НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

«На правах рукопису»
УДК _004.75_

До захисту допущено:
Завідувач кафедри
_____ Олександр РОЛІК
«___»_____2024 р.

Магістерська дисертація

на здобуття ступеня магістра

за освітньо-професійною програмою
«Інформаційні управляючі системи та технології»

зі спеціальності 126 «Інформаційні системи та технології»

на тему: «Інформаційна система підтримки діяльності служби доставки продуктів харчування»

Виконав:
студент 2 курсу, групи ІС-32мп
Аносов Ігор Андрійович _____

Керівник:
професор каф. ІСТ, д.т.н., проф.
Теленик Сергій Федорович _____

Рецензент:
доцент каф. ОТ, к.т.н., доц.
Волокита Артем Миколайович _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2024 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Рівень вищої освіти – другий (магістерський)

Спеціальність – 126 «Інформаційні системи та технології»

Освітньо-професійна програма «Інформаційні управляючі системи та технології»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр РОЛІК
«___»_____2024 р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Аносову Ігорю Андрійовичу

1. Тема дисертації «Інформаційна система підтримки діяльності служби доставки продуктів харчування», науковий керівник дисертації Теленик Сергій Федорович, д.т.н., проф., затверджені наказом по університету від «08» 11 2024 р. № 5016-с

2. Термін подання студентом дисертації «09» 12 2024 р.

3. Об’єкт дослідження: процес діяльності служби доставки продуктів харчування.

4. Вихідні дані: технічне завдання на розробку інформаційної системи підтримки діяльності служби доставки продуктів харчування.

5. Перелік завдань, які потрібно розробити: провести аналіз предметної області й вивчити існуючі рішення; визначити підходи до виконання бізнес-процесів на основі BPMN; спроєктувати базу даних; розробити телеграм-бот для комунікації з користувачами; розробити програмне забезпечення інформаційної системи. підготувати текстову та графічну частину пояснювальної записки

6. Орієнтовний перелік графічного (ілюстративного) матеріалу: структурна схема, діаграма варіантів використання, діаграма діяльності, ER-діаграма.

7. Орієнтовний перелік публікацій «Розвиток і реалізація технології створення широкого класу застосувань на зразок чат-ботів на основі формальних моделей».

8. Дата видачі завдання 02.09.2024 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1.	Аналіз існуючих рішень	21.09.2024	
2.	Формулювання функціональних/не функціональних вимог до системи	28.09.2024	
3.	Реалізація програмного забезпечення	05.10.2024	
4.	Оформлення роботи	09.11.2024	
5.	Подання роботи на попередній захист		
6.	Подання роботи на основний захист		

Студент Ігор АНОСОВ

Науковий керівник Сергій ТЕЛЕНИК

РЕФЕРАТ

Інформаційна система підтримки діяльності служби доставки продуктів харчування: 101 с., 29 табл., 23 рис., 10 дод., 15 джерел.

ІНФОРМАЦІЙНА СИСТЕМА, ДОСТАВКА ЇЖІ, BPMN, БІЗНЕС-ПРОЦЕСИ, АВТОМАТИЗАЦІЯ, ТЕЛЕГРАМ-БОТ.

Актуальність. Сучасний ринок доставки їжі характеризується високим рівнем конкуренції та постійним зростанням обсягів послуг. Це вимагає автоматизації процесів для підвищення швидкості, точності виконання замовлень і мінімізації витрат ресурсів.

Метою роботи є створення інформаційної системи для автоматизації процесів доставки їжі з використанням BPMN-діаграм. Для досягнення цієї мети необхідно виконати такі задачі: провести аналіз предметної області й вивчити існуючі рішення; визначити підходи до виконання бізнес-процесів на основі BPMN; спроектувати базу даних; розробити телеграм-бот для комунікації з користувачами; розробити програмне забезпечення інформаційної системи.

Об’єкт дослідження. Процеси діяльності служби доставки продуктів харчування.

Предмет дослідження. Технології автоматизації бізнес-процесів доставки їжі з використанням BPMN.

Результати досліджень, реалізації, застосування інформаційної системи були представлені в статті «Розвиток і реалізація технології створення широкого класу застосувань на зразок чат-ботів на основі формальних моделей», що вийшла в науковому журналі “Наукові записки НаУКМА. Комп'ютерні науки”. У статті викладено теоретичні основи та практичні реалізації запропонованої структури, запропоновано комплексний огляд її впливу та масштабованості.

ABSTRACT

Information System to Support the Operations of a Food Delivery Service: 101 p., 29 tab., 23 draw., 9 app., 9 sources.

INFORMATION SYSTEM, FOOD DELIVERY, BPMN, BUSINESS PROCESSES, AUTOMATION, TELEGRAM BOT.

Relevance. The modern food delivery market is characterized by a high level of competition and continuously increasing service volumes. This necessitates the automation of processes to enhance the speed and accuracy of order fulfillment while minimizing resource expenditures.

Objective. The goal of this work is to develop an information system for automating food delivery processes using BPMN diagrams. To achieve this objective, the following tasks must be completed: analyze the subject area and study existing solutions; define approaches for executing business processes based on BPMN; design a database; develop a Telegram bot for user communication; and create software for the information system.

Object of Research. The operational processes of a food delivery service.

Subject of Research. Technologies for automating food delivery business processes using BPMN.

The research results, implementation, and application of the information system were presented in the article "Development and Implementation of a Technology for Creating a Broad Class of Applications Similar to Chatbots Based on Formal Models," published in the scientific journal *“Scientific Notes of NaUKMA. Computer Science.”* The article outlines the theoretical foundations and practical implementations of the proposed structure, providing a comprehensive overview of its impact and scalability.

ЗМІСТ

ВСТУП

У сучасному світі сектор послуг з доставки їжі зазнав швидкого зростання завдяки зростаючому попиту на зручність та швидкість. Щоб залишатися конкурентоспроможними, підприємства в цій галузі повинні впроваджувати передові інформаційні системи, які впорядковують свою діяльність, забезпечуючи ефективне управління замовленнями та доставку.

Початково такою автоматизацією займались лише великі гравці на ринку. Деякі підприємства також почали створювати собі сайти і автоматизовувати процес замовлення і доставки. А малі підприємства покладались здебільшого на телефонні замовлення. Але не так давно стався розквіт застосунків агрегаторів доставки їжі. І тепер аби залишатися конкурентоспроможним малий бізнес має також впроваджувати рішення для автоматизації своїх процесів замовлення та доставки.

Актуальність розробки обумовлена необхідністю зниження витрат та підвищення ефективності процесів у секторі доставки їжі, особливо для малих і середніх підприємств. Високий рівень конкуренції на ринку послуг із доставки вимагає запровадження рішень, які дозволяють швидко адаптуватися до змін у ринкових умовах, зберігаючи якість обслуговування клієнтів. Запропонована система базується на використанні BPMN-діаграм для опису бізнес-процесів і їх автоматизації, що значно спрощує створення та підтримку таких систем.

Метою створення цієї системи є здешевлення автоматизації процесу доставки, полегшення та прискорення внесення змін до нього.

Об’єктом дослідження є процеси діяльності служби доставки продуктів харчування, які вимагають автоматизації для підвищення ефективності та зниження витрат.

Предметом дослідження є технології автоматизації бізнес-процесів доставки їжі з використанням BPMN-діаграм.

Одне з таких рішень передбачає інтеграцію динамічних програмних засобів, які автоматизують процеси, впорядковують комунікацію з клієнтами, оптимізують

управління робочим процесом, зменшують кількість часу необхідну для внесенні змін в процесс.

Розроблена система може бути використана малими та середніми підприємствами для автоматизації процесів замовлення та доставки їжі, що дозволить підвищити ефективність їхньої роботи, знизити витрати на операційні процеси та покращити якість обслуговування клієнтів.

Результати досліджень були представлені в статті «Розвиток і реалізація технології створення широкого класу застосувань на зразок чат-ботів на основі формальних моделей», опублікованій у науковому журналі "Наукові записки НаУКМА. Комп'ютерні науки". У статті описано теоретичні основи та практичну реалізацію запропонованої структури, включаючи її масштабованість та вплив на ефективність.

Магістерська дисертація складається зі вступу, основних розділів, висновків, списку використаних джерел (15 найменувань) та 10 додатків. Загальний обсяг роботи — 101 сторінок, включаючи 29 таблиць і 23 рисунки.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд існуючих аналогів

Розробка ефективної інформаційної системи для служби доставки їжі вимагає оцінки різних технологічних рішень. Можливі варіанти: платформи-агрегатори ресторанів, фірмові веб-сайти та рішення для автоматизації процесу доставки за допомогою соціальних мереж.

Платформи-агрегатори ресторанів – це сторонні сервіси, які з'єднують кілька ресторанів з клієнтами через єдиний інтерфейс, обробляючи розміщення замовлень, обробку платежів та логістику доставки від імені підприємств-учасників. Приклади включають Glovo і UberEats. Хоча вони пропонують негайний доступ до ринку та логістичну підтримку, вони часто включають значні комісії та забезпечують менший контроль над відносинами з клієнтами.

Власні веб-сайти передбачають створення спеціальної онлайн-платформи для служби доставки їжі, яка пропонує повний контроль над брендом, користувацьким досвідом та даними клієнтів. Компанії можуть адаптувати функції веб-сайту для задоволення конкретних операційних потреб. Прикладом є сервіс онлайн-доставки «Сільпо», який дозволяє без посередницьких перешкод безпосередньо взаємодіяти з клієнтами.

Рішення для автоматизації чату використовують платформи обміну повідомленнями та чат-боти для взаємодії з клієнтами, обробки замовлень та надання підтримки. Такі платформи, як Chatfuel, дозволяють компаніям створювати власні чат-боти, використовуючи існуючі канали зв'язку для економічно ефективної автоматизації та розширеної доступності для користувачів, знайомих з програмами обміну повідомленнями.

Кожен з цих варіантів представляє унікальні переваги та проблеми, що впливають на такі фактори, як вартість, контроль даних клієнтів, досвід користувачів та логістичні вимоги. Комплексний аналіз має важливе значення для

визначення найбільш відповідного рішення на основі конкретних цілей і ресурсів служби доставки їжі.

Огляд аналогів подібних ІС наведено в таблиці 1.1.

Таблиця 1.1 – Аналіз приведених аналогів

Параметри	Glovo	Silpo	Chatfuel
Легкість налаштування	Потребує реєстрації та інтеграції, може зайняти час	Високі витрати часу та ресурсів на розробку та підтримку сайту	Потребує налаштування; може бути складним для некваліфікованих користувачів
Витрати та комісії	Високі комісії за кожне замовлення	Високі витрати на розробку та підтримку	Низькі початкові витрати; можливі плати за використання платформи чат-бота
Легкість для користувачів	Треба встановити спеціальний додаток; Може не працювати на застарілих телефонах	Потрібно відвідати сайт; може бути менш зручно на мобільних пристроях	Користувачі використовують знайомі месенджери; зручно та швидко
Простота інтеграції та підтримки	Складно; підпорядкована правилам і умовам Glovo	Складно; потребує технічних знань та ресурсів	Складна інтеграція з існуючими системами; Помірна складність підтримки

1.2 Критичний аналіз приведених аналогів

У таблиці 1.1 було проведено критичний аналіз за наступними критеріями:

- легкість налаштування;
- витрати та комісії;
- легкість для користувачів;
- простота інтеграції та підтримки.

Слабкими місцями інформаційних систем для доставки їжі є складність розробки, складність підтримки та швидкість внесення змін до системи, витрати для найму спеціалістів.

Незмінні зарегульовані етапи замовлення агрегаторів ресторанів підходять не для кожного бізнесу. А потреба найняти спеціаліста для побудови свого веб сайту і подальшого впровадження змін в процес доставки уповільнює процес і потребує додаткових коштів.

Застосунки конструктори чат ботів мають значні переваги в гнучкості і швидкості впровадження змін, але з ними складно інтегрувати вже існуючі системи і не кожен застосунок це підтримує.

Висновки до розділу

В ході аналізу аналогів виявлено ключові аспекти впровадження різних варіантів рішень автоматизації процесу доставки їжі. Згідно результатів проаналізовані інформаційні системи мають ряд своїх недоліків та обмежень.

Системи в яких можна динамічно і без залучення технічних спеціалістів розробити та впровадити зміни в процес значно спрощують процес інтеграції і підтримки. Це сприяє зменшенню витрат на технічні аспекти бізнесу і поліпшує його конкурентоспроможність.

