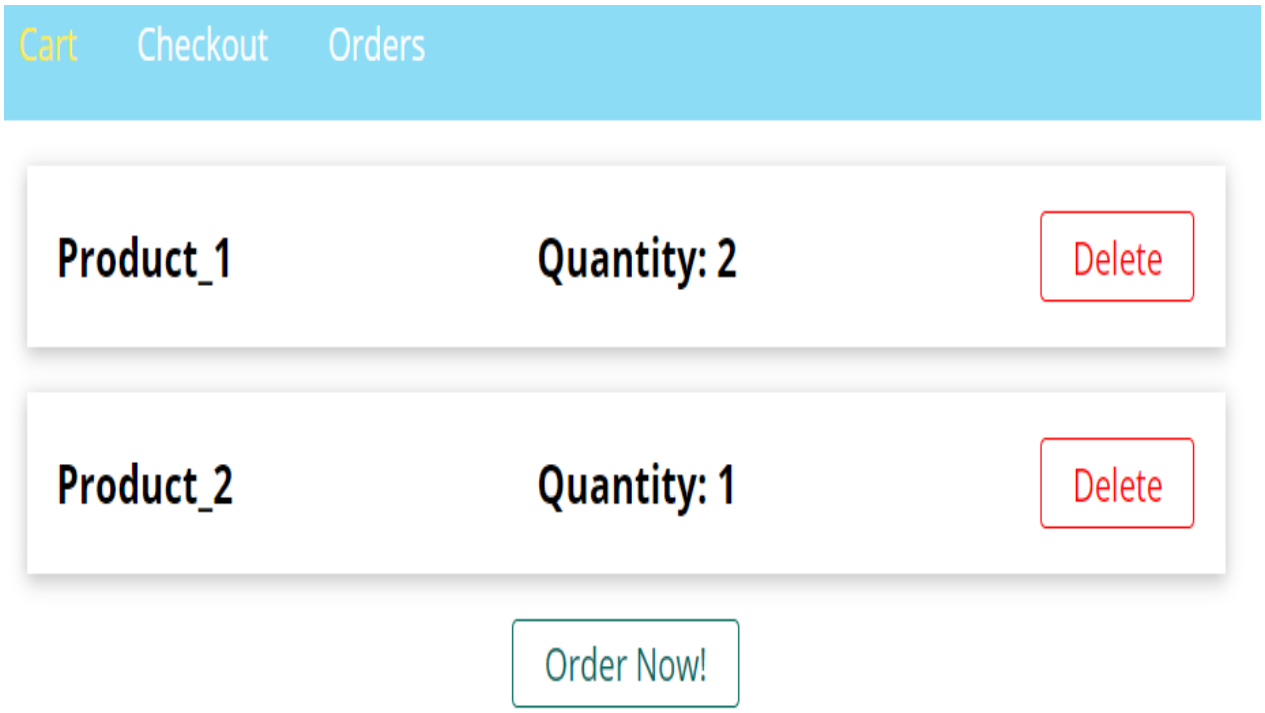

Рисунок 3.15 — Відображення вкладки Cards

Checkout — відображення остаточної версії продуктів що будуть входити до замовлення, на даній сторінці користувач може перевірити коректність даних та впевнено створити замовлення рисунок 3.16.

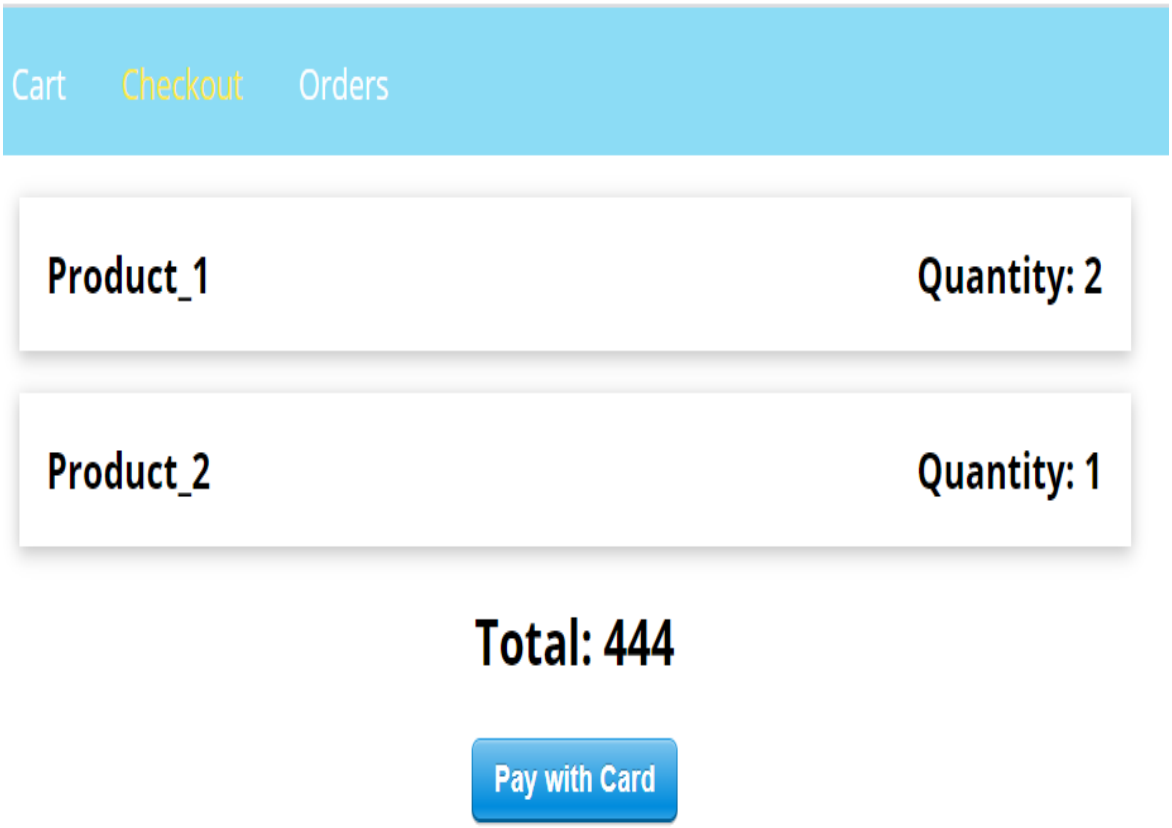

Рисунок 3.16 — Відображення вкладки Checkout

Після успішної оплати (рисунок 3.17), адміністратор може переглянути всі створенні оплати користувачів та детальну інформацію про кожен з них.

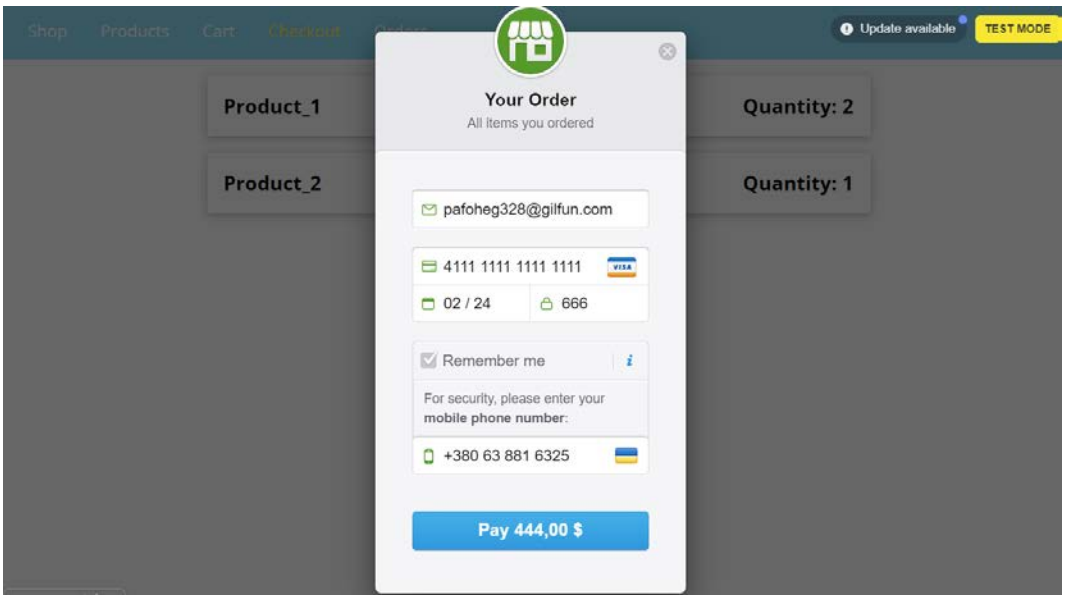

Рисунок 3.17 — Форма заповнення даних для проведення оплати

В адмін панелі допоміжного сервісу stripe рисунок 3.18, що ми використовуємо для проведення оплат товарів.

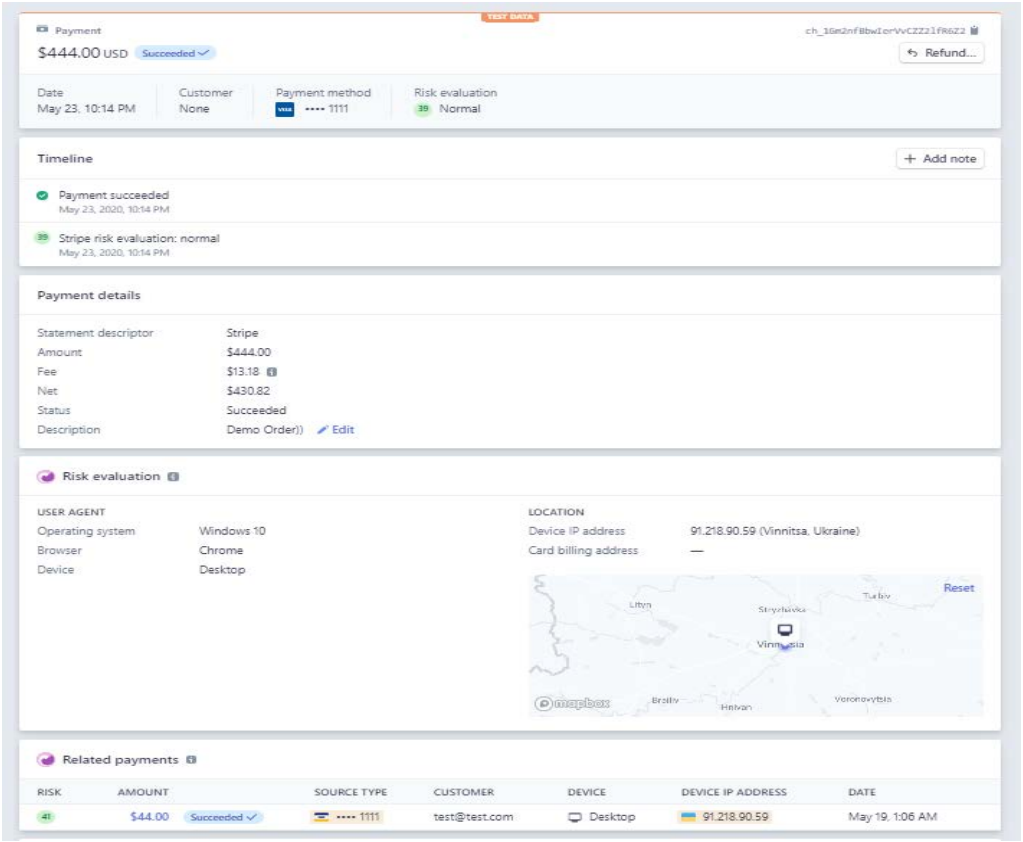

Рисунок 3.18 — Відображення даних щодо створеного платежу

Orders — відображення всіх замовлень, які було створено користувачем (рисунок 3.19).