2 ФОРМУВАННЯ ВИМОГ ДО СИСТЕМИ

2.1 Опис процесу діяльності

Об'єктом автоматизації є процес замовлення та доставки бізнесу з доставки харчових продуктів. Цей процес включає ручний прийом замовлень, координацію з працівниками служби доставки та індивідуальну взаємодію з клієнтами, що може бути неефективним та схильним до помилок. Впровадивши інформаційну систему, яка автоматизує процес замовлення та доставки, бізнес може зменшити ручне навантаження та підвищити задоволеність клієнтів. Пропонована система дозволяє бізнесу швидко створити та інтегрувати чат-бота для замовлень тим самим спрощуючи та прискорюючи весь процес замовлення та доставки.

2.2 Бізнес-процеси

Основним бізнес-процесом є створення замовлення онлайн. Етапи створення замовлення можуть змінюватись з часом. Для уніфікації опису бізнес процесів краще використовувати BPMN – Business Process Model and Notation.

Додаток дозволяє бізнесу створювати користувацький Telegram-бот, не вимагаючи великих знань у програмуванні.

Автоматизуючи ці процеси, система покращує операційну ефективність, зменшує потенціал помилок та підвищує загальну задоволеність клієнтів.

Система автоматизує процес доставки їжі за допомогою створення та управління ботів Telegram за допомогою діаграм BPMN (Business Process Model and Notation). Це дозволяє адміністратору визначати та змінювати поведінку бота, а користувачам взаємодіяти з ботом відповідно до вказаних робочих процесів.

2.3 Функціональні вимоги

Функціональні вимоги описують специфічну поведінку, особливості та взаємодії, які система повинна підтримувати для досягнення своїх цілей. Ці

вимоги окреслюють дії, які користувачі можуть виконувати, такі як створення робочих процесів, взаємодія з ботами Telegram та інтеграція зовнішніх систем. Визначаючи ці функції, система забезпечує оптимізовані операції, покращений користувальницький досвід та узгодження з бізнес-цілями автоматизації процесу доставки їжі. В таблиці 2.1 наведено список функціональних вимог.

Таблиця 2.1 – Функціональні вимоги та пріоритети для різних акторів

№	Функціональна вимога	Актор	Варіант використання	Пріоритет
1	Система повинна відображати список усіх існуючих діаграм BPMN, дозволяючи адміністратору вибрати одну для редагування.	Адміністратор	Відкриває сторінку "diagrams"	Високий
2	Система повинна дозволяти адміністратору створювати нову діаграму BPMN для розробки робочих процесів і шаблонів взаємодії ботів.	Адміністратор	Натисніть кнопку, щоб "create new diagram"	Високий
3	Система повинна дозволяти адміністратору редагувати існуючу діаграму BPMN і зберігати зміни.	Адміністратор	Відкриває наявну діаграму BPMN для редагування	Високий
4	Система повинна дозволяти адміністратору призначати діаграму BPMN боту Telegram для виконання.	Адміністратор	Призначає діаграму BPMN боту Telegram	Високий

№	Функціональна вимога	Актор	Варіант використання	Пріоритет
5	Система повинна виконувати діаграми BPMN через бота Telegram, проводячи користувача через визначений робочий процес.	Користувач Telegram	Надсилає повідомлення для взаємодії з ботом	Високий
6	Бот повинен відображати користувачеві каталог товарів із актуальною інформацією про продукт.	Користувач Telegram	Надсилає запит на перегляд товарів	Високий
7	Бот повинен дозволяти користувачеві вибирати продукти та оформляти замовлення, включаючи деталі доставки.	Користувач Telegram	Заповнює деталі доставки розміщує замовлення	Високий
8	Система повинна дозволяти інтеграцію із зовнішніми каталогами продуктів для відображення точної та актуальної інформації про продукт.	Адміністратор	Налаштовує інтеграцію каталогу продуктів	Середній
9	Система повинна надавати опції для інтеграції зовнішніх систем керування замовленнями для зберігання та обробки замовлень, розміщених через бота.	Адміністратор	Налаштовує інтеграцію керування замовленнями	Середній

№	Функціональна вимога	Актор	Варіант використання	Пріоритет
10	Система повинна дозволяти запити в Інтернеті, щоб забезпечити інтеграцію із зовнішніми системами та сервісами.	система	Виконує інтернет-запити під час робочого процесу	Середній

2.4 Нефункціональні вимоги

Нефункціональні вимоги визначають атрибути якості, системні обмеження та операційні можливості, необхідні для ефективного функціонування системи. На відміну від функціональних вимог, які описують специфічну поведінку та особливості, нефункціональні вимоги гарантують, що система є надійною, масштабованою, безпечною та ремонтопридатною. Ці вимоги відіграють важливу роль у забезпеченні надійного, зручного додатку, який відповідає очікуванням продуктивності та підтримує майбутнє зростання.

В таблиці 2.2 наведено список нефункціональних вимог.

Таблиця 2.2 – Нефункціональні вимоги

№	Категорія	Вимога
1	Доступність	Система має бути доступною через сучасні браузери (Chrome) і підтримувати пристрої з адаптивним дизайном.
2	Ефективність	Система повинна дозволити адміністраторам виконувати такі завдання, як створення та призначення діаграм ботам з легкістю, мінімізація складності та забезпечення комфортного користувацького досвіду.

№	Категорія	Вимога
3	Розширюваність	Система повинна дозволяти додаткові інтеграції або вдосконалення функцій без серйозних змін архітектури.
4	Відмовостійкість	Система має обробляти несподівані введення або збої, забезпечуючи, щоб боти залишалися функціональними під час незначних збоїв у сервісах бекенд частини.
5	Підтримуваність	Зміни коду одного мікросервісу не повинні вимагати модифікації інших незалежних сервісів, дотримуючись модульної архітектури мікросервісу.
6	Портативність	Система повинна працювати на серверах Linux і Windows з мінімальними змінами конфігурації.
7	Час відгуку	Відповіді Telegram-бота на повідомлення користувача мають відбуватися протягом 5 секунд за нормальних умов завантаження.
8	Масштабованість	Система повинна масштабуватися горизонтально, додаючи більше екземплярів мікросервісів, підтримуючи майбутнє зростання без значних змін архітектури.

Висновки до розділу

У цьому розділі викладені вимоги до запропонованої системи для автоматизації процесу замовлення та доставки харчових підприємств. Висвітлено неефективність ручного управління замовленнями та переваги впровадження інформаційної системи, яка використовує Telegram-ботів та діаграми BPMN. Система надає підприємствам інструменти для створення та управління ботами, що дозволяє оптимізувати робочі процеси без великих знань програмування. Автоматизуючи ці процеси, система підвищує операційну ефективність, зменшує помилки та підвищує задоволеність клієнтів.

Основні функціональні вимоги включають створення, редагування та призначення діаграм BPMN ботам Telegram, що дозволяє користувачам взаємодіяти з ботами для таких завдань, як перегляд каталогів продуктів, розміщення замовлень та надання деталей доставки. Крім того, система підтримує інтеграцію з зовнішніми каталогами продуктів і системами управління замовленнями, забезпечуючи точну обробку даних і виконання робочого процесу.

Нефункціональні вимоги підкреслюють доступність системи, ефективність, масштабованість, відмовостійкість і розширюваність. Ці вимоги гарантують, що система надійно працює в різних умовах, підтримує майбутнє зростання та підтримує модульну архітектуру мікросервісу для зручності оновлення та обслуговування.

Забезпечуючи ці функції, система спрощує процес створення та управління ботами, дозволяючи динамічну настройку без великих знань програмування. Користувачі отримують переваги від автоматизованих взаємодій, таких як розміщення замовлень та надання адрес доставки, підвищення загальної ефективності та користувацького досвіду.

Загалом, цей розділ визначає основу для надійної та орієнтованої на користувача системи, яка відповідає потребам підприємств, які прагнуть автоматизувати та оптимізувати свої операції з доставки їжі.

3 СЦЕНАРІЇ ВИКОРИСТАННЯ СИСТЕМИ

3.1 Актори і функції

У системі визначені наступні актори: адміністратор і користувач Telegram.

Ці актори представляють основні ролі, що взаємодіють з системою, визначаючи окремі дії та обов'язки в процесі автоматизації доставки їжі.

Дії Адміністратора:

- переглянути список всіх діаграм BPMN;
- створити нову діаграму BPMN;
- редагувати існуючу діаграму BPMN;
- зберегти зміни в діаграмі BPMN;
- призначити BPMN діаграму на виконання Telegram ботом;
- налаштування інтеграцій з зовнішніми сервісами, з каталогом продуктів тощо.

Дії користувача Telegram:

- переглянути доступну продукцію;
- вибрати продукти для замовлення;
- надати деталі доставки замовлення;
- оформити замовлення через бота.

3.2 Опис варіантів використання

Ключові випадки використання системи, зосереджені на двох основних областях: управління діаграмами BPMN через панель діаграм та взаємодія з ботом Telegram. Адміністратори використовують панель інструментів діаграми для створення, редагування та призначення робочих процесів BPMN, а також налаштування інтеграції з зовнішніми сервісами, такими як каталоги продуктів та системи управління замовленнями. Користувачі Telegram взаємодіють з ботом, щоб переглядати продукти, розміщувати замовлення та надавати деталі доставки.

Ці випадки використання визначають функціональну сферу системи та ілюструють, як кожен актор досягає своїх цілей у додатку.

В таблиці 3.1 наведено детальний опис варіантів використання.

Таблиця 3.1 – Детальний опис варіантів використання

№	Опис
1	Актор: Адміністратор Варіант використання: Переглянути список усіх діаграм BPMN Опис: Адміністратор переходить на сторінку «діаграми», щоб переглянути всі доступні діаграми BPMN. Передумова: Адміністратор повинен увійти в систему, а діаграми BPMN повинні існувати в базі даних. Післяумова: Відображається список усіх діаграм BPMN, і адміністратор може вибрати діаграму для подальших дій.
2	Актор: Адміністратор Варіант використання: Створити нову діаграму BPMN Опис: Адміністратор натискає кнопку «create new diagram», щоб створити новий робочий процес BPMN. Передумова: Адміністратор повинен знаходитись на сторінці списку діаграм. Післяумова: Нова діаграма BPMN створюється та зберігається, готова для редагування.
3	Актор: Адміністратор Варіант використання: Редагувати наявну діаграму BPMN Опис: Адміністратор відкриває наявну діаграму BPMN і змінює робочий процес. Передумова: Діаграма BPMN повинна існувати в системі. Післяумова: Зміни, внесені до діаграми BPMN, зберігаються, оновлюючи робочий процес.

№	Опис
4	Актор: Адміністратор Варіант використання: Збережіть зміни в діаграмі BPMN Опис: Після редагування діаграми BPMN адміністратор натискає кнопку «save». Передумова: Діаграма BPMN має бути відкритою для редагування. Післяумова: Оновлена діаграма BPMN збережена та доступна для подальшого використання чи призначення.
5	Актор: Адміністратор Варіант використання: Призначає діаграму BPMN для виконання Telegram Bot-у Опис: Адміністратор призначає діаграму BPMN боту Telegram, що змінює поведінку бота. Передумова: Адміністратор повинен знаходитись на сторінці редагування діаграми та натиснути кнопку «assign». Післяумова: бот Telegram тепер виконує призначений робочий процес BPMN.
6	Актор: Адміністратор Варіант використання: Налаштування взаємодії із зовнішніми сервісами Опис: Адміністратор за допомогою елементів діаграми налаштовує взаємодію із зовнішніми службами, такими як каталоги продуктів або системи керування замовленнями. Передумова: Зовнішні сервіси повинні надавати кінцеві точки API. Післяумова: Зовнішні сервіси інтегровані, що дозволяє боту за потреби отримувати дані з зовнішніх сервісів, або відправляти дані у зовнішні сервіси.
7	Актор: Користувач Telegram Варіант використання: Переглянути доступні продукти Опис: Користувач надсилає боту запит на перегляд доступних товарів,

№	Опис
	отриманих із інтегрованого каталогу. Передумова: Бот повинен мати доступ до каталогу продуктів із внутрішніх даних або зовнішньої інтеграції. Післяумова: Бот виводить користувачеві список доступних товарів.
8	Актор: Користувач Telegram Варіант використання: Додати продукти для замовлення Опис: Користувач при перегляді продуктів може додати продукт до замовлення. Передумова: Продукт повинен бути відображений користувачеві. Післяумова: Вибраний продукт додаються до поточного замовлення користувача.
9	Актор: Користувач Telegram Варіант використання: Надайти деталі доставки замовлення Опис: Користувач вводить дані доставки (ім’я, адреса, контактна інформація) через бот. Передумова: У користувача повинні бути вибрані товари для замовлення. Післяумова: Деталі доставки зберігаються та пов’язуються із замовленням користувача.
10	Актор: Користувач Telegram Варіант використання: Зробити замовлення через бот Опис: Користувач підтверджує та оформляє замовлення через бот. Передумова: Користувач повинен мати вибрані товари та вказати деталі доставки. Післяумова: Замовлення зберігається в системі.
11	Актор: Адміністратор Варіант використання: Налаштовує інтеграцію каталогу продуктів

№	Опис
	Опис: Адміністратор налаштовує інтеграцію із зовнішнім каталогом продуктів, щоб динамічно отримувати актуальну інформацію про продукт. Передумова: Зовнішній каталог продуктів повинен мати доступний API, а адміністратор повинен мати облікові дані. Післяумова: Каталог товарів інтегрований, а інформація про товар доступна боту.
12	Актор: Адміністратор Варіант використання: Налаштовує інтеграцію сервісу з системою замовлень Опис: Адміністратор налаштовує інтеграцію із зовнішньою системою замовлень для зберігання та обробки замовлень, розміщених ботом. Передумова: Зовнішня система управління замовленнями повинна надавати кінцеві точки API. Післяумова: Замовлення, оформлені через бота, зберігаються і обробляються в інтегрованій системі керування замовленнями.

У додатку А наведено основні варіанти використання інформаційної системи.

Висновки до розділу

Цей розділ визначає основних учасників і випадки використання системи, підкреслюючи окремі ролі Адміністратора і Користувача Telegram в автоматизації процесів доставки їжі. Адміністратор взаємодіє з панеллю інструментів діаграм, керуючи робочими процесами BPMN, налаштовуючи ботів Telegram та інтегруючі зовнішні системи, такі як каталоги продуктів та рішення для управління

замовленнями. З іншого боку, користувачі Telegram взаємодіють з Telegram Bot для перегляду продуктів, надання деталей доставки та розміщення замовлень.

Детальні випадки використання ілюструють функціональну сферу системи, демонструючи, як визначені актори досягають своїх цілей. Дозволяючи адміністраторам ефективно керувати робочими процесами та інтеграціями, дозволяючи користувачам безперешкодно взаємодіяти з ботом, система забезпечує ефективне та масштабоване рішення для автоматизації процесів доставки їжі.

Включення діаграми випадків використання та детальної таблиці (таблиця 3.1) забезпечує всебічний огляд функціональності системи, забезпечуючи чіткість та структуру у визначенні взаємодії та відповідальності користувачів. Ці випадки використання складають основу для реалізації програмною частини системи, узгодження її дизайну та розвитку з потребами користувачів та бізнес-цілями.

4 МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

4.1 Постановка задачі

Процес визначається за допомогою діаграми, яка описує послідовність кроків, яких повинен дотримуватися чат-бот. Для підтримки діяльності служби доставки продуктів харчування діаграмою задано кроки процесу оформлення замовлення, ввід адреси, контактних даних та підтвердження.

Завдання полягає в синтезі скінченного автомата для моделювання та виконання робочого процесу.

4.2 Математична модель задачі

Синтезувати детермінований скінченний автомат $A = \{S, S_0, X, Y, T, G\}$, де:

– $S = \{ S_i \mid i = \overline{1,n} \}$ – множина станів,

де n – кількість станів;

– $X = \{ X_i \mid i = \overline{1,m} \}$ – множина входів,

де m – кількість входів;

– $Y = \{ Y_i \mid i = \overline{1,k} \}$ – множина виходів,

де k – кількість виходів;

– $T : S \times X \to S$ – функція, яка визначає наступний стан на основі поточного стану та входу;

– $G : S \times X \to Y$ – функція, яка визначає вихід системи для заданого стану та входу.

Автомат має задовольняти наступним обмеженням:

– процес повинен правильно обробляти послідовності вхідних подій X;

– у випадку некоректного введення інформації автомат повинен залишатися в поточному стані і вивести відповідне повідомлення-уточнення щодо

формату вводу.

4.3 Синтез скінченного автомату

Відповідно до навчального посібника [1] «Методи моделювання інформаційних систем» синтезуємо скінченний автомат за наступними єтапами:

- задається закон функціонування автомата;
- мінімізується кількість внутрішніх станів автомата;
- кодуються стани автомата;
- визначаються функції збудження елементів пам’яті і функції виходів, також забезпечується їх мінімізація.

Визначимо правила функціонування автомата:

- запустити процес можна за допомогою команди /start;
- ввід адреси від користувача;
- ввід контактної інформації (наприклад, номер телефону);
- підтвердження деталей замовлення;
- завершити процес після успішного підтвердження.

Мінімізуємо кількість станів:

- S_0: початковий стан – автомат чекає команду /start для ініціювання процесу;
- S_1: стан введення адреси – автомат очікує, що користувач надасть коректну адресу;
- S_2: стан введення контакту – автомат очікує дійсну контактну інформацію;
- S_3: стан підтвердження – автомат очікує підтвердження даних замовлення користувачем;
- S_4: остаточний стан – процес успішно завершено.

Некоректні входи в будь-якому стані призводять до повідомлення про помилку і відсутності переходу стану (автомат залишається в тому ж стані).

Кодуємо стани автомату:

- $S_0 = 0$;
- $S_1 = 1$;
- $S_2 = 2$;
- $S_3 = 3$;
- $S_4 = 4$.

Визначимо та мінімізуємо можливі входи X, нехай кожен вхід для спрощення сприйняття буде закодовано текстом:

- /start: початок процесу;
- valid: коректне введення даних;
- invalid: некоректний вхід;
- /cancel: відміна замовлення;
- /confirm: підтвердження замовлення, кінець процесу.

Сформуємо таблицю станів та переходів. Переходи перелічені в таблиці 4.1.

Таблиця 4.1 – Таблиця переходів

Поточний стан (S)	Вхід (X)	Наступний стан (S′)	Вихід (Y)
0	/start	1	Процес розпочато. Введіть свою адресу
1	valid	2	Адреса прийнята. Будь ласка, надайте контакт отримувача
1	invalid	1	Недійсний формат адреси. Повторіть спробу
2	valid	3	Контакт прийнято. Підтвердьте замовлення
2	invalid	2	Недійсна контактна інформація. Повторіть спробу
3	/confirm	4	Замовлення прийнято!
3	/cancel	4	Замовлення скасовано

Переходи скінченного автомату можна представити у вигляді орієнтованого графа, де:

- кожний стан S відповідає вузлу
- ребра між вузлами представляють переходи на основі поточного стану S та входу X у інший стан S з повідомленням Y.

Висновки до розділу

У цьому розділі описано постановку задачі, математичну модель задачі, та синтез скінченного автомату. Формулювання задачі визначило набір станів, переходів, входів і виходів. Описано детальний синтез скінченного автомату.

Описаний підхід висвітлює можливість перетворення діаграм BPMN в математичні моделі, що дозволяє потужний систематичний синтез скінченних автоматів для виконання процесів BPMN. Включення механізмів обробки помилок, таких як перевірка входів користувачів і управління недійсними випадками, забезпечує практичне застосування в реальних сценаріях.

Математична забезпечення служить критичним компонентом у синтезі скінченних автоматів для систем, визначених діаграмами BPMN, долаючи розрив між теоретичними моделями та практичними реалізаціями.

5 АРХІТЕКТУРА СИСТЕМИ

5.1 Модель надання сервісу

При розробці та впровадженні сучасних інформаційних систем вибір відповідної моделі надання послуг є критичним рішенням. Серед найбільш відомих моделей – Інфраструктура як послуга (IaaS), Платформа як послуга (PaaS) та Програмне забезпечення як послуга (SaaS). Кожна модель являє собою різний рівень абстракції в наданні ресурсів і функціональних можливостей для користувачів.

SaaS була обрана як модель доставки послуг з кількох причин.

Рішення SaaS усувають необхідність для бізнесу керувати інфраструктурою або обробляти складні процеси розгортання. Користувачі можуть безпосередньо отримати доступ до програмного забезпечення через веб-браузер або API, що робить його ідеальним для малих бізнесів з обмеженими ресурсами та технічними знаннями.

SaaS усуває авансові витрати, пов'язані з закупівлями обладнання та програмного забезпечення. Це також зменшує поточні витрати, оскільки технічне обслуговування, оновлення та масштабованість управляються постачальником, а не клієнтом.

Платформи SaaS розроблені з урахуванням кінцевих користувачів. Керуючи оновленнями, оптимізацією продуктивності та безпекою централізовано, провайдер забезпечує послідовний та надійний користувальницький досвід.

5.2 Обрана архітектура

Для реалізації інформаційної системи була обрана мікросервісна архітектура для забезпечення масштабованості, розширюваності та ремонтопридатності [2]. Ця архітектура розділяє систему на незалежні сервіси, кожен з яких відповідає за певну функціональність, що полегшує розробку, розгортання та масштабування.

Мікросервісна Архітектура була обрана з кількох причин, узгоджених з нефункціональними вимогами системи:

Масштабованість: сервіси можуть бути масштабовані по горизонталі для обробки підвищеного навантаження без значних архітектурних змін.

Підтримуванність: поділ функціональних можливостей на окремі сервіси зменшує складність, полегшуючи підтримку та оновлення системи.

Гнучкість: архітектура дозволяє легко додавати нові функції або сервіси, такі як інтеграція нових платформ обміну повідомленнями або зовнішніх систем.

Розширюваність: нові сервіси можуть бути розроблені та інтегровані без впливу на існуючі компоненти, підтримуючи майбутнє зростання та адаптацію.

Стійкість: відмова в одному сервісі не обов'язково руйнує всю систему, підвищуючи загальну надійність.

Технологічне різноманіття: кожен сервіс може використовувати найбільш відповідний стек технологій для своїх конкретних потреб, підвищуючи продуктивність та ефективність.

5.3 Компоненти системи

5.3.1 Компонент Diagram Service

Diagram Service: керує BPMN діаграмами, які визначають робочі процеси та поведінку ботів. Цей сервіс дозволяє адміністраторам створювати, редагувати та зберігати діаграми BPMN. Він виконує робочі процеси BPMN за запитом з інших сервісів. Також цей сервіс в разі необхідності має можливість робити інтернет запити до сторонніх сервісів. В цілях демонстрації, тестування та налагодження API викликів сторонніх сервісів в якості стороннього сервіса виступатиме Product Service.

5.3.2 Компонент Telegram Bot Service

Telegram Bot Service: обробляє взаємодію з користувачами через Telegram. Він спілкується з API Telegram для надсилання та отримання повідомлень. За для виконання кроків процесу BPMN він звертається до API Diagram Service.

5.3.3 Компонент Frontend Service

Frontend Service: надає інтерфейс адміністратора для створення та редагування діаграм BPMN, налаштування інтеграції та керування ботами. Він взаємодіє з бекенд сервісами через API, з Diagram Service для створення, збереження, редагування BPMN діаграм, і з Telegram Bot Service для того аби призначити обрану діаграму на виконання.

5.3.4 Компонент Product Service

Product Service: керує каталогом продукції та обробкою замовлень. Деякі підприємства вже можуть мати каталоги продуктів і системи управління замовленнями. Включення цього сервісу до архітектури позитивно впливає на можливості тестування інтеграції з вже існуючими рішеннями і гарантує, що підприємства можуть продовжувати використовувати свої встановлені системи, користуючись новими можливостями автоматизації, наданими додатком. У додатку Б зображено архітектуру системи.

5.3.5 Поділ на сервіси

Модульний характер архітектури мікросервісу означає, що додавання підтримки іншої програми обміну повідомленнями (наприклад, WhatsApp, Facebook Messenger) є простим. Новий бот-сервіс для потрібної платформи може бути розроблений для використання тих же робочих процесів BPMN від сервісу

діаграм. Така конструкція уникає надмірності – повторно використовує існуючі робочі процеси та логіку. Підтримує узгодженість, гарантує, що користувачі на різних платформах мають послідовний досвід роботи.

Розділення Diagram Service і Telegram Bot Service підвищує розширюваність і гнучкість системи. Цей поділ дозволяє:

− незалежно працювати над кожним сервісом окремо, а оновлення або виправлення можуть бути розгорнуті без впливу на інші сервіси;

− масштабути сервіси індивідуально на основі навантаження, наприклад, якщо взаємодія користувачів зростає, а обчислювання кроків BPMN процесу не надто важке за обчислювальними ресурсами, масштабувати потрібно тільки Telegram Bot Service;

− додавати нові платформи обміну повідомленнями шляхом розробки додаткових ботів-сервісів, які взаємодіють з вже існуючим сервісом діаграм, без зміни інших компонентів.