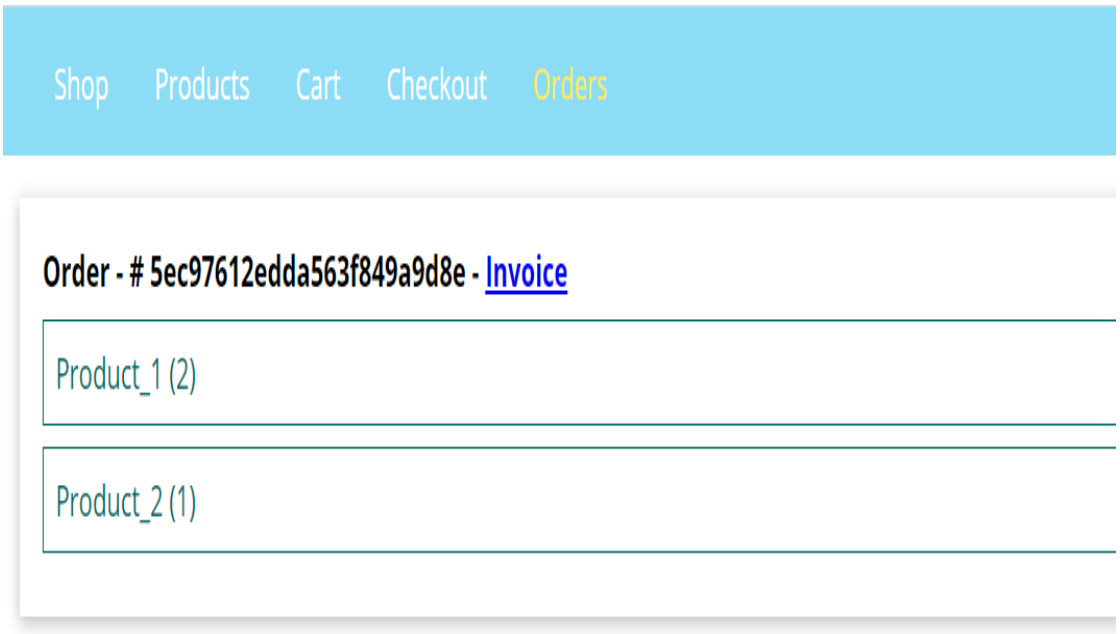

Рисунок 3.19 — Відображення вкладки Orders

Кожен користувач має можливість завантажити чек в PDF форматі, щодо створеного замовлення який буде підтверджувати дійсність замовлення (рисунок 3.20).

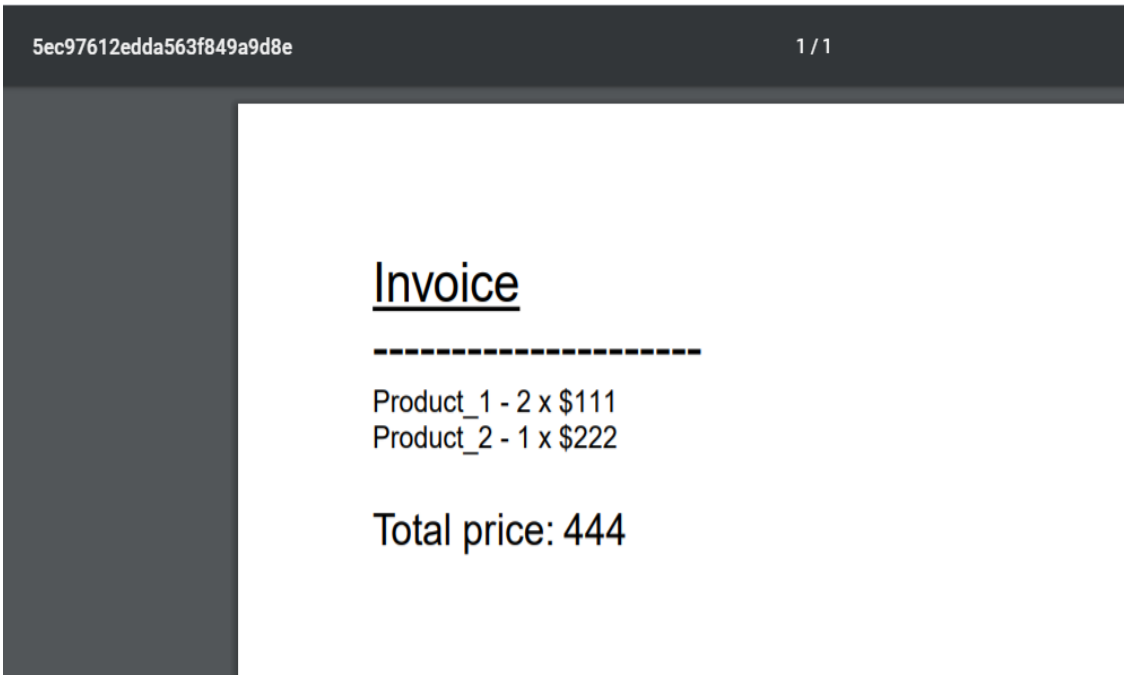

Рисунок 3.20 — Кінцевий чек щодо створеного продукту

Головна сторінка користувача з правами доступу адмін, включає в собі всі вкладки що має звичайний користувач та декілька інших що не включенні до основного функціоналу користувача.

Add products — відображає форму для додавання нового продукту (рисунок 3.21).

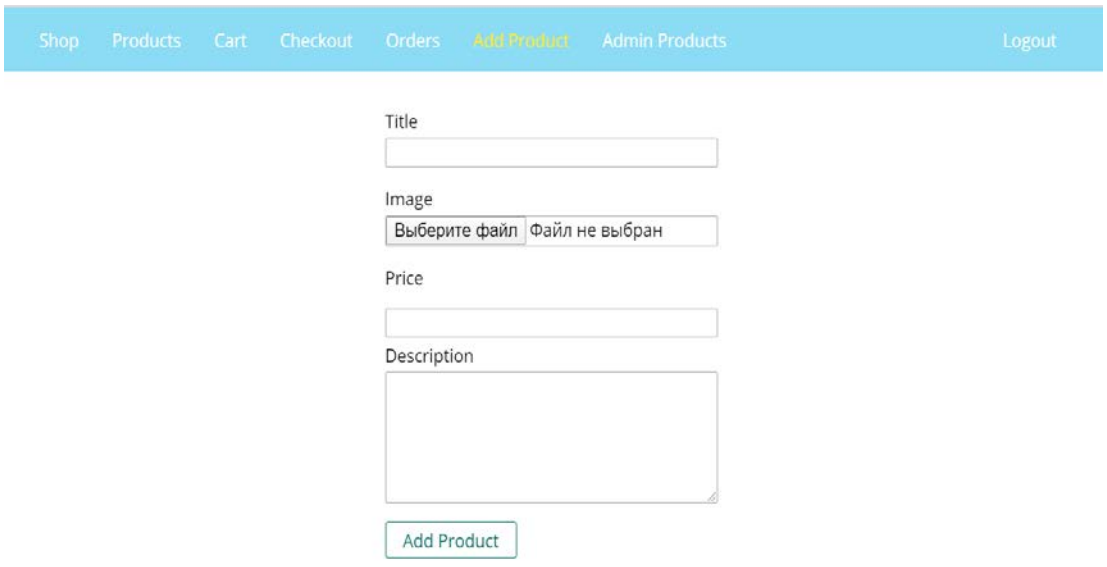

Рисунок 3.21 — Відображення вкладки Add products

Admin products — відображає всі доступні продукти з можливістю змінювати конкретний продукт чи видаляти продукт з бази даних (рисунок 3.22).

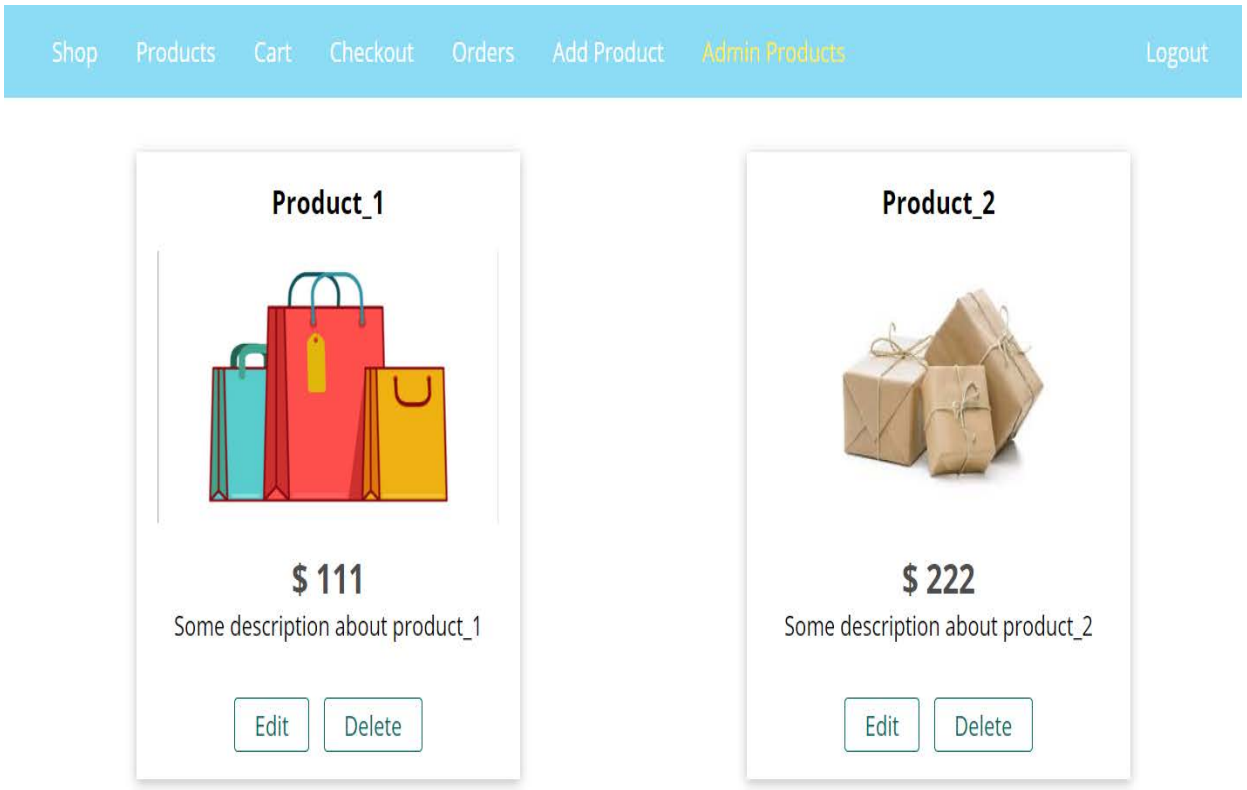

Рисунок 3.22 — Відображення вкладки Admin products

У випадку якщо користувач не пам’ятає власний пароль, потрібно перейти на вкладку для відновлення паролю, до неї можна перейти використавши посилання на сторінці логіна. На вкладці для відновлення паролю (рисунок 3.23), користувач повинен ввести свій електронний адрес на який буде надіслане повідомлення з посиланням для відновлення пароля (рисунок 3.24).

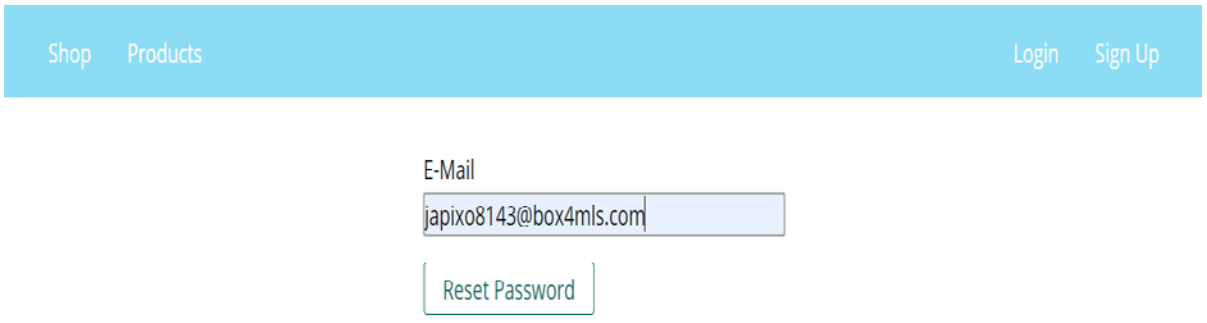

Рисунок 3.23 — Відображення вкладки відновлення паролю

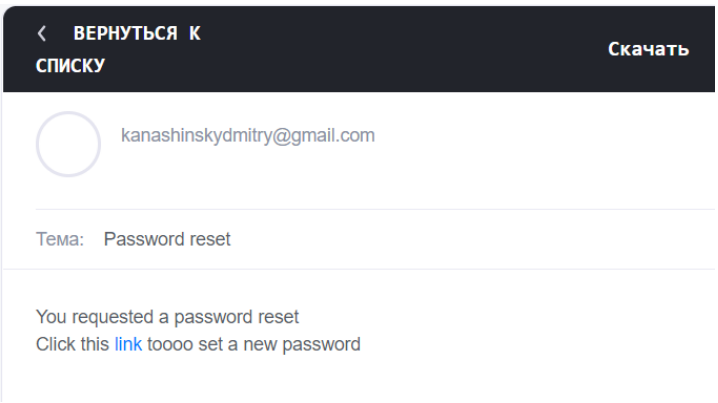

Рисунок 3.24 — повідомлення з посиланням для відновлення пароля

Після натиснення на посилання з повідомлення користувач буде автоматично переправлений на вкладку з формою створення нового паролю для аккаунту (рисунок 3.25).