5.4 Інфраструктурні вимоги

Для забезпечення масштабованості, ремонтопридатності і надійності системи був обраний контейнерний підхід розгортання з використанням Docker і Kubernetes. Ця архітектура дозволяє оптимізувати управління послугами, плавне масштабування та підвищену відмовостійкість за допомогою оркестрування. Сервісний реєстр використовується в Kubernetes для підтримки зв'язку між мікросервісами.

У таблиці 5.1 викладено вимоги до обладнання для розгортання системи у виробничому середовищі.

Таблиця 5.1 – Технічні вимоги

Ресурс	Frontend Service	Сервіс діаграм	Сервіс Telegram Bot	Всього
Ядра ЦП	1 ядро	2 ядра	1 ядро	4 ядра

Ресурс	Frontend Service	Сервіс діаграм	Сервіс Telegram Bot	Всього
Оперативна пам'ять	1 ГБ	2 ГБ	1 ГБ	4 ГБ
Дискове сховище (SSD)	10 ГБ	20 ГБ	10 ГБ	40 ГБ
Пропускна здатність мережі	1 Гбіт/с	1 Гбіт/с	1 Гбіт/с	1 Гбіт/с (спільно)
Мінімальна кількість контейнерів	1	1	1	3

Висновки до розділу

У цьому розділі наведено повний огляд архітектури системи, яка базується на модульному мікросервісному підході. Система розділена на кілька незалежних сервісів, кожна з яких відповідає за обробку певного набору функціональних можливостей. До таких сервісів відноситься:

- Frontend Service, який надає інтерфейс користувача для адміністраторів;
- Diagram Service, яка керує робочими процесами BPMN;
- Telegram Bot Service, який обробляє взаємодію користувачів через Telegram;
- Product Service, яка керує каталогом продукції та обробкою замовлень.

Це модульне розділення дозволяє системі працювати ефективно, при цьому кожна служба функціонує автономно, при цьому все ще легко взаємодіючи з іншими компонентами.

6 ВИБІР ТА ОБҐРУНТУВАННЯ ЕЛЕМЕНТІВ ТА ТЕХНОЛОГІЙ

У цьому розділі викладено вибір і обґрунтування технологій, що використовуються в системі. Використовуючи Python і його надійну екосистему, бекенд забезпечує ефективну продуктивність і гнучкість в архітектурі мікросервісу. Фронтенд побудований з використанням TypeScript та React, що забезпечуэ зручний для користувача досвід. Обрані технології були обрані на основі їх здатності відповідати вимогам проекту.

6.1 Backend

6.1.1 Мова програмування Python

Python був обраний як мова програмування бекенд частини застосунку завдяки своїй універсальності, великій підтримці бібліотек та придатності для мікросервісних архітектур. Його лаконічний синтаксис дозволяє швидко розвиватися, скорочуючи час виходу на ринок, зберігаючи читабельність і ремонтопридатність коду. Вбудовані асинхронні можливості програмування Python, що надаються через стандартну бібліотеку asyncio, роблять його відмінним вибором для обробки взаємодій у реальному часі, таких як ті, що необхідні для Telegram-ботів.

Сильна екосистема бібліотек Python, таких як FastAPI [3] для створення API та Aiogram для розробки ботів Telegram, гарантує, що система може ефективно відповідати як функціональним, так і нефункціональним вимогам. Крім того, активна спільнота Python та широке впровадження гарантують доступ до постійної підтримки та оновлень, що робить його перспективним вибором для масштабованих та гнучких серверних систем.

Python було вирішено використовувати в усіх бекенд сервісах, тому що він задовольняє сформульованним вимогам, а така уніфікація дає можливість забезпечити узгодженість і багаторазове використання по всій системі. Загальні шаблони проектування та підходи, такі як ін'єкція залежностей, шаблон сховища

репозиторій (Рис 6.1), можуть бути повторно використані в усіх мікросервісах де потрібно. Це зменшує час розробки, дозволяє уникнути дублювання зусиль та гарантує, що всі сервіси дотримуються однакових принципів та практик проектування, що полегшує підтримку та масштабування системи.

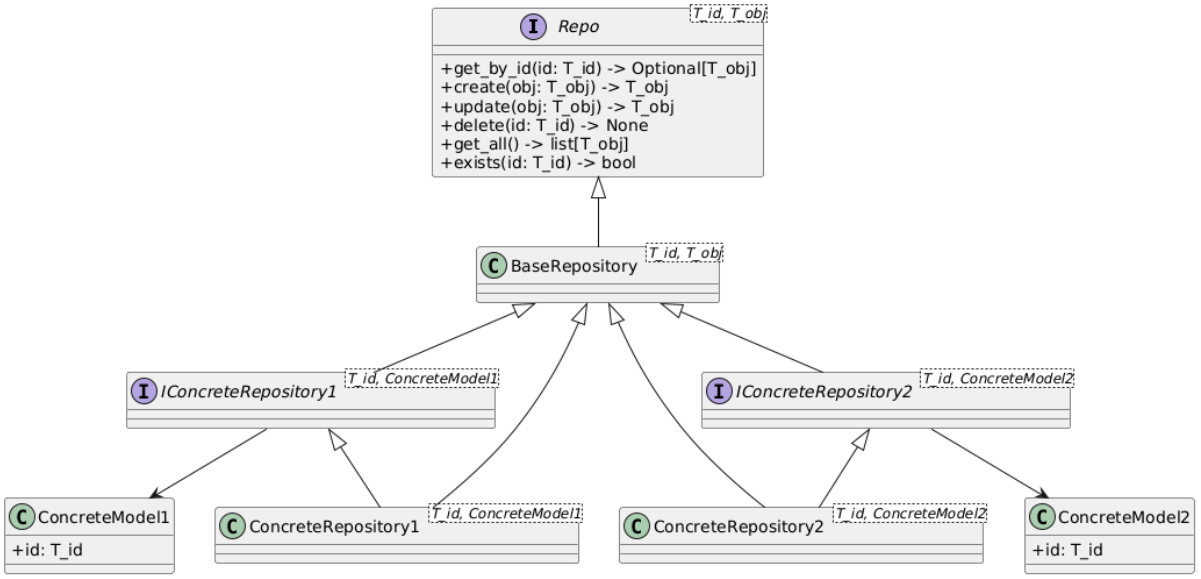

Рисунок 6.1 – Шаблон репозиторій

Далі буде розглянуто ключові фреймворки та бібліотеки, які використовуються при розробці сервера системи. Ці інструменти були обрані на основі їх сумісності з архітектурою мікросервісу, їх продуктивністю та здатністю відповідати вимогам системи.

6.1.2 Веб-фреймворк FastAPI

FastAPI – це сучасний високопродуктивний веб-фреймворк для створення API за допомогою Python. Він розроблений з урахуванням асинхронного програмування, використовуючи можливості asyncio для обробки декількох запитів одночасно.

FastAPI – продуктивний, побудований на Starlette і Pydantic, забезпечує виняткову швидкість і ефективність, у порівнянні з іншими фреймворками.

Він пропонує підказки типу, автоматичну перевірку та створення документації OpenAPI та Swagger, спрощує розробку та забезпечує високу якість коду.

FastAPI використовується в системі для управління кінцевими точками API, обробки HTTP-запитів.

6.1.3 Асинхронний сервер Uvicorn

Uvicorn – це легкий і високопродуктивний сервер ASGI (Asynchronous Server Gateway Interface), призначений для сучасних веб-фреймворків Python, таких як FastAPI. Він повністю підтримує асинхронне програмування, що дозволяє ефективно обробляти кілька одночасних з'єднань.

Відомий своїми мінімальними накладними витратами, Uvicorn забезпечує швидку та надійну роботу сервісів. Він легко інтегрується з FastAPI, що робить його ідеальним вибором для розгортання додатків на основі FastAPI.

У системі Uvicorn використовується для обслуговування додатків FastAPI, забезпечуючи продуктивність і надійність для взаємодії в реальному часі, таких як обробка запитів користувачів через бот Telegram.

6.1.4 Телеграм-бот фреймворк Aiogram

Aiogram – це бібліотека призначена для створення Telegram-ботів [4]. Вона ефективно використовує асинхронне програмування для обробки високих навантажень та взаємодії в реальному часі.

Aiogram забезпечує простий і модульний API, що дозволяє легко створювати і підтримувати функціональність ботів.

Асинхронний характер забезпечує безперебійну роботу декількох користувачів і бесід одночасно.

Бібліотека пропонує вбудовані інструменти для управління командами, зворотними викликами та вбудованими запитами, оптимізуючи процес розробки.

У системі Aiogram використовується для реалізації Telegram Bot Service, що дозволяє взаємодіяти з користувачами та виконувати робочі процеси, визначені на діаграмах BPMN.

6.1.5 Бібліотека валідації даних Pydantic

Pydantic – це бібліотека валідації даних та керування налаштуваннями для Python, яка використовує підказки типів Python [5].

Це забезпечує сувору перевірку вхідних даних, мінімізацію помилок і підвищення цілісності даних.

Pydantic автоматично генерує валідацію для типів моделі, зменшуючи шаблонний код.

Він легко інтегрується з FastAPI, забезпечуючи швидку розробку надійних кінцевих точок API.

У системі Pydantic використовується в бекенд сервісах для перевірки вводу, серіалізації даних та забезпечення узгодженості API викликів.

6.1.6 Бібліотека парсингу cURL Curlparser

Curlparser – це бібліотека утиліт для перетворення команд cURL в код Python, що спрощує інтеграцію сторонніх API.

Він аналізує необроблені команди cURL і генерує сумісний з Python HTTP код запиту, економлячи час під час інтеграції API.

Curlparser усуває необхідність вручну переписувати складні команди cURL, зменшуючи помилки та покращуючи швидкість розробки.

У системі Curlparser використовується для оптимізації процесу інтеграції зовнішніх API, особливо для тестування та налагодження HTTP-запитів у робочих процесах BPMN.

6.1.7 Бібліотека асинхронних веб запитів Httpx

Httpx – сучасний і ефективний асинхронний HTTP-клієнт для Python, призначений для виконання HTTP/1.1 і HTTP/2 запитів. Він пропонує розширені можливості для зовнішнього API зв'язку.

Httpx підтримує синхронний і асинхронний режими, що робить його дуже універсальним для різних випадків використання.

Він забезпечує підключення з'єднання, тайм-аути, забезпечуючи надійну і ефективну комунікацію з зовнішніми службами.

Бібліотека сумісна з asyncio Python, що забезпечує безшовну інтеграцію з іншим асинхронним кодом.

У системі Httpx використовується для виконання внутрішніх викликів API, таких як комунікація між сервісами.

6.1.8 Фреймворк ін'єкцій залежностей Injector

Injector – це легкий фреймворк для ін'єкцій залежностей для Python, призначений для спрощення управління залежностями в додатках. Ін'єкція залежностей – це модель проектування, яка сприяє модульності, тестуваності та чистому коду шляхом роз'єднання компонентів та управління їх залежностями зовні.

Інжектор дозволяє розробнику визначати та налаштовувати залежності централізовано, які потім автоматично надаються там, де це необхідно, усуваючи необхідність ручного створення об’єктів класів.

Використовуючи інжектор можна реалізувати відокремлені та багаторазові компоненти, гарантуючи, що сервіси легше змінювати або замінювати реалізації залежностей, не впливаючи на інші частини системи.

Гнучкий та інтуїтивно зрозумілий API Injector забезпечує безперебійну інтеграцію з мікросервісами, сприяючи послідовному управлінню залежностями.

Ін'єкція залежності спрощує тестування, дозволяючи легко вводити макетні залежності в компоненти під час модульних тестів [6]. Це відокремлення компонентів гарантує, що тести можуть зосередитися на ізольованій функціональності, не покладаючись на зовнішні системи або послуги. Це сприяє швидкому та надійному розвитку системи.

У системі Injector використовується для управління залежностями, такими як сховища (шаблон репозиторій), конфігурації. Це гарантує, що сервіси залишаються модульними, спрощуючи майбутні розширення або інтеграцію з новими компонентами. Цей підхід відповідає принципам мікросервісної архітектури.

6.2 Frontend

Фронтенд побудований з використанням React та TypeScript. Для обробки візуалізації та редагування діаграм BPMN frontend інтегрує bpmn-js, спеціалізовану бібліотеку, призначену для робочих процесів BPMN. В якості інтерфейсу взаємодії користувача з Telegram ботом виступає додаток Telegram, де бот виконує робочі процеси, визначені на діаграмах BPMN. Ці рішення забезпечує безперебійну та ефективну взаємодію як для адміністраторів, так і для кінцевих користувачів.