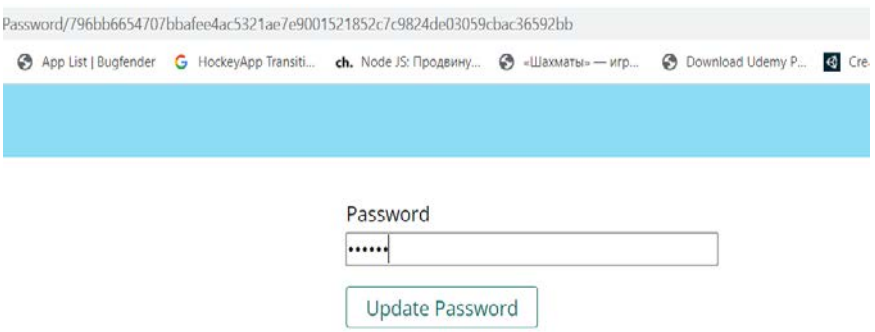

Рисунок 3.25 — вкладка для відновлення паролю користувача

В нашому додатку було розроблено дизайн для користувачів комп’ютерів та користувачів мобільних девайсів. Вигляд дизайну для мобільного пристрою зображено на рисунку 3.26.

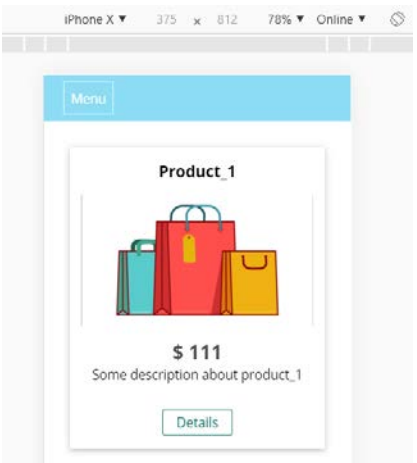

Рисунок 3.26 — Вигляд дизайну на прикладі смартфона IPhone X

Для мобільних девайсів було створено головну панель у вигляді бокової панелі (рисунок 3.27).

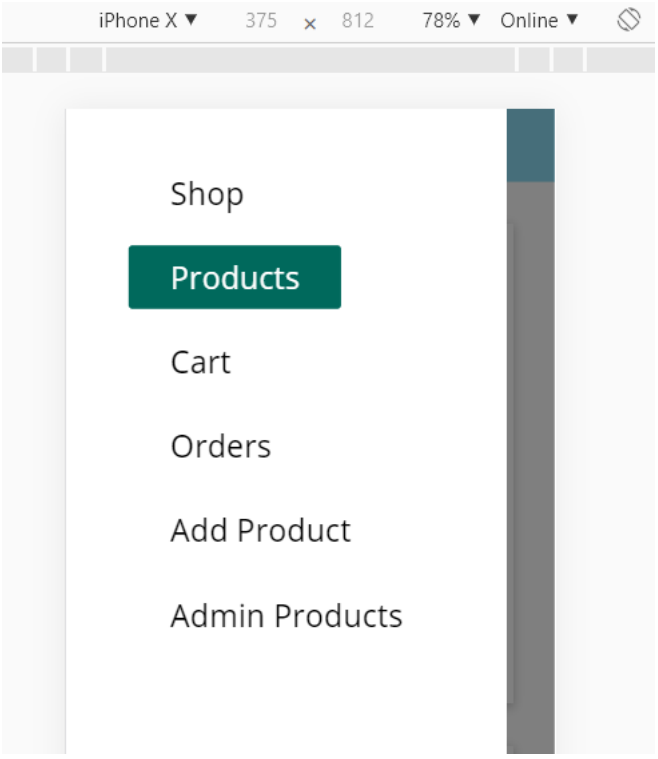

Рисунок 3.27 — Вигляд бокової панелі на прикладі смартфона IPhone X

4 РОЗРАХУНОК ЕКОНОМІЧНОЇ ДОЦІЛЬНОСТІ СТВОРЕННЯ ПРОГРАМНОГО ЗАСОБУ РОЗПІЗНАВАННЯ ОСОБИ ЗА ЗОБРАЖЕННЯМ ОБЛИЧЧЯ

Дослідження завжди дорогі. Ці витрати на виробництво та реалізацію товарів необхідно постійно зменшувати, оскільки це прогрес будь-якого виробництва. На основі економічних розрахунків [28] можна продемонструвати рентабельність та ефективність впровадження результатів досліджень у виробництво, тобто комерціалізація наукових досліджень. Дана магістерська кваліфікаційна робота відноситься до розряду прикладної науково-технічної роботи. Прогнозується виведення науково-технічної розробки на ринок із залученням потенційним інвестора. Дану послідовність приведено на рисунку 4.1.

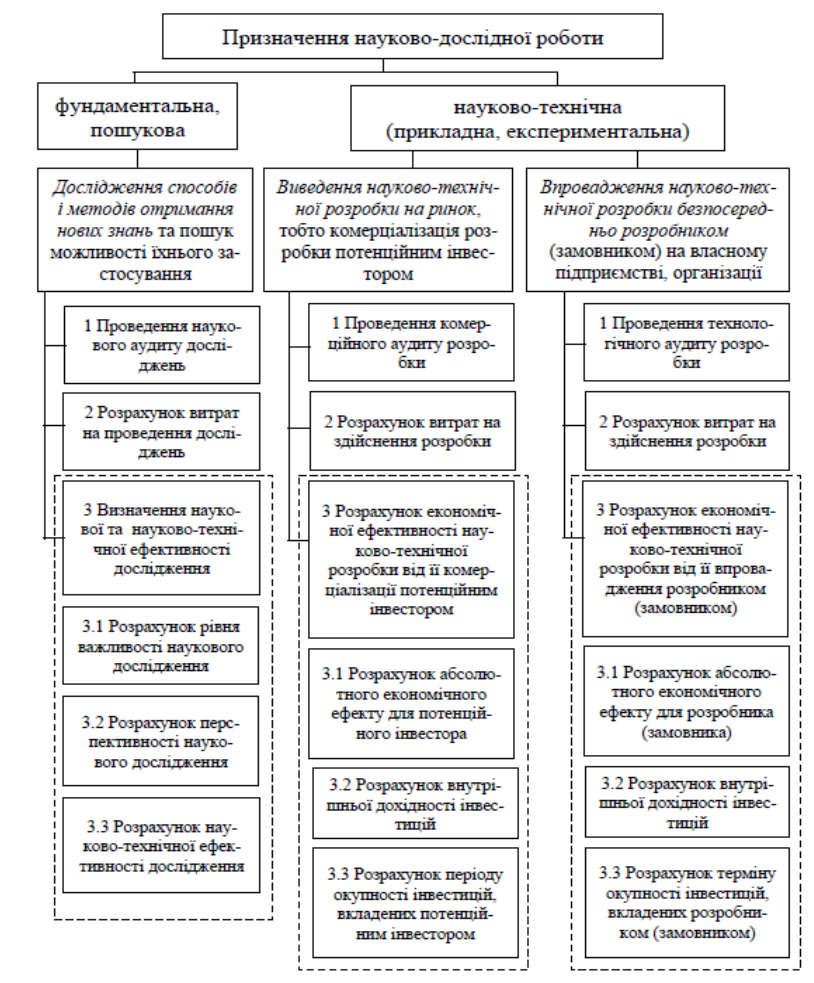

Рисунок 4.1 — Складові економічної частини магістерської кваліфікаційної роботи

Економічна частина цієї магістерської роботи буде поділена на такі елементи. Усі подальші економічні розрахунки будуть розглянуті у згаданих розділах економічної частини.

4.1 Проведення комерційного та технологічного аудиту науково-технічної розробки

Метою оцінки потенціалу комерційного розвитку є оцінка потенціалу комерційного розвитку, що випливає з науково-технічних досліджень. За результатами оцінки робляться висновки про напрямки (особливості) організації в майбутньому її впровадження з урахуванням встановленої оцінки. Комерційний потенціал інвестицій буде оцінюватись відповідно до дванадцяти критеріїв, наведених у таблиці 4.1.

Таблиця 4.1 — Оцінювання комерційного потенціалу розробки

Критерії оцінювання та бали (за 5-бальною шкалою)					
Критерій	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Багато аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів

Продовження таблиці 4.1

5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
Практична здійсненність					
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні.	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси.Джерела фінансування є	Потрібні незначніфінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років.Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років.Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років.Термін окупності інвестицій менше 3-х років

Закінченняя таблиці 4.1

12	Необхідно регламентні документи та велика кількість дозвільних документів на виробництво продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

На основі таблиці різні експерти, у нашому випадку викладачі кафедри ОТ визначають різні результати. Результати цієї оцінки комерційного потенціалу узагальнено у таблиці 4.2.

Таблиця 4.2 — Результати оцінювання комерційного потенціалу розробки

Критерії	Експерт (ПІБ, посада)		
	1 Ткаченко О. М., к.т.н., доц. кафедри ОТ	2 Крупельницький Л.В., к.т.н., доц. кафедри ОТ	3 Черняк О.І., к.т.н., доц. кафедри ОТ
	Бали:		
1.Технічна здійсненність концепції	3	2	3
2.Ринкові переваги (наявність аналогів)	3	2	2
3. Ринкові переваги (ціна продукту)	3	3	3
4.Ринкові переваги (технічні властивості)	3	3	3
5. Ринкові переваги (експлуатаційні витрати)	2	3	3
6. Ринкові перспективи (розмір ринку)	3	2	3
7. Ринкові перспективи (конкуренція)	3	3	3
8. Практична здійсненність (наявність фахівців)	3	3	2
9. Практична здійсненність (наявність фінансів)	2	3	3

Закінчення таблиці 4.2

10.Практична здійсненність (необхідність нових матеріалів)	1	3	2
11. Практична здійсненність (термін реалізації)	3	4	3
12. Практична здійсненність (розробка документів)	3	3	2
Сума балів	$СБ_1 = 34$	$СБ_1 = 34$	$СБ_1 = 32$
Середньо-арифметична сума балів $СБ_С$	$СБ_С = \frac{\sum_1^3 СБ_i}{3} = \frac{34 + 34 + 32}{3} = 33{,}3$		

Відповідно до таблиці 4.2, а також відповідно до рекомендацій, наведених у таблиці 4.3, можна зробити висновок про рівень потенціалу комерційного розвитку.

Таблиця 4.3 — Рівні комерційного потенціалу розробки

Середньоарифметична сума балів $\overline{СБ}$, розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0 — 10	Низький
11 — 20	Нижче середнього
21 — 30	Середній
31 — 40	Вище середнього
41 — 48	Високий

З урахуванням середніх арифметичних балів $СБ_С$ = 33,3, які були визначені експертами, можна зробити висновок, що рівень комерційного потенціалу цієї розробки буде вище середнього. Програмний ресурс COMFY.UA було використано для порівняння властивостей. Це програмне забезпечення має більш широкий спектр застосування в Інтернеті для підтримки інтернет.

Нова розробка, є широкоспрямованою системою що можна використовувати для підтримки інтернет торгівлі різних товарів та послух. Розповсюдження аналога відбувається за допомогою хостингу на серверах Heroku, після вигрузки додатку будь який користувач може використовувати

його для створення покупок. Одним з недоліків програмного ресурсу є те що користувач не може відразу оплатити товар що хоче замовити, в нашому випадку для оплати товарів використовується платіжний сервіс Stripe.

Також, ще один аналог розробленого програмного забезпечення — Watson.Ua, який призначений для продажу промислових товарів. Недоліком даного програмного забезпечення є те що для покупки товару користовачу потрібно зареєструватись на сайті та сайт не підтримує автризації за допомогою сервісів Google. Якщо ж здійснити порівняння із створеним програмним продуктом, то Watson.Ua має певні переваги — більш розширений функціонал, та приємний дизайн якийсь що подобається користувачам, наше програмне забезпечення має переваги в тому що в нас є модуль прогнозування попиту для ефективного продажу товарів та платіжна система для швидкої оплати. Порівння розробки з її аналогами приведено в таблиці 4.4.

Таблиця 4.4 — Порівняння характеристик розробки із аналогом

Показники	Розробка	Аналог1	Аналог2
Функціонал	8	9	9
Швидкодія	9	8	8
Надійність	8	8	8
Метод розповсюдження	7	8	8
Інтерфейс, простота використання	6	7	8

Продукт буде просуватися за допомогою реклами в соціальних мережах, пошукових системах та меседжерах. Використовуючи аналітику цих сервісів, можна буде націлити рекламу на цільову групу захисників інформації.