6.2.1 Мова програмування TypeScript

TypeScript це надмножина мови програмування JavaScript. TypeScript має статичну типізацію допомагає відловлювати помилки під час розробки, зменшуючи проблеми в рантаймі та підвищуючи надійність [7].

TypeScript пропонує чіткі та суворі визначення типів, що полегшує розуміння та підтримку коду на різних етапах розробки. Його інтеграція з сучасними фреймворками, такими як React, забезпечує безперервні робочі процеси розробки, зберігаючи суворе дотримання контрактів даних.

У системі TypeScript використовується в сервісі з фронтендом. Це рішення мінімізує помилки інтеграції між сервісами та забезпечує надійну основу для масштабування програми, зберігаючи чіткість та ремонтопридатність коду.

6.2.2 Фреймворк веб інтерфейсів React

React – це бібліотека JavaScript для побудови користувацьких інтерфейсів, розроблена з архітектурою на основі компонентів, що сприяє повторному використанню та масштабованості [8]. React використовує віртуальний DOM для ефективного рендерингу, забезпечуючи швидке оновлення користувацького інтерфейсу у відповідь на зміни даних. Декларативний характер React спрощує розробку складних інтерфейсів, дозволяючи розробникам зосередитися на тому, як інтерфейс повинен виглядати і поводитись в різних станах.

Екосистема React підтримує інтеграцію додаткових бібліотек для державного управління, маршрутизації та оптимізації продуктивності, що робить її дуже гнучкою для широкого кола додатків. Його потужна спільнота та активний розвиток забезпечують доступ до найкращих практик та постійного оновлення.

У системі React використовується для побудови інтерфейсу адміністратора. Структура на основі компонентів React дозволяє розробляти модульні та підтримувані функції, такі як сторінка списку діаграм та редактор діаграм, забезпечуючи безперешкодний та інтуїтивно зрозумілий користувальницький досвід.

6.2.3 Бібліотека редагування діаграм бізнес процесів BPMN-js

BPMN-js – потужна бібліотека JavaScript, призначена для візуалізації та редагування діаграм BPMN (Business Process Model and Notation) безпосередньо в браузері [9]. Він надає нестандартні інструменти для візуалізації складних робочих процесів і дозволяє проводити інтерактивні модифікації, що робить його

ідеальним вибором для додатків, які вимагають автоматизації процесів або управління робочим процесом.

Відповідно до роботи [10] «Найбільш перспективним формалізованим описом, на основі якого можна автоматично створювати боти, є описи бізнес-процесів».

Бібліотека підтримує такі функції, як редагування перетягування, налаштування елементів та перевірка стандартів BPMN у реальному часі. Крім того, він розширюваний, що дозволяє розробникам інтегрувати користувацьку поведінку, стилі та компоненти з урахуванням конкретних потреб додатків.

У системі bpmn-js використовується для управління діаграмами BPMN в інтерфейсі адміністратора. Це дозволяє адміністраторам створювати, редагувати та зберігати робочі процеси, які визначають поведінку ботів. Його інтеграція з React забезпечує динамічну та чуйну взаємодію діаграм, забезпечуючи інтуїтивно зрозумілий досвід проектування та управління робочими процесами.

У системі bpmn-js був налаштований для поліпшення користувацького досвіду редагування діаграм. Налаштування включають зміни “палітри”, контекстної панелі та меню заміни, що робить інтерфейс простим, зрозумілим, та ефективним для дизайну робочого процесу.

Меню заміни: адаптовано, щоб дозволити заміну елементів системними параметрами, забезпечуючи сумісність з попередньо визначеними поведінками ботів та інтеграцією.

Контекстна панель: покращена за допомогою спеціальних дій та параметрів, які з'являються під час взаємодії з елементами діаграми, пропонуючи контекстні інструменти для редагування робочих процесів.

Палітра: модифіковано для включення спеціальних інструментів та елементів, що відповідають конкретним потребам системи, оптимізуючи процес створення діаграми.

6.3 Бази даних

Вибір баз даних відіграє важливу роль в архітектурі системи, забезпечуючи цілісність даних, масштабованість і модульність. Для вирішення різноманітних вимог, PostgreSQL використовується для структурованих даних зі складними зв'язками, в той час як MongoDB використовується для неструктурованих динамічних даних.

Кожен бекенд сервіс повинен мати власну базу даних, забезпечуючи слабке зчеплення (low coupling) між сервісами та забезпечувати незалежне масштабування та розвиток.

6.3.1 Система управління базами даних PostgreSQL

PostgreSQL був обраний для сервісів, які вимагають управління структурованими даними, такими як Diagram Service і Products Service. Його потужні розширені реляційні можливості, відповідність ACID та роблять його ідеальним вибором для транзакційних операцій [11]. Розширюваність і надійна продуктивність PostgreSQL забезпечують надійність обробки запитів і цілісність даних.

У системі сервер баз даних PostgreSQL дозволяє розміщувати кілька специфічних для обслуговування баз даних на одному сервері, знижуючи операційні витрати при збереженні модульності та ізоляції даних. Його масштабованість та активна підтримка спільноти ще більше роблять його перспективним рішенням, що відповідає вимогам системи щодо продуктивності, надійності та ефективності витрат.

6.3.2 Система управління базами даних MongoDB

MongoDB був обраний для сервісів, що вимагають гнучкого зберігання даних без схеми, таких як в сервісі Telegram Bot Service. Його потужна документо

орієнтована модель та підтримка неструктурованих даних роблять його добре придатним для зберігання динамічної інформації [12], такої як стан сеансу користувача та дані взаємодії в реальному часі.

Висока масштабованість MongoDB і розподілена архітектура дозволяють їй обробляти великі обсяги одночасних запитів. Його асинхронні операції та можливість масштабування по горизонталі ідеально відповідають потребі Telegram Bot Service в обробці та масштабуванні в реальному часі. Використовуючи MongoDB, система забезпечує ефективність і надійність в управлінні динамічними даними, підтримуючи майбутнє зростання.

Висновки до розділу

У цьому розділі був зроблений обгрунтований вибір технологій та інструментів для реалізації інформаційної системи.

Були обрані мови програмування, бібліотеки та фреймворки для реалізації сервісів.

Цей комплексний вибір технологій забезпечує узгодження з вимогами системи, такими як масштабованість, ремонтопридатність та продуктивність, підтримуючи модульну архітектуру..

7 ОПИС СХЕМИ БАЗИ ДАНИХ

Застосунок використовує комбінацію баз даних SQL та NoSQL для ефективного зберігання та управління різними типами даних. PostgreSQL використовується для структурованих даних, зокрема діаграм, тоді як MongoDB обробляє неструктуровані та динамічні дані, такі як дані взаємодії користувачів, введені або розраховані під час взаємодії з Telegram ботом.

7.1 Компонент Diagram Service

Сервіс діаграм керує зберіганням та обробкою діаграм BPMN, які визначають робочі процеси для Telegram-ботів. Сервіс використовує базу даних PostgreSQL для зберігання своїх даних, забезпечуючи надійність і структуроване управління інформацією, пов'язаною з діаграмою.

7.1.1 Сутності

Діаграма: Представляє діаграму BPMN, створену адміністратором. Кожна діаграма містить метадані, такі як її унікальний ідентифікатор та необов'язкове описове ім'я, для полегшення організації та пошуку. Основний вміст діаграми зберігається як XML-представлення, що відповідає стандарту BPMN (Business Process Model and Notation), який визначає структуру, логіку та взаємодію робочого процесу.

7.1.2 Таблиці

У таблиці 7.1 наведено таблиці бази даних сервісу діаграм.

Ця структура даних гарантує, що діаграми BPMN зберігаються і можуть бути отримані або змінені відповідно до вимог системи.

Таблиця 7.1 – Таблиці Diagram Service

Назва таблиці	Поле	Тип даних	Обмеження	Опис
diagram	id	UUID	Primary Key	Унікальний ідентифікатор для діаграми BPMN.
	name	VARCHAR	Nullable	Ім'я діаграми можна залишити порожнім.
	xml_content	TEXT	Not Null	XML-представлення робочого процесу діаграми BPMN.

7.2 Компонент Telegram Bot Service

Сервіс Telegram Bot керує взаємодіями з користувачами Telegram, виконуючи робочі процеси, визначені діаграмами BPMN, що зберігаються в сервісі діаграм. Цей сервіс використовує MongoDB для зберігання динамічних даних і даних без схеми, забезпечуючи високу продуктивність і гнучкість для обробки взаємодії користувачів в режимі реального часу.

7.2.1 Компонент Сутності

Дані сеансу: відстежує специфічні для користувача стани взаємодії та дані під час робочих процесів ботів, забезпечуючи бездоганний та персоналізований користувальницький досвід.

7.2.2 Колекції

У таблиці 7.2 наведено колекції бази даних Telegram Bot Service.

Ця структура MongoDB гарантує, що Telegram Bot Service може ефективно обробляти взаємодію користувачів у режимі реального часу та великих обсягів, забезпечуючи гнучку основу для розвитку робочих процесів та потреб інтеграції.

Таблиця 7.2 – Колекції Telegram Bot Service

Назва колекції	Поле	Тип даних	опис
aiogram_state	chat_id	Integer	Унікальний ідентифікатор чату Telegram (користувача або групи).
	user_id	Integer	Унікальний ідентифікатор користувача Telegram.
	state	String	Поточний стан взаємодії користувача з ботом.
aiogram_data	chat_id	Integer	Унікальний ідентифікатор чату Telegram (користувача або групи).
	user_id	Integer	Унікальний ідентифікатор користувача Telegram.
	data	JSON	Серіалізовані дані користувача для поточної взаємодії.
aiogram_bucket	chat_id	Integer	Унікальний ідентифікатор чату Telegram (користувача або групи).
	user_id	Integer	Унікальний ідентифікатор користувача Telegram.
	bucket	JSON	Технічні дані aiogram такі як прогрес рейт лімітів користувача

7.3 Компонент Products Service

Сервіс продуктів відповідає за управління інформацією, пов'язаною з продуктами, їх категоріями та замовленнями. База даних для цього сервісу реалізована за допомогою PostgreSQL для зберігання структурованих даних з чітко визначеними зв'язками.

7.3.1 Сутності

Продукт: кожен продукт має назву, вартість та детальний опис.

Категорія: представляє категорії, до яких належать продукти, що дозволяють логічне групування та фільтрування. Категорії можуть застосовуватися до декількох продуктів.

Product Category: таблиця, яка встановлює відносини між продуктами та категоріями.

Замовлення: являє кожне замовлення пов'язане з конкретним користувачем і записує час створення.

OrderItem: представляє індивідуальний продукт в замовленні, зберігаючи кількість і вартість продукту.

7.3.2 Таблиці

У таблиці 7.3 наведено таблиці бази даних Products Service.

Таблиця 7.3 – Таблиці Products Service

Назва таблиці	Назва стовпця	Тип даних	обмеження	Опис
Products	id	Integer	Primary Key	Унікальний ідентифікатор товару.
	title	TEXT	Not Null	Назва продукту.

Назва таблиці	Назва стовпця	Тип даних	обмеження	Опис
	cost	Integer	Not Null	Ціна товару в найменшій грошовій одиниці.
	description	TEXT	Nullable	Детальна інформація про товар.
Categories	id	Integer	Primary Key	Унікальний ідентифікатор для категорії.
	name	TEXT	Unique, Not Null	Назва категорії.
Product Category	id	Integer	Primary Key	Унікальний ідентифікатор зв'язку.
	product_id	Integer	Foreign Key	Посилається на поле id у таблиці Products.
	category_id	Integer	Foreign Key	Посилається на поле ідентифікатора в таблиці категорій.
Orders	id	Integer	Primary Key	Унікальний ідентифікатор для замовлення.
	user_id	VARCHAR	Not Null	Ідентифікатор користувача, який розміщує замовлення.
	created_at	DATETIME	Not Null	Мітка часу створення замовлення.
OrderItems	id	Integer	Primary Key	Унікальний ідентифікатор позиції замовлення.
	product_id	Integer	Foreign Key	Посилається на поле id у таблиці Products.
	order_id	Integer	Foreign	Посилається на поле ідентифікатора в

Назва таблиці	Назва стовпця	Тип даних	обмеження	Опис
			Key	таблиці замовлень.
	cost	Integer	Not Null	Вартість товару на момент замовлення.
	count	Integer	Not Null	Кількість товару в замовленні.