Продукт також може бути використаний в інформаційній безпеці для розмежування доступу, банківської справи, соціальних мереж, криміналістики, комп’ютерних ігор та інших сфер.

Новизна дослідження полягає в тому, що ми використовуємо Stacking для поєднання декількох алгоритмів прогнозування попиту в один з метою отримання кращих результатів.

Виходячи з результатів цього порівняння, можна з упевненістю сказати, що новий продукт є конкурентоспроможним, оскільки в деяких аспектах в ньому є нові можливості яких не мають аналоги для ефективної торгівлі в інтернеті.

4.3 Розрахунок витрат на здійснення науково-дослідної роботи

У магістерській роботі розглядається інформаційна технологія для підтримки інтернет торгівлі, тому значну частину витрат складають витрати на розробку, а не на виробництво та відтворення. Відповідно, є певна специфіка розрахунків.

4.3.1 Витрати на оплату праці

Основна заробітна плата розробників, що працюють над проектом, визначена у формулі 4.1:

$$З_о = \sum_{i=1}^{k} \frac{M_{ni} \cdot t_i}{T_p}, \qquad (4.1)$$

де k — кількість посад дослідників, залучених до процесу досліджень;

M_{ni} — місячний посадовий оклад конкретного дослідника, грн;

T_p — середня кількість робочих днів в місяці, T_p = 21…23 дні; обрано 22 дні;

t_i— кількість днів роботи конкретного дослідника, дн.

Над створенням розробки працював менеджер проекту та інженер програмного забезпечення, тому ми виконаємо для данх працівників усі необхідні розрахунки, та після чого вносимо їх до таблиці 4.5.

$$Зо.к. = \frac{24200 \cdot 28}{22} = 30800(грн),$$

$$З_{о.в.} = \frac{17600 \cdot 64}{22} = 51200 \text{ (грн)}.$$

Таблиця 4.5 — Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату, грн.
Керівник проекту	24200	1100	28	30800
Старший інженер-програміст	17600	800	64	51200
Всього				82000

Витрати на основну заробітну плату робітників за відповідними найменуваннями робіт відсутні, тобто $З_р = 0$. Додаткова винагорода ($З_{дод.}$) усіх розробників та працівників, які брали участь у цьому етапі роботи, обчислюється як 10 ... 12% від суми основної заробітної плати дослідників та робітників за формулою 4.2:

$$З_{дод.} = (З_о + З_р) \cdot \frac{Н_{дод.}}{100\%}, \quad (4.2)$$

де $Н_{дод.}$ — норма нарахування додаткової заробітної плати.

$$З_{дод.к.} = \frac{10 \cdot 30800}{100} = 3080 \text{ (грн)},$$

$$З_{дод.в.} = \frac{10 \cdot 51200}{100} = 5120 \text{ (грн)},$$

$$З_{дод} = З_{дод.к.} + З_{дод.в.} = 8200 \text{ (грн)}.$$

4.3.2 Відрахування на соціальні заходи

Заробітна плата робітників відсутня, тому $З_р = 0$. Нарахування на заробітну плату дослідників та нарахування на заробітну плату працівників, які брали участь у цьому етапі роботи, розраховується як 22% від суми

основної та додаткової заробітної плати дослідників і робітників за наступною формулою 4.3:

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%}, \quad (4.3)$$

де $Н_{зп}$ — норма нарахування на заробітну плату.

$$З_н = (82000 + 0 + 8200) \cdot \frac{22\%}{100\%} = 19844 \text{ (грн.)}$$

4.3.3 Сировина та матеріали

Витрати на матеріали (М), у вартісному вираженні розраховуються окремо по кожному виду матеріалів за наступною формулою 4.4:

$$М = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot К_j - \sum_{j=1}^{n} В_j \cdot Ц_{вj}, \quad (4.4)$$

де $Н_j$ — кількість матеріалу j-го виду, шт.;

n — кількість видів матеріалу.

$Ц_j$ — ціна матеріалу j-го виду, грн;

$К_j$ — коефіцієнт транспортних витрат, обираємо $К_j = 1{,}15$;

$В_j$ — маса відходів j-го найменування, кг;

$Ц_{вj}$ — вартість відходів j-го найменування, грн/кг.;

Результати розрахунків занесено до таблиці 4.6.

Таблиця 4.6 — Витрати на матеріали

Найменування комплектуючих	Ціна за 1 штуку, грн	Кількість матеріалу, штук	Величина відходів, кг	Ціна відходів, грн/кг	Вартість витраченого матеріалу, грн
Ручка	30,00	1	0,06	1,80	29,89
Карта пам’яті	650,00	1	0	0,00	650,00
Пачка офісного папіру	180,00	1	0,5	2,50	90
Всього (з урахуванням транспортних витрат)					769,89

4.3.4 Розрахунок витрат на комплектуючі

Оскільки кінцевий продукт, який ми створюємо — це програмний інструмент, це не спричиняє жодних витрат на компоненти та $К_в = 0$.

4.3.5 Спецустаткування для наукових (експериментальних) робіт

Спецустаткування для проведення експериментальних робіт для створення інформаційної технології для підтримки інтернет торгівлі.

4.3.6 Програмне забезпечення для наукових (експериментальних) робіт

Програмне забезпечення для створення програмного продукту по розпізнаванню особи за зображенням її обличчя використовується таке, що є у вільному розповсюдженні, тому витрати на придбання такого забезпечення відсутні.

4.3.7 Амортизація обладнання, програмних засобів та приміщень

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо можуть бути розраховані з використанням прямолінійного методу амортизації за формулою 4.5:

$$А_{обл} = \frac{Ц_б}{Т_в} \cdot \frac{t_{вик}}{12,} \quad (4.5)$$

Таблиця 4.7 — Амортизаційні відрахування по кожному виду

Найменування обладнання	Балансова вартість, грн.	Строк корисного використання, років	Термін використання, місяців.	Амортизаційні відрахування, грн
ЕОМ	21000	3	2	1316,67
Приміщення	70000	15	2	633,33
Всього				1950

4.3.8 Паливо та енергія для науково-виробничих цілей

Витрати на силову електроенергію ($В_е$) розраховують за формулою 4.6:

$$В_е = \sum_{i=1}^{n} \frac{W_{yi} \cdot t_i \cdot Ц_е \cdot К_{впі}}{\eta_i}, \quad (4.6)$$

де W_{yi} — встановлена потужність обладнання на певному етапі розробки, кВт;

t_i — тривалість роботи обладнання на етапі дослідження, год;

$Ц_е$ — вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), $Ц_е = 4{,}62$ [29];

$К_{впі}$ — коефіцієнт, що враховує використання потужності, $К_{впі} < 1$; обираємо $К_{впі} = 0{,}7$;

η_i — коефіцієнт корисної дії обладнання, $\eta_i < 1$; обираємо $\eta_i = 0{,}8$.

$$В_е = \sum_{i=1}^{1} \frac{0{,}07 \cdot 736 \cdot 4{,}62 \cdot 0{,}7}{0{,}8} = 208{,}26 \text{ (грн.)}$$

Проведені розрахунки необхідно звести до таблиці 4.8.

Таблиця 4.8 — Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
ЕОМ	0,07	736	208,26
Всього			208,26

4.3.9 Службові відрядження

Під час розробки програмного забезпечення відрядження штатних працівників, працівників організацій, які працюють за договорами цивільно-правового характеру, магістрів, зайнятих розробленням досліджень, відрядження, пов’язані з проведенням випробувань машин та приладів, а також витрати на відрядження на наукові з’їзди, конференції, наради, пов’язані з виконанням конкретних досліджень, не плануються.

4.3.10 Витрати на роботи, які виконують сторонні підприємства, установи і організації

Витрати за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації» не плануються, так як у цьому не має потреби.

4.3.11 Інші витрати

Витрати за статтею «Інші витрати» розраховуються як 50…100% від суми основної заробітної плати дослідників та робітників за формулою 4.7:

$$I_в = \left(З_о + З_р\right) \cdot \frac{Н_{ів}}{100\%}, \quad (4.7)$$

де $Н_{ів}$ — норма нарахування за статтею «Інші витрати».

$$I_{в.к.} = 30800 \cdot \frac{50\%}{100\%} = 15400 \text{ (грн.)},$$

$$I_{в.в.} = 51200 \cdot \frac{50\%}{100\%} = 25600 \text{ (грн.)},$$

$$I_в = I_{в.к.} + I_{в.в.} = 41000 \text{ (грн.)}.$$

4.3.12 Накладні (загальновиробничі) витрати

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуються як 100…150% від суми основної заробітної плати дослідників та робітників за формулою 4.8:

$$В_{нзв} = \left(З_о + З_р\right) \cdot \frac{Н_{нзв}}{100\%}, \quad (4.8)$$

де $Н_{нзв}$ — норма нарахування за статтею «Накладні (загальновиробничі) витрати».

Беремо норму нарахування 100%.

$$В_{нзв.в.} = 30800 \cdot \frac{100\%}{100\%} = 30800 \text{ (грн.)},$$

$$В_{нзв.в.} = 51200 \cdot \frac{100\%}{100\%} = 51200\ (грн.),$$

$$В_{нзв} = В_{нзв.к.} + В_{нзв.в.} = 82000\ (грн.).$$

Витрати на проведення науково-дослідної роботи розраховуються як сума всіх попередніх статей витрат за формулою 4.9:

$$В_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + В_{спец} + В_{прг} + А_{обл} + В_е + В_{св} + В_{сп} + І_в + В_{нзв}. \quad (4.9)$$

У нашому випадку:

$З_р = 0, К_в = 0, В_{спец} = 0, В_{прг} = 0, В_{св} = 0, В_{сп} = 0$, тому отримаємо:

$$В_{заг} = 82000 + 8200 + 19844 + 769{,}89 + 1950 + 208{,}26 + 41000 + 82000 = 235972{,}15\ (грн).$$

Загальні витрати ЗВ на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховуються за формулою 4.10:

$$ЗВ = \frac{В_{заг}}{\eta}, \quad (4.10)$$

де η — коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи; обираємо η = 0,5.

$$ЗВ = \frac{235972{,}15}{0{,}5} = 471944{,}3\ (грн).$$

4.4 Розрахунок економічної ефективності науково-технічної розробки за її можливої комерціалізації потенційним інвестором

Розробка чи суттєве вдосконалення програмного засобу (програмного забезпечення, програмного продукту) для використання масовим споживачем.

Для всіх наведених випадків можливе збільшення чистого прибутку у потенційного інвестора $\Delta П_і$ для кожного із років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховується за формулою 4.11:

$$\Delta П_i = (\pm \Delta Ц_o \cdot N \cdot Ц_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot \left(1 - \frac{\vartheta}{100}\right) \qquad (4.11)$$

де $\pm\Delta Ц_o$— зміна основного якісного показника від впровадження результа-тів науково-технічної розробки в аналізованому році;

N — основний кількісний показник, який визначає величину попиту на аналогічні чи подібні розробки у році до впровадження результатів нової науково-технічної розробки;

$Ц_o$ — основний якісний показник, який визначає ціну реалізації нової науково-технічної розробки в аналізованому році, $Ц_o = Ц_б \pm \Delta Ц_o$;

$Ц_б$ — основний якісний показник, який визначає ціну реалізації існуючої (базової) науково-технічної розробки у році до впровадження результатів;

ΔN — зміна основного кількісного показника від впровадження результа-тів науково-технічної розробки в аналізованому році;

λ — коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2021 році ставка податку на додану вартість становить 20%, а коефіцієнт λ=0,8333;

ρ — коефіцієнт, який враховує рентабельність інноваційного продукту (послуги). Рекомендується брати ρ=0,2...0,5. Обраємо $\rho = 0{,}26$;

ϑ — ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2021 році ϑ=18%.

В результаті впровадження результатів наукових розробок поліпшується якість програмного забезпечення, що дозволяє подорожчати за його впровадження, а кількість потенційних користувачів ресурсу збільшиться — у перший рік — на 1310 одиниць, на другий рік — ще на 850 одиниць, на третій рік — ще 1550 штук.