Структура бази даних для сервісу продуктів призначена для підтримки ефективного зберігання та пошуку інформації про продукти, категорії та замовлення. Нормалізуючі дані через об'єднання таблиць та зовнішніх ключів, структура забезпечує уникнення дублювання даних та підтримує узгодженість даних. Використання PostgreSQL відповідає потребі сервісу в управлінні реляційними даними та масштабованості. У додатку Е наведено ER діаграму.

Висновки до розділу

У розділі детально описано схему бази даних для системи, що охоплює службу діаграм, Telegram Bot Service та Products Service, кожна з яких має свою власну спеціалізовану базу даних. Структура підтримує модульність та власність даних, поєднуючись з архітектурою мікросервісу.

Основні сутності, такі як діаграми BPMN, дані сеансу Telegram-бота, продукти та замовлення складають основу схеми бази даних. Кожна сутність відіграє вирішальну роль у взаємодії та функціональності системи, відображаючи основні об'єкти та процеси.

Структура бази даних, представлена в таблицях і діаграмах ER, забезпечує ефективне зберігання, пошук і організацію даних. Поділ служб та їх відповідних баз даних підтримує масштабованість та ремонтопридатність, зберігаючи цілісність та безпеку даних у всій системі.

8 РЕАЛІЗАЦІЯ БІЗНЕС-ЛОГІКИ СИСТЕМИ

Цей розділ описує реалізацію бізнес-логіки всередині сервісів системи, зосереджуючись на архітектурі мікросервісу та багаторазових компонентах, що забезпечують масштабованість, модульність, повторне використання коду. У додатку В діаграму пакетів системи.

8.1 Модуль Bases

Модуль Bases – це спільна бібліотека, яка використовується в мікросервісах для стандартизації та спрощення реалізації загальних функцій. Централізуючи ці основні компоненти, модуль потужно зменшує дублювання коду, підвищує узгодженість і сприяє підтримуваності всієї системи.

8.1.1 Логування

Ведення логування є невід'ємною частиною будь-якої програми, а модуль bases забезпечує централізовану та уніфіковану конфігурацію ведення журналу, що використовується всіма сервісами. Ці утиліти стандартизують, як створюються, форматуються логи, забезпечуючи послідовну практику ведення логування в мікросервісах. Модуль підтримує структуроване логування, полегшуючи інтеграцію із зовнішніми системами зберігання, аналізу та моніторингу логів.

Крім того, даний модуль дозволяє розробникам вказувати рівні логування (наприклад, налагодження, інформацію, попередження, помилку), гарантуючи, що відповідний рівень деталізації фіксується на різних етапах розробки та виробництва.

Використовуючи базові утиліти логування, розробники можуть зосередитися на бізнес-логіці без необхідності вручну налаштовувати журналювання в кожному сервісі.

На рисунку 8.2 зображено код модулю логування.

```python
import logging
import sys


def configure_logging():
    logger = logging.getLogger(__name__)
    logger.setLevel(logging.DEBUG)

    handler = logging.StreamHandler(sys.stdout)
    handler.setLevel(logging.DEBUG)
    formatter = logging.Formatter('%(levelname)-9s %(asctime)s - %(name)s - %(message)s')
    handler.setFormatter(formatter)
    logger.addHandler(handler)

```

Рисунок 8.2 – Код модулю логування

8.1.2 Ін'єкція залежностей

Модуль bases включає в себе інструменти для управління ін'єкціями залежностей, спрощення підстановки екземплярів класів в мікросервісах. Ці утиліти зменшують шаблонний код і покращують модульність, автоматично вирішуючи і вводячи необхідні залежності в класи або функції. Утиліти для ін'єкцій залежностей сумісні як з синхронними, так і асинхронними додатками, забезпечуючи потужну інтеграцію з фреймворками, такими як FastAPI. Використовуючи ці інструменти, розробники можуть підтримувати чисту та відокремлену архітектуру, що полегшує модифікацію або заміну компонентів, не впливаючи на інші частини системи.

Базовий модуль відіграє важливу роль у забезпеченні того, щоб всі мікросервіси слідували послідовним шаблонам проектування та стандартам кодування, сприяючи загальній стабільності, масштабованості та ремонтопридатності системи. Його модульна конструкція дозволяє розробляти та інтегрувати нові сервіси, зберігаючи існуючу інфраструктуру.

На рисунку 8.3 зображено приклад утіліти ін’єкції залежностей яка дозволяє бібліотеці Injector працювати разом з фреймворком FastAPI.

```python
from typing import Type, TypeVar

from fastapi import Depends
from injector import Injector, ProviderOf

injector = Injector()
T = TypeVar('T')


def di(type: Type[T]) -> T:
    return Depends(injector.get(ProviderOf[type]).get)

```

Рисунок 8.3 – Приклад утиліти ін’єкції залежностей

8.1.3 Шаблон репозиторій

Модуль bases включає в себе загальну реалізацію шаблону репозиторію, забезпечуючи багаторазовий і розширюваний спосіб управління сутностями бази даних. Ця реалізація абстрагує загальні операції з базою даних, такі як створення, читання, оновлення, видалення та перевірка наявності записів. Патерн репозиторій відокремлює бізнес-логіку від доступу до бази даних, це спрощує тестування та підтримку.

Цей дизайн абстрагує загальні операції з базою даних, такі як створення, читання, оновлення, видалення (CRUD) та перевірки існування, що дозволяє розробникам більш інтуїтивно та ефективно взаємодіяти з базою даних. Патерн репозиторію служить мостом між бізнес-логікою програми та базою даних, забезпечуючи чисте розділення проблем.

Ця абстракція дозволяє розробникам перемикати постачальників баз даних або змінювати основні механізми зберігання, не впливаючи на бізнес-логіку. Наприклад, перехід від бази даних на основі SQL до бази даних NoSQL може бути

досягнутий з мінімальними змінами, оскільки інтерфейс репозиторію залишається послідовним. На додаток до операцій CRUD, шаблон репозиторію в цьому модулі розроблений з урахуванням розширюваності. Розробники можуть легко додавати власні методи, адаптовані до конкретних потреб обробки даних, зберігаючи при цьому дотримання архітектурних принципів системи.

На рисунку 8.1 зображено інтерфейс репозиторію. На рисунку 6.1 зображено шаблон репозиторій.

```python
T_id = TypeVar("T_id")
T_obj = TypeVar("T_obj")


class Repo(Generic[T_id, T_obj], ABC):
    @abstractmethod
    async def get_by_id(self, id: T_id) -> Optional[T_obj]:
        pass

    @abstractmethod
    async def create(self, obj: T_obj) -> T_obj:
        pass

    @abstractmethod
    async def update(self, obj: T_obj) -> T_obj:
        pass

    @abstractmethod
    async def delete(self, id: T_id) -> None:
        pass

    @abstractmethod
    async def get_all(self) -> list[T_obj]:
        pass

    @abstractmethod
    async def exists(self, id: T_id) -> bool:
        pass
```

Рисунок 8.1 – Інтерфейс репозиторій

8.2 Компонент Diagram Service

8.2.1 Структура

Сервіс діаграм є основним компонентом системи, відповідальним за управління діаграмами BPMN та забезпечення виконання робочого процесу. Сервіс розроблений відповідно до моделі Model-View-Controller (MVC), забезпечуючи чітке розділення шарів та сприяючи ремонтопридатності та масштабованості. Його потужна структура включає в себе різні шари, варто відмітити Service Layer і Data Access Layer, кожен з яких виконує певну роль в сервісі.

WebAPI обробляє HTTP-запити та відповіді, розкриваючи кінцеві точки для управління діаграмами. Цей рівень не містить ніякої бізнес-логіки, він містить структуру кінцевих точок, використовує FastAPI, а саму обробку даних делегує Service Layer.

Service Layer виступає посередником між бізнес логікою, шаром доступу до даних та WebAPI. Цей шар впроваджений для подальшої гнучкості і можливості використовувати код сервісу діаграм з іншими фреймворками WebAPI.

Data Access Layer відповідає за прямі взаємодії з базою даних, забезпечуючи збереження та отримання даних діаграми.

Business Logic Layer включає в себе моделі бізнес сутностей та код бізнес логіки – двигун машини станів для виконання кроків BPMN діаграм, парсер, та інші. У Додатку Г наведено діаграму класів сервісу.

Цей структурований підхід гарантує, що кожен компонент має чітку відповідальність, підвищуючи повторне використання коду та спрощуючи тестування та обслуговування.

На рисунку 8.4 зображено структуру файлів сервісу Diagram Service.

В папці examples приведено декілька прикладів простих BMPN діаграм:

- приклад використання математичних операцій;
- приклад виклику стороннього API;
- приклад “відлуння”.

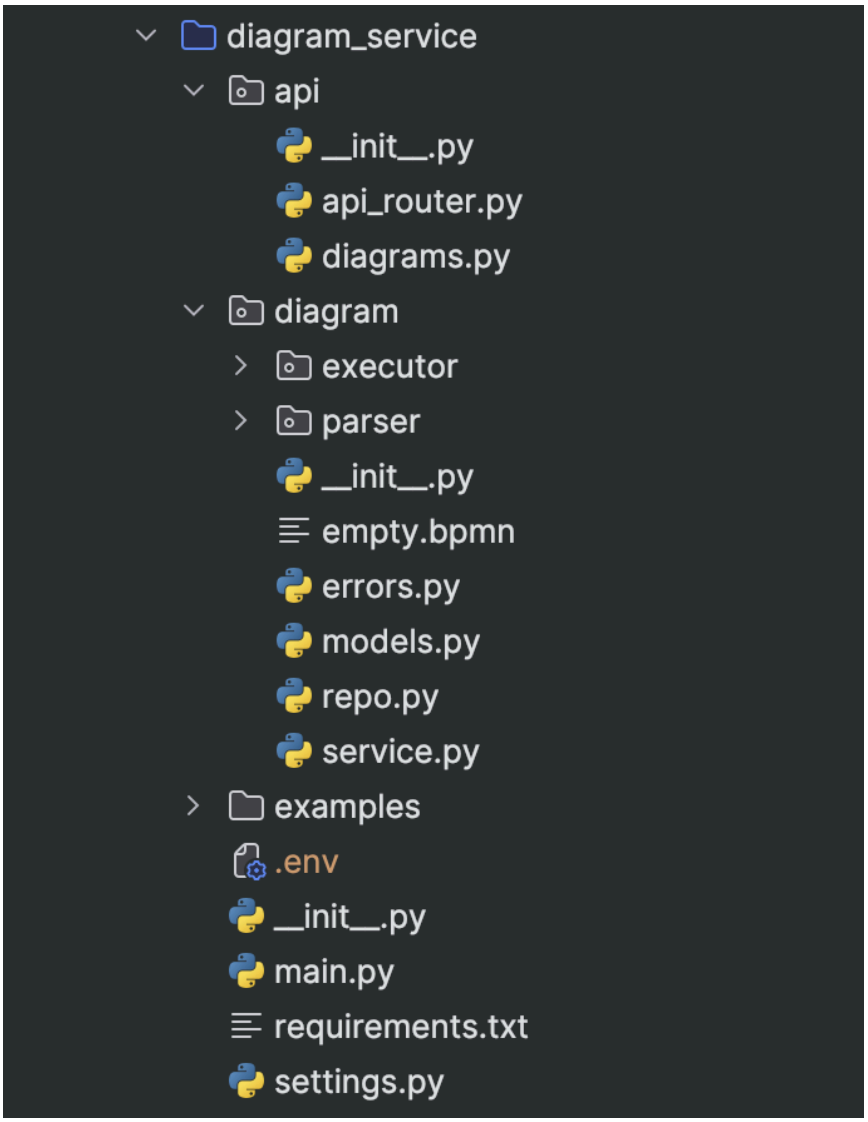

Рисунок 8.4 – Структура Diagram Service

8.2.2 Парсер

Парсер діаграм – це спеціальний компонент, який перевіряє та інтерпретує діаграми BPMN, надані адміністратором. Це гарантує, що діаграми відповідають стандартам BPMN і підходять для виконання. Парсер обробляє XML-представлення діаграм, витягуючи критичну інформацію, таку як завдання, події та шлюзи. Ці витягнуті дані використовуються для побудови моделі робочого процесу в пам'яті, яка може бути виконана двигуном машини станів.

На рисунку 8.5 наведено приклад моделей валідації.

```python
import xmltodict
from pydantic import BaseModel, Field


class Model(BaseModel):
    class Config:
        populate_by_name = True


class MessageEventDefinition(Model):
    id: str = Field(alias='@id')


class Event(Model):
    id: str = Field(alias='@id')
    name: str | None = Field( default: None, alias='@name')
    incoming: str | list[str] | None = Field( default: None, alias='bpmn:incoming')
    outgoing: str | list[str] | None = Field( default: None, alias='bpmn:outgoing')
    message_event_definition: MessageEventDefinition | None = Field( default: None, alias='bpmn:messageEventDefinition')
    type: str


class SequenceFlowItem(Model):
    id: str = Field(alias='@id')
    source_ref: str = Field(alias='@sourceRef')
    target_ref: str = Field(alias='@targetRef')
    name: str | None = Field( default: None, alias='@name')


class Process(Model):
    id: str = Field(alias='@id')
    is_executable: str = Field(alias='@isExecutable')
    events: list[Event] = Field()
    sequence_flow: list[SequenceFlowItem] = Field(alias='bpmn:sequenceFlow')

```

Рисунок 8.5 – Моделі валідації BPMN

8.2.3 Двигун машини станів

Двигун машини станів – це модуль, який відповідає за виконання робочих процесів, визначених на діаграмах BPMN. Він інтерпретує модель в пам'яті, і організовує завдання, події та точки прийняття рішень на основі логіки робочого процесу. Двигун підтримує інтеграцію з зовнішніми сервісами за допомогою викликів API, забезпечуючи динамічні взаємодії в рамках виконання робочого процесу. Наприклад, він може викликати такі дії, як запит на інформацію про продукт.

На рисунку 8.6 зображено інтерфейс двигуна BPMN.

Рисунок 8.6 – Інтерфейс класу BpmnExecutor

Повну реалізацію класу BpmnExecutor наведено у Додатку К.

8.2.4 Web API

Веб-API є точкою входу для взаємодії з службою діаграм, виставляючи кінцеві точки для адміністраторів для управління діаграмами BPMN. Він забезпечує функціональність для:

- створення нових діаграм шляхом прийняття вмісту BPMN XML;
- отримання збережених діаграм для перегляду або редагування;
- оновлення існуючих діаграм за допомогою нової або модифікованої логіки BPMN;
- видалення діаграм.

Web API реалізований за допомогою FastAPI, використовуючи свої асинхронні можливості для високої продуктивності. Всі вхідні запити направляються через рівень API і делегуються на рівень обслуговування, гарантуючи, що жодна бізнес-логіка не знаходиться в кінцевих точках. Цей підхід

спрощує реалізацію API і гарантує, що зміни в бізнес-логіці не впливають на структуру API.

На рисунку 8.7 зображено декілька кінцевих точок API. Тут можна побачити взаємодію між Injector і утилітою di, FastAPI, та ServiceLayer.

```
/diagrams/
@router.post( path: "/", response_model=Diagram)
async def create_diagram(
    create_diagram_req: CreateDiagram,
    ds: DiagramService = di(DiagramService)
):
    return await ds.create(create_diagram_req)


/diagrams/{diagram_id}
@router.put( path: "/{diagram_id}", response_model=Diagram)
async def update_diagram(
    diagram_id: uuid.UUID,
    diagram_update: UpdateDiagram,
    ds: DiagramService = di(DiagramService)
):
    return await ds.update(diagram_id, diagram_update)


/diagrams/{diagram_id}/run
@router.post("/{diagram_id}/run")
async def step_diagram(
    diagram_id: uuid.UUID,
    payload: RunDiagramPayload,
    ds: DiagramService = di(DiagramService),
):
    actions, new_state = await ds.run_diagram(diagram_id, message=payload.message, state=payload.state)
    return RunDiagramResultDTO(actions=actions, new_state=new_state)
```

Рисунок 8.7 – Кінцеві точки API

8.3 Модуль Diagram Service API Client

Клієнт API служби діаграм – це багаторазовий модуль, призначений для полегшення безперебійного зв'язку між іншими мікросервісами та сервісом діаграм. Він абстрагує складності взаємодії з веб-API служби діаграм, забезпечуючи простий і надійний інтерфейс для створення запитів і обробки відповідей. Цей клієнт відіграє вирішальну роль у наданні послуг, таких як Telegram Bot Service, для доступу та виконання робочих процесів, визначених на діаграмах BPMN.

Клієнт API спрощує взаємодію зі службою діаграм:

– надання методів високого рівня для загальних операцій, таких як виконання діаграми, отримання діаграми або перевірка її структури, що дозволяє іншим службам зосередитися на своїй основній функціональності;

– інкапсулювання всіх HTTP запитів і відповідей в централізованому модулі;

– обробка помилок та повторні спроби для викликів API, забезпечуючи надійність та надійність;

– централізована валідація запитів.

Крім того, клієнт API є розширюваним і може бути повторно використаний в майбутніх мікросервісах, таких що будуть підтримувати інші платформи обміну повідомленнями, демонструючи його модульний і масштабований дизайн.

На рисунку 8.8 зображено структуру DiagramApiClient – методи та сигнатури цих методів.

```
class DiagramApiClient:
    def __init__(self, base_url: str):...

    async def get_all(self) -> List[Diagram]:...

    async def get_by_id(self, diagram_id: uuid.UUID) -> Diagram:...

    async def get_by_name(self, name: str) -> Diagram:...

    async def create(self, request: CreateDiagram) -> Diagram:...

    async def update(self, diagram_id: uuid.UUID, request: UpdateDiagram) -> Diagram:...

    async def run_diagram(self, diagram_id: uuid.UUID, message: str, state: dict) -> RunDiagramResult:...
```

Рисунок 8.8 – структура DiagramApiClient

Клієнт API служби діаграм є важливим компонентом архітектури системи, що забезпечує плавний і послідовний зв'язок між мікросервісами. Інкапсулюючи складність взаємодій API, він підвищує модульність, масштабованість та підтримуванність інформаційної системи. Цей підхід гарантує, що нові функції або зміни в службі діаграм можуть бути легко інтегровані без порушення залежних служб.

8.4 Компонент Telegram Bot Service

За структурою Telegram Bot Service сервіс схож на Diagram Service. Він містить шари бізнес логіки, доступу до даних, сервісний, WebAPI.

Telegram Bot Service служить інтерфейсом між кінцевими користувачами і системою, що дозволяє взаємодіяти через Telegram. Ця служба не виконує робочі процеси, визначені безпосередньо діаграмами BPMN, замість цього він делегує виконання кроків робочого процесу службі діаграм. Telegram Bot Service зосереджується на управлінні взаємодіями користувачів, підтримці станів сеансу. У Додатку Д наведено діаграму класів сервісу. У додатку Ж зображено діаграму послідовності обробки повідомлення користувача

8.4.1 Модуль TgBot

Підмодуль TgBot є компонентом Telegram Bot Service, що відповідає за обробку взаємодій користувачів відповідно до кроків, визначених на діаграмах BPMN. Цей підмодуль реалізовано за допомогою бібліотеки Aiogram, легкого та асинхронного фреймворку Python для побудови Telegram-ботів. Aiogram забезпечує ефективну взаємодію в режимі реального часу, використовуючи asyncio Python для високої паралельності та продуктивності.

Цей компонент виступає посередником між Telegram Bot і Diagram Service, полегшуючи виконання окремих кроків робочого процесу.

Він використовує клієнт API служби діаграм для виконання кроків діаграм BPMN.

Модуль використовує MongoDB для відстеження стану сеансу користувача та інших динамічних даних, що є важливим для забезпечення персоналізованого та безперебійного досвіду користувача [13].

На рисунку 8.9 зображено код Telegram Bot Service для взаємодії з користувачем та Diagram Service.

```python
@dp.message()
async def process_message(message: types.Message, state: FSMContext):
    await bot.send_chat_action(message.chat.id, action: 'typing')
    data = await state.get_data()
    bpmn_state = State(**data['bpmn_state']) if data else State()
    bpmn_state.data['user_id'] = message.from_user.id

    try:
        res = await ds.run_diagram(
            tg_bot.diagram_id,
            message.text,
            bpmn_state.model_dump()
        )
    except ValueError as e:
        traceback.print_exc()
        await message.answer('internal error')
        return
    except ValidationError as e:
        await message.answer(e.message)
        return
    except NoResponseError as e:
        return

    logger.info(f'Actions: {res.actions}')

    for action in res.actions:
        logger.info(f'Action: {action}')
        match action:
            case SendMessage(text):
                await message.answer(text)
            case WaitMessage():
                await message.answer('Waiting for your message')
            case _:
                pass

    logger.info(f'New state: {res.new_state}')
    await state.set_data({'bpmn_state': res.new_state})
    logger.info(f'Processed message: {message.text}')
```

Рисунок 8.9 – структура DiagramApiClient

8.4.2 Web API

Telegram Bot Web API є важливою частиною Telegram Bot Service, надаючи кінцеві точки адміністраторам для налаштування та управління Telegram-ботами. Подібно до веб-API служби діаграм, він слідує чистому архітектурному дизайну, забезпечуючи чітке розділення шарів.

Web API був реалізований за допомогою FastAPI, слідуючи тим же дизайнерським рішенням, що і Diagram Service Web API. Бізнес-логіка

знаходиться в рівні обслуговування, причому кінцеві точки служать виключно точками входу для запитів. Цей підхід узгоджується з кращими практиками, сприяючи модульності, масштабованості та простоті тестування.

На рисунку 8.10 зображено приклад кінцевих точок WebAPI.

```
/bot/tg/assign_default/{diagram_id}
@router.put("/assign_default/{diagram_id}")
async def assign_default_bot(
    diagram_id: UUID,
    bots=di(TgBotService)
):
    bot = await bots.get_default()
    bot.diagram_id = diagram_id
    await bots.update(bot.id, UpdateBot(diagram_id=diagram_id))


/bot/tg/{bot_id}/restart
@router.post("/{bot_id}/restart")
async def restart_bot(
    bot_id: UUID,
    bots=di(TgBotService)
):
    await bots.restart_bot(bot_id)
    return {"status": "ok"}
```

Рисунок 8.10 – Приклад кінцевих точок WebAPI

8.5 Компонент Products Service

Products Service – це сервіс, розроблений в першу чергу для цілей тестування, що демонструє здатність системи інтегруватися з вже існуючими рішеннями для управління каталогами продуктів і замовленнями. Його дизайн підкреслює гнучкість архітектури системи для адаптації до зовнішніх сервісів, демонструючи варіанти безшовної інтеграції з іншими системами управління продуктами та замовленнями.

Сервіс надає API для взаємодії з іншими компонентами, зокрема службою діаграм, що дозволяє робочим процесам динамічно отримувати дані про продукт

або створювати замовлення на основі взаємодії користувачів у боті Telegram. Це демонструє реальну можливість інтеграції зі сторонніми рішеннями.

Сервіс побудований за допомогою Python і FastAPI, дотримуючись тих же архітектурних принципів, що і інші мікросервіси в системі. Він включає в себе патерн репозиторію для доступу до даних, відокремлюючи взаємодію баз даних від бізнес-логіки для забезпечення ремонтопридатності. PostgreSQL використовується як база даних, забезпечуючи надійні реляційні можливості для обробки структурованих даних.

Впроваджуючи цю сервіс, система демонструє свою розширюваність і готовність до реальних сценаріїв інтеграції, таких як підключення до встановлених каталогів продуктів або систем обробки замовлень. Це гарантує, що архітектура може вмістити різні випадки використання за межами початкової реалізації.

На рисунку 8.11 зображено приклад структур запитів та відповідей.

```python
class ProductResponse(BM):
    id: int
    title: str
    cost: int
    description: str


class ProductsResponse(BM):
    products: List[ProductResponse]
    has_next: bool
    has_prev: bool


class OrderItemRequest(BM):
    product_id: int


class OrderRequest(BM):
    user_id: str
    items: List[OrderItemRequest]


class OrderResponse(BM):
    id: int
    user_id: str
    created_at: datetime.datetime

```

Рисунок 8.11 – Приклад структур запитів

Висновки до розділу

Розділ детально описує розробку основних сервісів системи, включаючи сервіс діаграм, сервіс Telegram Bot та сервіс продуктів. Кожен сервіс слідує модульній архітектурі з чітко визначеними шарами, використовуючи спільні компоненти, такі як модуль bases для узгодженості. Служби легко взаємодіють, а Telegram Bot Service делегує виконання робочого процесу службі діаграм та службі продуктів, виділяючи можливості інтеграції. Цей підхід забезпечує масштабованість, ремонтопридатність і відповідність архітектурним цілям системи.

Важливим аспектом архітектури є акцент на міжсервісній комунікації. Служби плавно взаємодіють через чітко визначені API, використовуючи ін'єкції залежностей та модульні моделі для забезпечення вільного зв'язку та високої згуртованості. Ця конструкція дозволяє системі масштабуватися горизонтально, з окремими послугами, здатними відтворюватися або незалежно масштабуватися на основі попиту.