Ми прогнозуємо щорічний приріст чистого прибутку компанії від впровадження результатів наукових розробок щодо вихідного стану. Збільшення чистого прибутку підприємства $\Delta П_i$за перший рік складе:

$$\Delta П_1 = [1100 \cdot 0 + (3500 + 1100) \cdot 1310] \cdot 0,8333 \cdot 0{,}26 \cdot \left(1 - \frac{18\%}{100\%}\right)$$
$$= 1070771,93 \text{ (грн)}.$$

Збільшення чистого прибутку компанії $\Delta П_i$ на другий рік (порівняно з базовим, тобто роком, що передує впровадженню результатів наукових досліджень) складе:

$$\Delta П_2 = [1100 \cdot 0 + (3500 + 1100) \cdot (1310 + 850)] \cdot 0,8333 \cdot 0,26 \cdot \left(1 - \frac{18\%}{100\%}\right) = 1765420,81 \text{ (грн)}.$$

Збільшення чистого прибутку підприємства $\Delta П_i$ на третій рік складе:

$$\Delta П_3 = [1100 \cdot 0 + (3500 + 1100) \cdot (1310 + 850 + 1550)] \cdot 0,8333 \cdot 0,26 \cdot \left(1 - \frac{18\%}{100\%}\right) = 3032133,47 \text{ (грн)}.$$

Далі розраховують приведену вартість збільшення всіх чистих прибутків ПП, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки (формула 4.12):

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \quad (4.12)$$

де $\Delta П_i$ — збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

Т — період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки;

τ — ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ = 0,05…0,15, обираємо $\tau = 0,1$;

t— період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

$$ПП = \frac{1070771,93}{(1+0,1)^1} + \frac{1765420,81}{(1+0,1)^2} + \frac{3032133,47}{(1+0,1)^3} = 973429,02 + 1459025,46 + 2278086,75 = 4710541,23 \text{ (грн.)}$$

Далі розраховують величину початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки. Для цього можна використати формулу 4.13:

$$PV = k_{інв} \cdot ЗВ, \quad (4.13)$$

де $k_{інв}$ — коефіцієнт, що враховує витрати інвестора на впровадження науко-во-технічної розробки та її комерціалізацію

ЗВ — загальні витрати на проведення науково-технічної розробки та оформлення її результатів, грн.

$$PV = 2 \cdot 471944{,}3 = 943888{,}6 \text{ (грн.)}$$

Тоді абсолютний економічний ефект $E_{абс}$ або чистий приведений дохіддля потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме (формула 4.14):

$$E_{абс} = ПП - PV \quad (4.14)$$

де ПП — приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, грн;

PV — теперішня вартість початкових інвестицій, грн.

$$E_{абс} = 4710541{,}23 - 943888{,}6 = 3766652{,}63 \text{ (грн.)}$$

Внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки, розраховується за формулою 4.15:

$$E_в = \sqrt[Т_ж]{1 + \frac{E_{абс}}{P}} - 1, \quad (4.15)$$

де $E_{абс}$ — абсолютний економічний ефект вкладених інвестицій, грн;

PV — теперішня вартість початкових інвестицій, грн;

$Т_ж$ — життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, роки.

$$E_в = \sqrt[3]{1 + \frac{3766652,63}{943888,6}} - 1 = 0,708$$

Далі визначають бар'єрну ставку дисконтування τ_{min}, тобто мінімальну внутрішню економічну дохідність інвестицій, нижче якої кошти у впровадження науково-технічної розробки та її комерціалізацію вкладатися не будуть.

Мінімальна внутрішня економічна дохідність вкладених інвестицій $\tau_{мін}$визначається за формулою 4.16:

$$\tau_{мін} = d + f, \qquad (4.16)$$

де d— середньозважена ставка за депозитними операціями в комерційних банках; в 2021 році в Україні d = 0,9...0,12, обираємо d = 0,11;

f — показник, що характеризує ризикованість вкладення інвестицій; зазвичай величина f=0,05...0,5, але може бути і значно вищою. Обираємо f = 0,2;

$$\tau_{мін} = d + f = 0,11 + 0,2 = 0,31\%$$

Величина $E_в > \tau_{мін}$, отже інвестор може бути зацікавлений у фінансуванні цього дослідження.

Далі розраховуємо період окупності інвестицій $T_{ок}$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки (формула 4.17):

$$T_{ок} = \frac{1}{E_в}, \qquad (4.17)$$

де $E_в$ — внутрішня економічна дохідність вкладених інвестицій.

$$T_{ок} = \frac{1}{0,708} = 1,4 \text{ року}$$

Оскільки $T_{ок}$= 1,4 року тоді розвиток доречний

ВИСНОВКИ

Під час виконання завдання дипломної роботи було створено програмне забезпечення для забезпечення інтернет торгівлі.

У першому розділі було розглянуто основну інформацію про мову програмування Node.js, що застосовується для розробки серверної частини веб додатків, також було описано як застосовувати C++ модулі для серверної розробки на Node.js. Також в даному розділі було обрано базу даних що найкраще підходить для розробки веб додатку, та застосування шаблону MVC для швидкої розробки програмного забезпечення.

У другому розділі було розглянуто та проаналізованно методи прогнозування даних за допомогою різних Phyton бібліотек та було створенне їх порівння для вибору тієї що найбільше підходе для наших потреб. Було обрано використовувати Stacking для поєднання декількох моделей щоб отримати найбільш точні дані прогнозування.

У третьому розділі було розглянуто структуру проекту, які допоміжні залежності використовуються в нашому проекті. Також було детальніше розглянуто застосування шаблону MVC використовуючи Node.js та Express. Також в розділі було детально розглянуто всі три основні частини шаблона MVC з невеликими основними лістингами коду для кращого розуміння як використовувати шаблон MVC разом з фреймворком Express та основні елементи що потрібно встановити на сервер в якому буде запущений над додаток. Основними елементами мають бути Node.js та PostgreSQL. Також було розглянуто версії кожного з елементів для успішного старту сервера. Для користувача було створено детальну документацію по продукту, після перегляду якої будь-який користувач зможе використовувати всі можливості продукту як на рівні звичайного користувача, так і користувача з правами доступу адмін.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Березко О.Л. Концепція створення вебсайта Національного університету „Львівська політехніка” / О.Л. Березко, А.М. Пелещишин, П.І. Жежнич [Електронний ресурс]. Режим доступу: http://science.lpnu.ua/sites/default/files/journal-paper/2017/jun/3146/12-berezko-57 — 65.pdf.

2. Берковський В.В. Зіставлення технологій створення сайтів з їх цільовим призначенням / В.В. Берковський, В.О. Радіонов, В.С Бурковський // Наука і техніка Повітряних Сил Збройних Сил України. — 2020. — № 2(19).––122-124 с.

3. Браэм Г. Психология цвета / Г. Браэм. — Москва: АСТ, 2009. — 158 с.

4. Васильев А.Н. JavaScript в примерах и задачах / А.Н. Васильев. — Москва: Издательство „Э”, 2019. — 720 с.

5. Гушев А.А. Классификация web-сайтов образовательных учреждений / А.А. Гушев // Современные научные исследования и инновации. — 2019. — № 11. [Електронний ресурс]. Режим доступу: http://web.snauka.ru/issues/2019/11/74309.

6. Дакетт Д. JavaScripte и jQuery. Интерактивная веб-разработка / Д. Дакетт. — Москва: Издательство „Э”, 2019. — 640 с.

7. Евдокимов Н.С. Создание сайтов / Н.С. Евдокимов. — Санкт-Петербург: Питер, 2017. — 410 с.

8. Компанєєтс М.О. Принципи проектування ефективних вебсайтів / М.О. Компанєєтс // Молодий вчений. — 2019. — № 9. — Част. 2. — С. 106-108. [Електронний ресурс]. Режим доступу: http://molodyvcheny.in.ua/files/journal/ 2019/9/66.pdf.

9. Котеров Д.В. PHP 7 / Д.В. Котеров. — Санкт-Петербург: БХВ-Петербург, 2016. —1088 с.

10. Круг С. Веб-дизайн: книга Стива Круга или не заставляйте меня думать!” / С. Круг. — 2-е изд. — Санкт-Петербург: Символ-Плюс, 20018. — 224 с. 99

11. Лещев А.В. Создание интерактивного web-сайта : учебный курс / А.В. Лещев. — Санкт-Петербург: Питер, 2003. — 544 с.

12. Мак-Дональд М. HTML5. Недостающее руководство / М. Мак-Дональд. — Санкт-Петербург: БХВ-Петербург, 2012. — 480 с.

13. Мак-Дональд М. Создание Web-сайта. Недостающее руководство / М. Мак-Дональд. — 3-е изд. — Санкт-Петербург: БХВ-Петербург, 2017. — 624 с.

14. Макфарланд Д. Новая большая книга CSS / Д. Макфарланд. — СанктПетербург: Питер, 2019. — 720 с.

15. Мальцев И.П. Проектирование веб-сайта: пошаговое руководство для тех, кто понял, что сайт ему необходим, но пока не знает, что с этим делать / И.П. Мальцев. — ЛитРес: Самиздат, 2019. – 118 с. [Електронний ресурс]. Режим доступу: https://prowebber.ru/text-lessons/20608-kniga-proektirovanie-veb-saytaimalcev.html.

16. Мельник О. Выбираем профессию frontend- и backend-разработчика: принципы и отличия / О. Мельник [Електронний ресурс]. – Режим доступу: https://skillbox.ru/media/code/frontend_i_backend_razrabotka/.

17. Мейер Э. CSS: полный справочник / Э. Мейер, Э. Уэйл. – СанктПетербург: ООО „Диалектика”, 2019. – 1088 с.

18. Милин В. Обзор редакторов кода / В. Милин [Електронний ресурс]. Режим доступу: https://htmlacademy.ru/blog/useful/programming/editors-for-the-coders.

19. Никсон Р. Создаем динамические веб-сайты с помощью PHP, MySQL, JavaScript, CSS и HTML5 / Р. Никсон. — Санкт-Петербург: Питер, 2016. — 768 с.

20. Пасічник Н.Р. Формалізм в постановці задачі створення якісного сайту / Н.Р. Пасічник // Наукові праці Донецького національного технічного університету. Інформатика, кібернетика та обчислювальна техніка. —

Донецьк. — 2011. — Вип. 14 (188). — С. 325-329. [Електронний ресурс]. — Режим доступу: http://nbuv.gov.ua/UJRN/Npdntu_inf_2011_14_49.

21. Ривкінд Й.Я. Інформатика: підручник для 11 кл. загальноосвіт. навч. закл. : академ. рівень, профільн. рівень / Й.Я. Ривкінд, Т.І. Лисенко, Л.А. Чернікова, В.В. Шакотько / за заг. ред. М.З. Згуровського. – Київ: Генеза, 2011. — 304 с.

22. Робин Н. Эффективный Web-сайт : учебн. пос. / Н. Робин, К. Греди. — Москва: Триумф, 2004. — 560 с.

23. Самойлова Т.И. Основные виды сайтов / Т.И. Самойлова [Електронний ресурс]. Режим доступу: https://tutor-web.susu.ru/2017/07/12/osnovnyie-vidyi-saytov/

24. Самойлова Т.И. Основные типы сайтов / Т.И. Самойлова [Електронний ресурс]. Режим доступу: https://tutor-web.susu.ru/2019/07/12/osnovnyie-tipyi-saytov/.

25. Томал Р. Основы Web-Дизайна. Руководство / Р. Томал [Електронний ресурс]. Режим доступу: http://padabum.com/d.php?id=191574.

26. Фримен Э. Изучаем программирование на JavaScript / Э. Фримен, Э. Робсон. — Санкт-Петербург: Питер, 2015. — 640 с.

27. Шевченко Д.А. Эффективность веб-сайтов высших учебных заведений. Методика оценки конкурентоспособности сайта в Интернет / Д.А. Шевченко, Ю.В. Локтюшина. — Москва: МИПК, 2019. — 141 с..

28. Способ идентификации личности человека с помощью двух и более разнесенных видеокамер (патент РФ №2370817 от 06.10.2014, МКП H03J 09/54).

29. OpenCV — библиотека компьютерного зрения с открытым исходным кодом [Електронний ресурс]. Режим доступу: http://software.intel.com/en-us/articles/.

30. Поняття алгоритму [Електронний ресурс]. Режим доступу: https://sites.google.com/site/vchimoinformatikurazom/zavdanna/ponatta-algoritmu.

31. Распознавание ключевых точек лица на изображении человека [Електронний ресурс]. Режим доступу: https://moluch.ru/archive/264/61268.

32. Модель бустинга биоинспирированных алгоритмов для решения задач классификации и кластеризации [Електронний ресурс]. Режим доступу: https://cyberleninka.ru/article/n/model-bustinga-bioinspirirovannyh-algoritmov-dlya-resheniya-zadach-klassifikatsii-i-klasterizatsii.

33. Лінійна нейронна мережа [Електронний ресурс]. Режим доступу: https://uk.discografie.org/369813-linear-vs-nonlinear-neural-network-PUTODO.

34. Моделі і методи розпізнавання образів [Електронний ресурс]. Режим доступу: http://kiis.knu.ua/temi-modeli-i-metodi-rozpiznavannja-obraziv/.

35. Створення системи розпізнавання облич на основі вейвлет-перетворень [Електронний ресурс]. Режим доступу: http://journals.kntu.net.ua/index.php/pit/article/view/623.

36. Завдання розпізнавання зображень [Електронний ресурс]. Режим доступу: https://uadoc.zavantag.com/text/2103/index-1.html.

37. Можливості Face ID: ідентифікація облич за фото [Електронний ресурс]. Режим доступу: https://evergreens.com.ua/ua/articles/digital-facial-recognition.html.

38. Timothy F Cootes, Gareth J Edwards, and Christopher J Taylor. Active appearance models / Timothy F Cootes, Gareth J Edwards, and Christopher J Taylor // IEEE Transactions on Pattern Analysis and Machine Intelligence, 2001, (6) — pp.681 — 685.

39. Timothy F Cootes, Christopher J Taylor, David H Cooper, and Jim Graham. Active shape models-their training and application / Timothy F Cootes, Christopher J Taylor, David H Cooper, Jim Graham Computer // Vision and Image Understanding, 1995, 61(1) — pp.38 — 59.

40. Gareth J Edwards, Timothy F Cootes, and Christopher J Taylor. Face recognition using active appearance models / Gareth J Edwards, Timothy F Cootes, Christopher J Taylor // In ECCV, 1998 — pp.178 — 189.

41. A. D. Jepson, D. J. Fleet, and T. F. El-Maraghi. Robust online appearance models for visual tracking / A. D. Jepson, D. J. Fleet, T. F. El-Maraghi // IEEE Trans. on Pattern Analysis and Machine Intelligence (TPAMI), 2003, 25(10) — pp.1296 — 1311.

42. I. Matthews and S. Baker. Active appearance models revisited / I. Matthews and S. Baker // Int. Journal of Computer Vision (IJCV), 2004, 60(2) — pp.135 — 164.

43. Kazemi V., J. Sullivan. One millisecond face alignment with an ensemble of regression trees / Kazemi V., J. Sullivan // 2014 IEEE Conference on Computer Vision and Pattern Recognition, 2014 — pp. 1867 — 1874.

44. Shengtao Xiao, Shuicheng Yan, Ashraf A. Kassim. Facial Landmark Detection via Progressive Initialization / Shengtao Xiao, Shuicheng Yan, Ashraf A. Kassim // Vision–ECCV, 2015 — pp. 33 — 40.

45. Christos Sagonas, Georgios Tzimiropoulos, Stefanos Zafeiriou, and Maja Pantic. 300 faces in-the-wild challenge: The first facial landmark localization challenge / Christos Sagonas, Georgios Tzimiropoulos, Stefanos Zafeiriou, Maja Pantic // In 2013 IEEE International Conference on Computer VisionWorkshops, 2013 — pp. 397 — 403.

46. Kostiantyn Khabarlak and Larysa Koriashkina. Fast facial landmark detection and applications: A survey, 2021. [Електронний ресурс]. https: //arxiv.org/ftp/arxiv/papers/2101/2101.10808.pdf.

47. Daniel Merget, Matthias Rock, and Gerhard Rigoll. Robust facial landmark detection via a fully-convolutional localglobal context network / Daniel Merget, Matthias Rock, Gerhard Rigoll // In CVPR, 2018 — pp. 256 — 260.

48. Wayne Wu, Chen Qian, Shuo Yang, Quan Wang, Yici Cai, and Qiang Zhou. Look at boundary: A boundary-aware face alignment algorithm / Wayne Wu, Chen Qian, Shuo Yang, Quan Wang, Yici Cai, Qiang Zhou // In CVPR, 2018 — pp. 340 — 351.

49. Burgos-Artizzu, X. P., P. Perona, P. Dollar. Robust Face Landmark Estimation under Occlusion / Burgos-Artizzu, X. P., P. Perona, P. Dollar // IEEE International Conference on Computer Vision, Sydney, NSW, 2013 — pp. 1513 — 1520.

50. Kumar Ku, Yuci Pai. LUVLi Face Alignment: Estimating Landmarks' Location, Uncertainty, and Visibility Likelihood / Kumar Ku, Yuci Pai. // IEEE/CVF

Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 2020 — pp. 8233 — 8243.

51. Кавецький В. В. Економічне обґрунтування інноваційних рішень: Практикум /В.В.Кавецький, В.О.Козловський, І.В.Причепа. — ВНТУ, 2013. — 110 с.

52. Адлер О.О. Методичні вказівки до підготовки та написання курсової роботи з дисципліни «Економічне обґрунтування інноваційних рішень» / Уклад. О.О.Адлер, І.В.Причепа, Н.М.Тарасюк. — Вінниця: ВНТУ, 2014. — 38 с.

ДОДАТОК А

Міністерство освіти та науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

ЗАТВЕРДЖУЮ
Завідувач кафедри ОТ
_____проф., д.т.н. О. Д. Азаров
«___» _____ 2021 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання магістерської кваліфікаційної роботи
«Інформаційна технологія для підтримки інтернет торгівлі з прогнозуванням попиту на основі аналізу статистики попередніх продаж»
08-23.МКР.024.00.000 ТЗ

Науковий керівник: д.т.н., професор
_____ Ткаченко Т.Б.
Магістрант групи 2КІ-20м
_____ Каташинський Д.О

Вінниця 2021 р.

1 Підстава для виконання магістерської кваліфікаційної роботи (МКР)

1.1 Актуальність даного дослідження визначається необхідністю створення прогнозування попиту на основі попередніх даних. За допомогою даного дослідження ми зможемо ефективніше продавати товари в наслідок чого зможемо отримати більший прибуток.

1.2 Наказ про затвердження теми магістерської кваліфікаційної роботи.

2 Мета і призначення МКР

2.1 Мета магістерської роботи полягає у розробці інформаційної технології з прогнозуванням попиту для підтримки інтернет торгівлі.

2.2 Призначення розробки — виконання магістерської кваліфікаційної роботи.

3 Вихідні дані

Вихідні дані для виконання МКР — ключі для доступу до бази даних, дані доступних товарів, секретні ключі для доступу до Heroku.

4 Вимоги до виконання МКР

МКР повинна задовольняти такі вимоги:

— запропонувати нові підходи для прогнозування попиту;

— розробити серверну та клієнтську частину;

— підключення оплати товарів за допомогою стороннього сервісу;

— результат роботи, а саме готовий додаток для підтримки інтернет торгівлі.

5 Етапи МКР та очікувані результати

Етапи МКР та очікуванні результати наведено в таблиці А.1

6 Матеріали

Матеріали, що подаються до захисту МКР — пояснювальна записка МКР, ілюстративні та графічні матеріали, протокол попереднього захисту МКР на кафедрі, відзив наукового керівника, відзив рецензента, протоколи

складання державних екзаменів, анотації до МКР українською та іноземною мовами, довідка про відповідність оформлення МКР діючим вимогам.

Таблиця А.1 — Етапи МКР та очікуванні результати

№	Назва етапу	Термін виконання		Очікувані результати
		початок	кінець	
1	Аналіз завдання. Вступ	01.09.21	10.09.21	Вступ
2	Аналіз літературних джерел для розпізнавання особи	13.09.21	22.09.21	розділ 1
3	Розробка технічного завдання	23.09.2021	24.09.21	Технічне завдання
4	Розробка структури інформаційної технології	25.09.21	08.10.21	Розділ 2, розробка структури
5	Розробка серверної та клієнтської частини	11.10.21	29.10.21	Розділ 3, розробка програми
6	Практична реалізація, результати.	01.11.21	14.11.21	Розділ 3
7	Розробка економічної частини	15.11.21	30.11.21	Розділ 4
8	Оформлення пояснювальної записки	02.12.21	15.12.21	ПЗ, презентація

7 Порядок контролю виконання та захисту МКР

Виконання етапів розрахункової та графічної документації МКР контролюється науковим керівником згідно зі встановленими термінами. Захист МКР відбувається на засіданні Державної екзаменаційної комісії, затвердженою наказом ректора.

8 Вимоги до оформлення МКР

Вимоги викладені в ДСТУ 3008:2015 та положення ВНТУ про МКР-2021.

Технічне завдання до виконання отримав _____ Каташинський Д. О.

ДОДАТОК Б

Admin Controller

```
exports.getAddProduct = (req, res, next) => {
    res.render('admin/edit-product', {
        pageTitle: 'Add Product',
        path: '/admin/add-product',
        editing: false,
        hasError: false,
        errorMessage: null,
        oldInput: undefined,
        validationsError: []
    });
};

exports.postAddProduct = (req, res, next) => {
    const title = req.body.title;
    const image = req.file;
    const price = req.body.price;
    const description = req.body.description;

    const errors = validationResult(req);
    if (!errors.isEmpty()) {
        return res.status(422).render('admin/edit-product', {
            path: '/admin/add-product',
            pageTitle: 'Add Product',
            errorMessage: errors.array()[0].msg,
            editing: false,
            hasError: true,
            product: {
                title: title,
                price: price,
                description: description
            },
            validationsError: errors.array()
        })
    }

    if(!image){
        return res.status(422).render('admin/edit-product', {
            path: '/admin/add-product',
            pageTitle: 'Add Product',
            errorMessage: "Attached file is not a image",
            editing: false,
            hasError: true,
```

```
                product: {
                    title: title,
                    price: price,
                    description: description
                },
                validationsError: []
            })
        }

        let imageUrl = image.path

        console.log(image)

        let product = new Product({ title, price, imageUrl, description, userId: req.
user._id })

        product.save()
            .then(result => {
                res.redirect('/admin/products');
            })
            .catch(err => {
                const error = new Error(err)
                error.httpStatusCode = 500
                return next(error)
            })
    };

    exports.getEditProduct = (req, res, next) => {
        const editMode = req.query.edit;

        if (!editMode) {
            return res.redirect('/');
        }
        const prodId = req.params.productId;

        Product.findById(prodId)
            .then(product => {
                res.render('admin/edit-product', {
                    pageTitle: 'Edit Product',
                    path: '/admin/edit-product',
                    editing: editMode,
                    hasError: false,
                    product: product,
                    errorMessage: null,
                    oldInput: null,
                    validationsError: []
```

```
            });
        })
        .catch(err => {
            const error = new Error(err)
            error.httpStatusCode = 500
            return next(error)
        });
};

exports.postEditProduct = (req, res, next) => {
    const prodId = req.body.productId;
    const { title, price, description } = req.body;
    const image = req.file

    const errors = validationResult(req);
    if (!errors.isEmpty()) {
        return res.status(422).render('admin/edit-product', {
            path: '/admin/add-product',
            pageTitle: 'Post Edit',
            errorMessage: errors.array()[0].msg,
            editing: true,
            hasError: true,
            product: {
                _id: prodId,
                title: title,
                price: price,
                description: description
            },
            validationsError: errors.array()
        })
    }
    Product.findById(prodId)
        .then(product => {
            if (product.userId.toString() !== req.user._id.toString()) {
                return res.redirect('/')
            }
            product.title = title
            product.price = price
            product.description = description
            if(image){
                fileHelper.deleteFile(product.imageUrl)
                product.imageUrl = image.path
            }

            return product.save()
                .then(result => {
```

```
                res.redirect('/admin/products');
            })
            .catch(err => {
                console.log(err)
            })
    })
    .catch(err => {
        const error = new Error(err)
        error.httpStatusCode = 500
        return next(error)
    })
};

exports.getProducts = (req, res, next) => {

    Product.find({ userId: req.user._id })
    // .select('title price -_id')
    // .populate('userId', 'email')
    .then(products => {
        res.render('admin/products', {
            prods: products,
            pageTitle: 'Admin Products',
            path: '/admin/products',
        });
    })
    .catch(err => {
        const error = new Error(err)
        error.httpStatusCode = 500
        return next(error)
    });
};

exports.postDeleteProduct = (req, res, next) => {
    const prodId = req.body.productId;

    Product.findById(prodId)
    .then(product => {
        fileHelper.deleteFile(product.imageUrl)
    })
    .catch(err => {
        const error = new Error(err)
        error.httpStatusCode = 500
        return next(error)
    })

    Product.findByIdAndDelete(prodId)
```

```
        .then(result => {
            res.redirect('/admin/products');
        })
        .catch(err => {
            const error = new Error(err)
            error.httpStatusCode = 500
            return next(error)
        });
};

exports.deleteProduct = (req, res, next) => {

    const prodId = req.params.productId;

    Product.findById(prodId)
        .then(product => {
            fileHelper.deleteFile(product.imageUrl)
        })
        .catch(err => {
            const error = new Error(err)
            error.httpStatusCode = 500
            return next(error)
        })

    Product.findByIdAndDelete(prodId)
        .then(result => {
            res.status(200).json({ message: "success" })
        })
        .catch(err => {
            res.status(500).json({message: "Daleting faild"})
        });
}
```

ДОДАТОК В

Auth Controller

```
const transport = nodeMailer.createTransport(sendgridTranport({
    auth: {
        api_key: 'SG.te8a0t7cTmCI3qAlDiomyQ.hGc1JyN1ip5wK7IHfrNuiFC
M9ZO7x7EeCxABiKOwFzQ'
    }
}));

exports.getLogin = (req, res, next) => {
    res.render('auth/login', {
        path: 'login',
        pageTitle: 'Login',
        errorMessage: req.flash('error')[0],
        oldInput: undefined,
        validationsError: []
    })
}

exports.postLogin = (req, res, next) => {
    const { email, password } = req.body;

    const errors = validationResult(req);
    if (!errors.isEmpty()) {
        return res.status(422).render('auth/login', {
            path: 'login',
            pageTitle: 'Login',
            errorMessage: errors.array()[0].msg,
            oldInput: {
                email: email,
                password: password
            },
            validationsError: errors.array()
        })
    }

    User.findOne({ email: email })
        .then(user => {
            if (user) {
                bcryptjs.compare(password, user.password)
                .then(result => {
                        if (result) {
                            req.session.user = user
                            req.session.isLoggedIn = true
```

```
                        return req.session.save(err => {
                            console.log(err)
                            res.redirect('/')
                        })
                    }
                    return res.status(422).render('auth/login', {
                        path: 'login',
                        pageTitle: 'Login',
                        errorMessage: 'Invalid email or password',
                        oldInput: {
                            email: email,
                            password: password
                        },
                        validationsError: []
                    })
                })
                .catch(err => {
                    console.log(err)
                    res.redirect('/login')
                })

            } else {
                res.redirect('/login')
            }
        })
        .catch(err => {
            const error = new Error(err)
            error.httpStatusCode = 500
            return next(error)
        })
}

exports.postLogout = (req, res, next) => {
    req.session.destroy(() => {
        res.redirect('/')
    })
}

exports.getSignUp = (req, res, next) => {
    res.render('auth/signup', {
        path: 'signup',
        pageTitle: 'Sign Up',
        errorMessage: req.flash('error')[0],
        oldInput: undefined,
        validationsError: []
    })
```

```
}

exports.postSignUp = (req, res, next) => {
    let { email, password, confirmPassword } = req.body;

    const errors = validationResult(req);
    console.log(errors.array())
    if (!errors.isEmpty()) {
        return res.status(422).render('auth/signup', {
            path: 'signup',
            pageTitle: 'Sign Up',
            errorMessage: errors.array()[0].msg,
            oldInput: {
                email: email,
                password: password
            },
            validationsError: errors.array()
        })
    }

    bcryptjs.hash(password, 12)
        .then(hashPassword => {
            if (hashPassword) {
                let newUser = new User({
                    name: email,
                    email: email,
                    password: hashPassword,
                    cart: { items: [] }
                })
                return newUser.save()
            }
        })
        .then(result => {
            res.redirect('/login')
            return transport.sendMail({
                to: email,
                from: 'kanashinskydmitry@gmail.com',
                subject: 'SignUp successed',
                html: '<h1>You seccessfully signed up!</h1>'
            })
                .then(result => {
                    console.log(result)
                })
                .catch(err => {
                    console.log(err)
                })
```

```
        })
        .catch(err => {
            const error = new Error(err)
            error.httpStatusCode = 500
            return next(error)
        })
    }

    exports.getResetPassword = (req, res, next) => {
        res.render('auth/reset', {
            path: 'reset',
            pageTitle: 'Reset Password',
            errorMessage: req.flash('error')[0]
        })
    }

    exports.postResetPassword = (req, res, next) => {
        let { email } = req.body;
        crypto.randomBytes(32, (err, buffer) => {
            if (err) {
                console.log(err)
                return res.redirect('/resetPassword')
            }
            const token = buffer.toString('hex')

            User.findOne({ email: email })
                .then(user => {
                    console.log(user)
                    if (!user) {
                        req.flash('error', "No accont with that email found.")
                        return res.redirect('/resetPassword')
                    }
                    user.resetToken = token
                    user.resetTokenExpiration = Date.now() + 3600000
                    return user.save()
                })
                .then(user => {
                    res.redirect('/')
                    transport.sendMail({
                        to: user.email,
                        from: 'kanashinskydmitry@gmail.com',
                        subject: 'Password reset',
                        html: `
                            <p>You requested a password reset</p>
                            <p>Click this <a href="http://localhost:3000/resetPassword/${token}">link</a> toooo set a new password</p>
```

```
                `
            })
                .then(result => {
                    console.log('result', result)
                })
                .catch(err => {
                    console.log('err', err)
                })
        })
        .catch(err => {
            const error = new Error(err)
            error.httpStatusCode = 500
            return next(error)
        })
    })
}

exports.getNewPassword = (req, res, next) => {

    const token = req.params.token;
    User.findOne({
        resetToken: token,
        resetTokenExpiration: { $gt: Date.now() }
    })
        .then(user => {
            res.render('auth/new-password', {
                path: 'new-password',
                pageTitle: 'New Password',
                errorMessage: req.flash('error')[0],
                userId: user._id.toString(),
                passwordToken: token
            })
        })
        .catch(err => {
            const error = new Error(err)
            error.httpStatusCode = 500
            return next(error)
        })

}
```

ДОДАТОК Г

ShopController

```
const ITEMS_PER_PAGE = 5

exports.getProducts = (req, res, next) => {
  const page = req.query.page ? parseInt(req.query.page) : 1
  let countProduct;

  Product.find().countDocuments()
    .then(count => {
      countProduct = count
      return Product.find()
        .skip((page - 1) * ITEMS_PER_PAGE) // pagination in mongoose
        .limit(ITEMS_PER_PAGE)
    })
    .then(products => {
      console.log(products)
      res.render('shop/product-list', {
        prods: products,
        pageTitle: 'Products',
        path: '/products',
        countProduct: countProduct,
        currentPage: page,
        hasNextPage: ITEMS_PER_PAGE * page < countProduct,
        hasPreviosPage: page > 1,
        nextPage: page + 1,
        previosPage: page - 1,
        lastPage: Math.ceil(countProduct / ITEMS_PER_PAGE)
      });
    })
    .catch(err => {
      return next(err)
    })
};

exports.getProduct = (req, res, next) => {
  const prodId = req.params.productId;

  Product.findById(prodId)
    .then(product => {
      res.render('shop/product-detail', {
        product: product,
        pageTitle: product.title,
        path: '/products',
```

```
      });
    })
    .catch(err => {
      const error = new Error(err)
      error.httpStatusCode = 500
      return next(error)
    });
};

exports.getIndex = (req, res, next) => {
  // let isLoggedIn = req.get('Cookie').split('=')[1] === 'true'
  const page = req.query.page ? parseInt(req.query.page) : 1
  let countProduct;

  Product.find().countDocuments()
    .then(count => {
      countProduct = count
      return Product.find()
        .skip((page - 1) * ITEMS_PER_PAGE) // pagination in mongoose
        .limit(ITEMS_PER_PAGE)
    })
    .then(products => {
      console.log(products)
      res.render('shop/index', {
        prods: products,
        pageTitle: 'Shop',
        path: '/',
        countProduct: countProduct,
        currentPage: page,
        hasNextPage: ITEMS_PER_PAGE * page < countProduct,
        hasPreviosPage: page > 1,
        nextPage: page + 1,
        previosPage: page - 1,
        lastPage: Math.ceil(countProduct / ITEMS_PER_PAGE)
      });
    })
    .catch(err => {
      return next(err)
    })
};

exports.getCart = (req, res, next) => {

  req.user
    .populate('cart.items.productId')
    .execPopulate()
```

```
    .then(user => {
      if (user.cart.items.length > 0) {
        res.render('shop/cart', {
          path: '/cart',
          pageTitle: 'Your Cart',
          products: user.cart.items,
        });
      } else {
        res.render('shop/cart', {
          path: '/cart',
          pageTitle: 'Your Cart',
          products: [],
        });
      }
    })
    .catch(err => {
      const error = new Error(err)
      error.httpStatusCode = 500
      return next(error)
    })
};

exports.postCart = (req, res, next) => {
  const prodId = req.body.productId;

  Product.findById(prodId)
    .then(product => {
      return req.user.addToCart(product)
        .then(result => {
          res.redirect('/cart');
        })
        .catch(err => {
          const error = new Error(err)
          error.httpStatusCode = 500
          return next(error)
        })

    })
    .catch(err => {
      const error = new Error(err)
      error.httpStatusCode = 500
      return next(error)
    })

};
```

```
exports.postCartDeleteProduct = (req, res, next) => {
  const prodId = req.body.productId;

  req.user.deleteCartProduct(prodId)
    .then(result => {
      res.redirect('/cart')
    })
    .catch(err => {
      const error = new Error(err)
      error.httpStatusCode = 500
      return next(error)
    })

};

exports.postCreateOrder = (req, res, next) => {
  const token = req.body.stripeToken; // Using Express
  let totalSum = 0
  req.user
    .populate('cart.items.productId')
    .execPopulate()
    .then(user => {

      user.cart.items.forEach(p => {
        totalSum += p.quantity * p.productId.price
      })

      const products = user.cart.items.map(i => {
        return { quantity: i.quantity, product: { ...i.productId._doc } }
      });

      const order = new Order({
        user: {
          name: req.user.name,
          userId: req.user._id
        },

        products: products
      })

      return order.save()
    })
    .then(result => {
      return req.user.clearCart()
    })
    .then(result => {
```

```
      (async () => {
        const charge = await stripe.charges.create({
          amount: totalSum * 100,
          currency: 'usd',
          description: 'Demo Order))',
          source: token,
        });
      })();

      res.redirect('/orders')
    })
    .catch(err => {
      const error = new Error(err)
      error.httpStatusCode = 500
      return next(error)
    })
}

exports.getOrders = (req, res, next) => {

  Order.find({ 'user.userId': req.user._id })
    .then(orders => {
      res.render('shop/orders', {
        path: '/orders',
        pageTitle: 'Your Orders',
        orders: orders,
      });
    })
    .catch(err => {
      const error = new Error(err)
      error.httpStatusCode = 500
      return next(error)
    })
};

exports.getInvoice = (req, res, next) => {
  const { orderId } = req.params
  const invoiceName = 'invoice-' + orderId + '.pdf'
  const invoicePath = path.join('data', 'invoice', invoiceName)

  Order.findById(orderId)
    .then(order => {
      if (!order) {
        return next(new Error('No order found'))
      }
```

```
            if (order.user.userId.toString() !== req.user._id.toString()) {
              return next(new Error('Unauthorized'))
            }
            // fs.readFile(invoicePath, (err, data) => {
            //   if (err) {
            //     console.log("Error", err)
            //     return next(err)
            //   }
            //   res.setHeader('Content-Type', 'application/pdf')
            //   res.setHeader('Content-Disposition', 'inline; fileName=Okkkk')
            //   res.send(data)
            // })

            const pdfDoc = new PDFDocument()

            res.setHeader('Content-Type', 'application/pdf')
            res.setHeader('Content-Disposition', 'inline; fileName=' + invoiceName)

            pdfDoc.pipe(fs.createWriteStream(invoicePath))
            pdfDoc.pipe(res)
            let totalPrice = 0
            pdfDoc.fontSize(25).text('Invoice', {
              underline: true
            })

            pdfDoc.text('----------------------')

            order.products.forEach(prod => {
              totalPrice += prod.quantity * prod.product.price
              pdfDoc.fontSize(14).text(prod.product.title + ' - ' + prod.quantity + ' x '
+ '$' + prod.product.price)
            })

            pdfDoc.fontSize(20).text('\nTotal price: ' + totalPrice)

            pdfDoc.end()

            // const file = fs.createReadStream(invoicePath)
            // file.pipe(res)
          })
          .catch(err => {
            return next(err)
          })

        }
```

```
exports.getCheckout = (req, res, next) => {
  req.user
    .populate('cart.items.productId')
    .execPopulate()
    .then(user => {
      let products = user.cart.items

      let total = 0
      products.forEach(p => {
        total += p.quantity * p.productId.price
      })

      if (user.cart.items.length > 0) {
        res.render('shop/checkout', {
          path: '/checkout',
          pageTitle: 'Your Checkout',
          products: products,
          totalSum: total
        });
      } else {
        res.render('shop/checkout', {
          path: '/checkout',
          pageTitle: 'Your Checkout',
          products: [],
        });
      }
    })
```

ДОДАТОК Д

Product List View

```
<%- include('../includes/head.ejs') %>
<link rel="stylesheet" href="/css/product.css">
</head>

<body>
    <%- include('../includes/navigation.ejs') %>

    <main>
        <% if (prods.length > 0) { %>
        <div class="grid">
            <% for (let product of prods) { %>
            <article class="card product-item">
                <header class="card__header">
                    <h1 class="product__title">
                        <%= product.title %>
                    </h1>
                </header>
                <div class="card__image">
                    <img src="/<%= product.imageUrl %>" alt="<%= product.title %>">
                </div>
                <div class="card__content">
                    <h2 class="product__price">$
                        <%= product.price %>
                    </h2>
                    <p class="product__description">
                        <%= product.description %>
                    </p>
                </div>
                <div class="card__actions">
                    <a href="/products/<%= product._id %>" class="btn">Details</a>
                    <% if(isAuthenticated) { %>
                    <%- include('../includes/add-to-cart.ejs', {product: product}) %>
                    <% } %>
                </div>
            </article>
            <% } %>
        </div>
        <section class="pagination">
            <% if(currentPage !== 1) {%> <a href="/?page=1">1 ... </a> <% } %>

            <% if(hasPreviosPage && previosPage !== 1){%>
            <a href="/?page=<%= previosPage%>"><%= previosPage %></a>
```

```
        <% } %>

        <a href="/?page=<%= currentPage%>"><%= currentPage %></a>

        <% if(hasNextPage){%>
        <a href="/?page=<%= nextPage%>"><%= nextPage %></a>
        <% } %>

        <% if(lastPage !== currentPage && lastPage !== nextPage){%>
        <a href="/?page=<%= lastPage%>"> ... <%= lastPage %></a>
        <% } %>
    </section>
    <% } else { %>
    <h1>No Products Found!</h1>
    <% } %>
</main>
<%- include('../includes/end.ejs') %>
```

ДОДАТОК Е

Phyton script

```
        import pandas as pd
        import numpy as np
        from fastai.tabular import *
        import pickle

        if os.path.getsize('/home/anshch/Documents/data_2.pkl') > 0:
            with open('/home/anshch/Documents/data_2.pkl', "rb") as f:
                unpickler = pickle.Unpickler(f)
                # if file is not empty scores will be equal
                # to the value unpickled
                data = unpickler.load()

        data = data.reset_index()
        data.head()

data = data[[
            'date_block_num',
            'shop_id',
            'shop_category',
            'item_id',
            'item_cnt_month',
            'city_code',
            'item_category_id',
            'type_code',
            'subtype_code',
            'item_cnt_month_lag_1',
            'item_cnt_month_lag_2',
            'item_cnt_month_lag_3',
            'item_cnt_month_lag_4',
            'item_cnt_month_lag_5',
            'item_cnt_month_lag_6',
            'item_cnt_month_lag_12',
            'date_avg_item_cnt_lag_1',
            'date_item_avg_item_cnt_lag_1',
            'date_item_avg_item_cnt_lag_2',
            'date_item_avg_item_cnt_lag_3',
            'date_cat_avg_item_cnt_lag_1',
            'date_shop_cat_avg_item_cnt_lag_1',
            'date_shop_type_avg_item_cnt_lag_1',
            'date_shop_subtype_avg_item_cnt_lag_1',
            'date_city_avg_item_cnt_lag_1',
            'date_item_city_avg_item_cnt_lag_1',
```

```
            'date_type_avg_item_cnt_lag_1',
            'date_subtype_avg_item_cnt_lag_1',
            'delta_price_lag',
            'month',
            'item_shop_last_sale',
            'item_last_sale',
            'item_shop_first_sale',
            'item_first_sale',
            'country_part',
            'weeknd_count',
            'days_in_month'
        ]]

        data["log_sales"] = np.log(data.item_cnt_month + 1)
        data["log_sales"].describe()

        # Split data

        X_train = data[data.date_block_num <= 33]
        X_train = data[data.item_cnt_month>0]
        X_train = X_train.reset_index(drop=True)
        # X_valid = data[data.date_block_num == 33]
        X_test = data[data.date_block_num == 34]

        #Start index for creating a validation set from train_data
        start_indx = X_train[X_train.date_block_num == 33].index[0]

        #End index for creating a validation set from train_data
        end_indx = max(X_train[X_train.date_block_num == 33].index)
        print(start_indx, end_indx)
        # Clean env
        import gc

        del data
        gc.collect()
        # Define feature type
        path = ''

        dep_var = 'log_sales'

        cat_feature = ["shop_id", "shop_category", "item_id", "city_code",
"item_category_id",
                       "type_code", "subtype_code", 'month', "country_part"]

        con_feature = ['item_cnt_month_lag_1',
'item_cnt_month_lag_2','item_cnt_month_lag_3',
```

```
                'item_cnt_month_lag_4', 'item_cnt_month_lag_5',
'item_cnt_month_lag_6',
                'item_cnt_month_lag_12', 'date_avg_item_cnt_lag_1',
'date_item_avg_item_cnt_lag_1',
                'date_item_avg_item_cnt_lag_2', 'date_item_avg_item_cnt_lag_3',
'date_item_avg_item_cnt_lag_6',
                'date_item_avg_item_cnt_lag_12',
'date_shop_avg_item_cnt_lag_1', 'date_shop_avg_item_cnt_lag_2',
                'date_shop_avg_item_cnt_lag_3', 'date_shop_avg_item_cnt_lag_6',
'date_shop_avg_item_cnt_lag_12',
                'date_cat_avg_item_cnt_lag_1',
'date_shop_cat_avg_item_cnt_lag_1',
                'date_shop_type_avg_item_cnt_lag_1',
'date_shop_subtype_avg_item_cnt_lag_1',
                'date_city_avg_item_cnt_lag_1',
'date_item_city_avg_item_cnt_lag_1',
                'date_type_avg_item_cnt_lag_1',
'date_subtype_avg_item_cnt_lag_1', 'delta_price_lag',
                'item_shop_last_sale', 'item_last_sale','item_shop_first_sale',
                'item_first_sale', 'city_coord_1', 'city_coord_2', 'weeknd_count',
'days_in_month']
        #List of Processes/transforms to be applied to the dataset
        procs = [FillMissing, Categorify, Normalize]
        X_train[con_feature] = X_train[con_feature].astype('int64')
        X_test[con_feature] = X_test[con_feature].astype('int64')
        #TabularList for Validation
        val = (TabularList.from_df(X_train.iloc[start_indx:end_indx].copy(),
path=path, cat_names=cat_feature, cont_names=con_feature))

        test = (TabularList.from_df(X_test, path=path, cat_names=cat_feature,
cont_names=con_feature, procs=procs))

        #TabularList for training
        data = (TabularList.from_df(X_train, path=path, cat_names=cat_feature,
cont_names=con_feature, procs=procs)
                            .split_by_idx(list(range(start_indx,end_indx)))
                            .label_from_df(cols=dep_var)
                            .add_test(test)
                            .databunch())
        #Display the data batch
        data.show_batch(rows = 5)
        #Initializing the network
        learn = tabular_learner(data, layers=[1024,512], metrics= [rmse,r2_score])
        #Initializing the network
        learn = tabular_learner(data, layers=[1024,512], metrics= [rmse,r2_score])
        #Predicting For The Complete Test set
```

```
test_predictions = learn.get_preds(ds_type=DatasetType.Test)[0]

#Converting the tensor output to a list of predicted values
test_predictions = [i[0] for i in test_predictions.tolist()]
if os.path.getsize('/home/anshch/Documents/test.pkl') > 0:
    with open('/home/anshch/Documents/test.pkl', "rb") as f:
        unpickler = pickle.Unpickler(f)
        # if file is not empty scores will be equal
        # to the value unpickled
        test = unpickler.load()
test.head()
submission = pd.DataFrame({
    "ID": test.index,
    "item_cnt_month": np.exp(test_predictions)-1
})
# submission[submission < 0] = 0
submission.to_csv('/home/anshch/Documents/fastai_nn.csv', index=False)
```

ДОДАТОК Ж

Таблиця Ж.1 — Прогнозування попиту торгової групи соки

Тиждень	Прогноз	Факт	Новий прогноз
1	2	3	4
1	1712,62	1712,62	1883,88
2	1712,62	1792,76	1730,78
3	1792,76	137,38	137,38
4	1671,41	2012,56	1271,91
5	2012,56	1671,41	3035,38
6	1900,37	4189,97	13,50
7	2129,33	1900,37	2855,35
8	1900,37	2129,33	1681,72
9	1785,89	1785,89	1964,48
10	1785,89	1602,72	2372,68
11	1831,68	1831,68	2014,85
12	2129,33	2129,33	2342,26
13	2552,77	2552,77	2808,05
14	2552,77	2672,22	2579,83
15	2672,22	204,77	204,77
16	2491,34	2999,85	1895,86
17	2999,85	2491,34	4524,43
18	2832,62	4245,42	1230,59
19	3173,90	2832,62	4256,09
20	2832,62	3173,90	2506,72
21	2730,24	2661,98	3143,28
22	2661,98	2730,24	2730,24
23	2730,24	2388,96	3783,33
24	3173,90	3173,90	3491,29
25	3489,87	3489,87	3838,86
26	3489,87	3653,16	3526,86
27	3653,16	293,93	393,93
28	3265,92	4101,06	2262,36
29	4101,06	3265,92	6608,57
30	3872,45	5538,05	1930,37
31	4339,01	3872,45	5818,45
32	3872,45	4339,01	3426,91
33	3797,80	3639,17	4508,67
34	3639,17	3797,80	3699,08
35	3797,80	3326,57	5253,53
36	4339,01	4339,01	4772,91
37	4782,41	4782,41	5260,65
38	4782,41	5006,19	4833,11
39	5006,19	402,80	402,80
40	4475,52	5619,97	3100,28
41	5619,97	4475,52	9056,19
42	5306,69	8979,01	1496,60
43	5306,69	5306,69	5837,36
44	4987,01	4667,33	6168,86
45	4987,01	4987,01	5485,71
46	4987,01	5204,39	5069,10
47	5204,39	4558,64	7199,29

Закінчення таблиці Ж.1

48	5252,34	5946,05	4552,03
23	2730,24	2388,96	3783,33
24	3173,90	3173,90	3491,29
25	3489,87	3489,87	3838,86
26	3489,87	3653,16	3526,86
27	3653,16	293,93	393,93
28	3265,92	4101,06	2262,36
29	4101,06	3265,92	6608,57
30	3872,45	5538,05	1930,37
31	4339,01	3872,45	5818,45
32	3872,45	4339,01	3426,91
33	3797,80	3639,17	4508,67
34	3639,17	3797,80	3699,08
35	3797,80	3326,57	5253,53
36	4339,01	4339,01	4772,91
37	4782,41	4782,41	5260,65
38	4782,41	5006,19	4833,11
39	5006,19	402,80	402,80
40	4475,52	5619,97	3100,28
41	5619,97	4475,52	9056,19
42	5306,69	8979,01	1496,60
43	5306,69	5306,69	5837,36
44	4987,01	4667,33	6168,86
45	4987,01	4987,01	5485,71
46	4987,01	5204,39	5069,10
47	5204,39	4558,64	7199,29
48	5252,34	5946,05	4552,03

ДОДАТОК К

ПРОТОКОЛ ПЕРЕВІРКИ НАВЧАЛЬНОЇ (КВАЛІФІКАЦІЙНОЇ) РОБОТИ

Назва роботи: Інформаційна технологія для підтримки інтернет торгівлі з прогнозуванням попиту на основі аналізу статистики попередніх продаж.

Тип роботи: магістерська кваліфікаційна робота

(кваліфікаційна роботи, курсовий проект (робота), реферат, аналітичний огляд, інше (зазначити))

Підрозділ кафедра обчислювальної техніки

(кафедра, факультет (інститут), навчальна група)

Науковий керівник к.т.н., доц. Ткаченко О. М.

(прізвище, ініціали, посада)

Показники звіту подібності

Plagiat.pl (StrikePlagiarism)		Unicheck	
КП1			
КП2		Оригінальність	80
Тривога/Білі знаки	/	Схожість	20

Аналіз звіту подібності (відмінити подібне)

- ✓ Запозичення, виявлення у роботі, оформлені коректно і не містять ознак плагіату.
- □ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності і відсутності самостійності її автора. Робот направити на доопрацювання.
- □ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Заявляю, що ознайомлений(-на) з повним звітом подібності, який був згенерований Системою щодо роботи (додається)

Автор _____ Каташинський Д. О.

(підпис) (прізвище, ініціали)

Опис прийнятого рішення

Ступінь оригінальності роботи відповідає вимогам, що висуваються до МКР

Особа, відповідальна за перевірку_____ Захарченко С.М.

(підпис) (прізвище, ініціали)

Експерт _____

(за потреби) (підпис) (прізвище, ініціали)