У додатку Л наведено посилання на репозиторій коду на сайті GitHub.

Описаний архітектурний підхід узгоджується з загальними цілями системи, підкреслюючи стійкість, гнучкість та ефективність. Реалізовані принципи проектування, дозволяють системі розвиватися для покриття користувацьких потреб, зберігаючи високий рівень продуктивності та надійності.

9 РОЗРОБЛЕННЯ ІНТЕРФЕЙСУ

Цей розділ буде зосереджений на реалізації інтерфейсу користувача для системи, описуючи структуру, ключові компоненти та те, як вони працюють разом, щоб забезпечити безперешкодний користувальницький досвід. Особливу увагу буде приділено бібліотеці bpmn-js та призначеному для користувача модулю, розробленому для будування та редагування діаграм.

9.1 Структура

Користувальницький інтерфейс системи реалізований за допомогою React з TypeScript, що забезпечує динамічний і чуйний веб-додаток. Архітектура слідує модульній конструкції, гарантуючи, що кожен компонент має чітко визначену мету. Frontend зв'язується з backend сервісами, такими як Diagram Service і Telegram Bot Service, через RESTful API, що дозволяє безперешкодно керувати діаграмами і конфігураціями ботів.

Основна логіка програми знаходиться в директорії src, яка включає ключові компоненти, такі як App.js, DiagramList.js і DiagramEditor.js. Ці компоненти обробляють маршрутизацію, взаємодію користувачів і потік даних по всій програмі. Тека customModule містить налаштування для bpmn-js, яка є основною бібліотекою для рендерингу та редагування діаграм BPMN. Файли, такі як CustomPalette.js і CustomContextPad.js в цій папці, змінюють поведінку bpmn-js за замовчуванням, пристосовуючи її до вимог системи.

Програма має дві основні сторінки: сторінка списку діаграм, на якій відображаються доступні діаграми BPMN для вибору, та сторінка редактора діаграм, де користувачі можуть створювати, редагувати та призначати діаграми ботам. Статичні файли, такі як іконки і логотипи, зберігаються в загальнодоступному каталозі, в той час як основною точкою входу програми є index.js. У додатку И зображено діаграму активності, що описує дії потрібні для конфігурування процесу та призначення його боту Telegram.

На рисунку 9.1 зображено структуру проекту користувацького інтерфейсу.

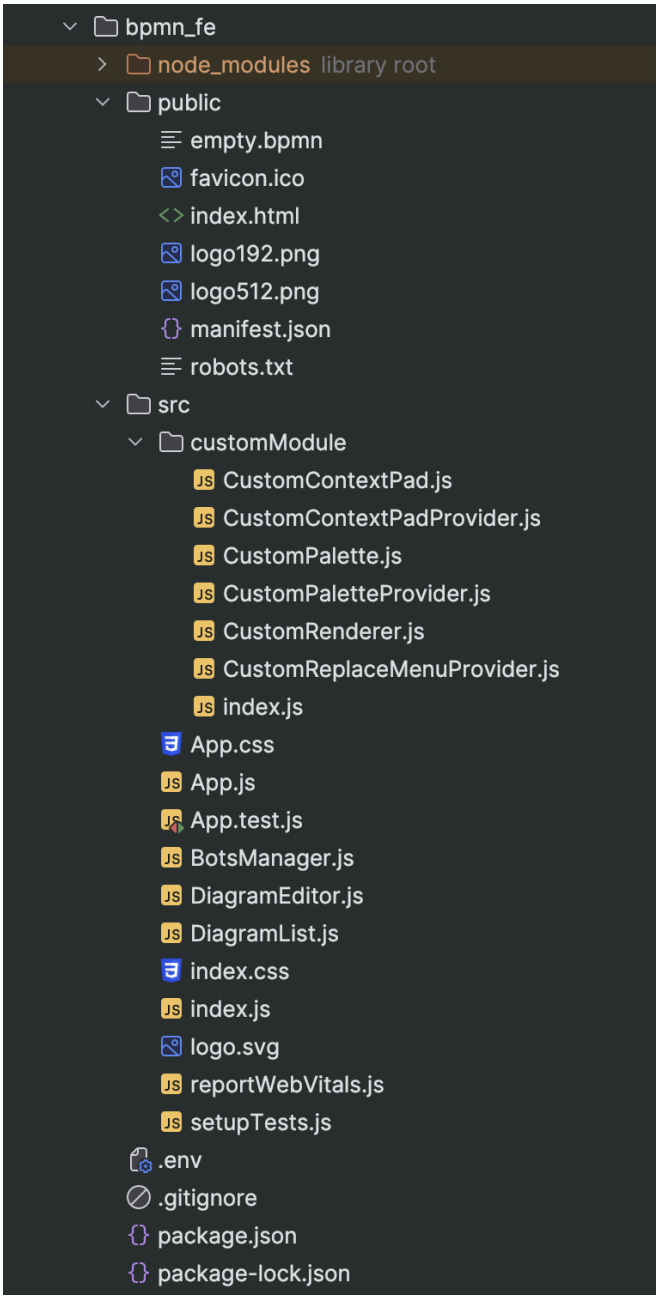

Рисунок 9.1 – Структура проекту користувацького інтерфейсу

9.2 Інтеграція BPMN-js

Система використовує BPMN-js, потужну бібліотеку для рендерингу та редагування діаграм BPMN, як основний інструмент для управління робочими процесами. Ця бібліотека надає візуальний інтерфейс для створення та

модифікації BPMN-сумісних діаграм, що дозволяє адміністраторам розробляти робочі процеси, не вимагаючи великих технічних знань. Його модульний і розширюваний характер робить його ідеальним вибором для застосування.

Для адаптації BPMN-js до конкретних вимог системи був розроблений спеціальний модуль. Цей модуль розширює типову функціональність бібліотеки та покращує користувацький досвід.

Налаштування палітри інструментів: файли CustomPalette.js та CustomPaletteProvider.js змінюють доступні інструменти в палітрі BPMN-js, забезпечуючи відображення лише потрібні активності та задачі [14-15], що мають відношення до системи. Це спрощує користувальницький інтерфейс і виключає зайву складність для адміністраторів.

```
77          'tool-separator': {
78              group: 'tools',
79              separator: true
80          },
81          'create.start-event': createAction(
82              type: 'bpmn:StartEvent',
83              group: 'event',
84              className: 'bpmn-icon-start-event-message',
85              translate('Create StartEvent'),
86              options: {eventDefinitionType: 'bpmn:MessageEventDefinition'}
87          ),
88          'create.intermediate-catch-message-event': createAction(
89              type: 'bpmn:IntermediateCatchEvent',
90              group: 'event',
91              className: 'bpmn-icon-intermediate-event-catch-message',
92              translate('Create StartEvent'),
93              options: {eventDefinitionType: 'bpmn:MessageEventDefinition'}
94          ),
95          'create.intermediate-catch-throw-event': createAction(
96              type: 'bpmn:IntermediateThrowEvent',
97              group: 'event',
98              className: 'bpmn-icon-intermediate-event-throw-message',
99              translate('Create IntermediateThrowEvent Event'),
100             options: {eventDefinitionType: 'bpmn:MessageEventDefinition'}
101         ),
```

Рисунок 9.2 – Приклад налаштування палітри інструментів

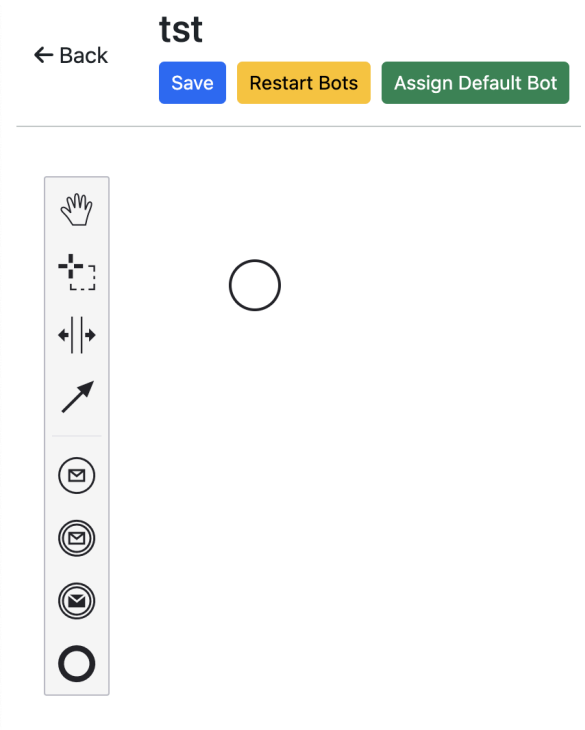

Рисунок 9.3 – Приклад палітри інструментів

На рисунку 9.2 зображено код налаштування палітри інструментів. На рисунку 9.3 зображено як ця палітра буде виглядати в веб інтерфейсі адміністратора.

Налаштування Context Pad: файли CustomContextPad.js і CustomContextPadProvider.js адаптують контекстну панель, яка з'являється при взаємодії з елементами BPMN.

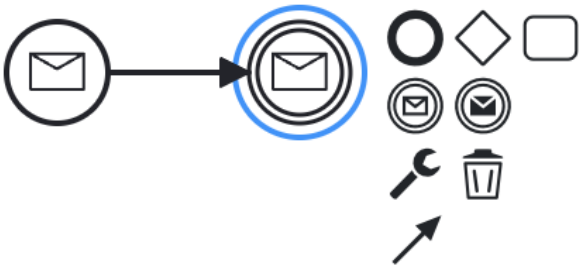

Рисунок 9.4 – Вигляд ContextPad

На рисунку 9.5 зображено код налаштування ContextPad. На рисунку 9.4 зображено вигляд ContextPad в веб інтерфейсі адміністратора.

```
let result : ContextPadEntries  = super.getContextPadEntries(element);
delete result["append.text-annotation"];
delete result["append.intermediate-event"];
if (element.type !== "bpmn:SequenceFlow") {
    assign(result, source: {
        'append.message-intermediate-event': appendAction(
            type: 'bpmn:IntermediateCatchEvent',
            className: 'bpmn-icon-intermediate-event-catch-message',
            translate('Append MessageIntermediateCatchEvent'),
            options: {eventDefinitionType: 'bpmn:MessageEventDefinition'}
        ),
        'append.message-intermediate-throw-event': appendAction(
            type: 'bpmn:IntermediateThrowEvent',
            className: 'bpmn-icon-intermediate-event-throw-message',
            translate('Append Intermediate/Boundary Event'),
            options: {eventDefinitionType: 'bpmn:MessageEventDefinition'}
        )
    })
}
return result;
```

Рисунок 9.5 – Налаштування ContextPad

Меню заміни: файл CustomReplaceMenuProvider.js налаштовує меню заміни, дозволяючи адміністраторам замінювати елементи BPMN, дотримуючись обмежень системи. Це гарантує, що діаграми залишаються послідовними і коректними.

9.3 Інтеграція Backend

Frontend легко взаємодіє з backend-сервісами, в першу чергу з Diagram Service і Telegram Bot Service, через RESTful API. Ця інтеграція дозволяє адміністраторам керувати діаграмами BPMN, зберігати зміни та налаштовувати призначення ботів, не вимагаючи прямого доступу до сервера. Використовуючи бібліотеки, такі як axios, асинхронні запити API ефективно обробляються, підтримуючи чуйний та інтерактивний користувальницький досвід.

Служба діаграм займає центральне місце в управлінні робочим процесом програми. Компонент DiagramEditor демонструє інтеграцію, де інтерфейс отримує

та завантажує діаграми BPMN, використовуючи їх ідентифікатори. Після редагування зміни зберігаються назад до сервера шляхом надсилання оновленого вмісту XML через запити HTTP PUT. Сторінка «Список діаграм» отримує всі доступні діаграми, щоб надати огляд, що дозволяє користувачам створювати або редагувати робочі процеси.

На рисунку 9.6 зображено код виклику Diagram Service для вилучення списку діаграм, для подальшого відображення як показано на рисунку 9.7.

```
useEffect( effect: () : void  => {
    const fetchDiagrams = async () : Promise<void>  => {
        const result : AxiosResponse<any>  = await axios(`${process.env.REACT_APP_DIAGRAM_BACKEND_URL}/diagrams/`);
        setDiagrams(result.data);
    };
    fetchDiagrams();
}, deps: []);
```

Рисунок 9.6 – Код відображення списку діаграм

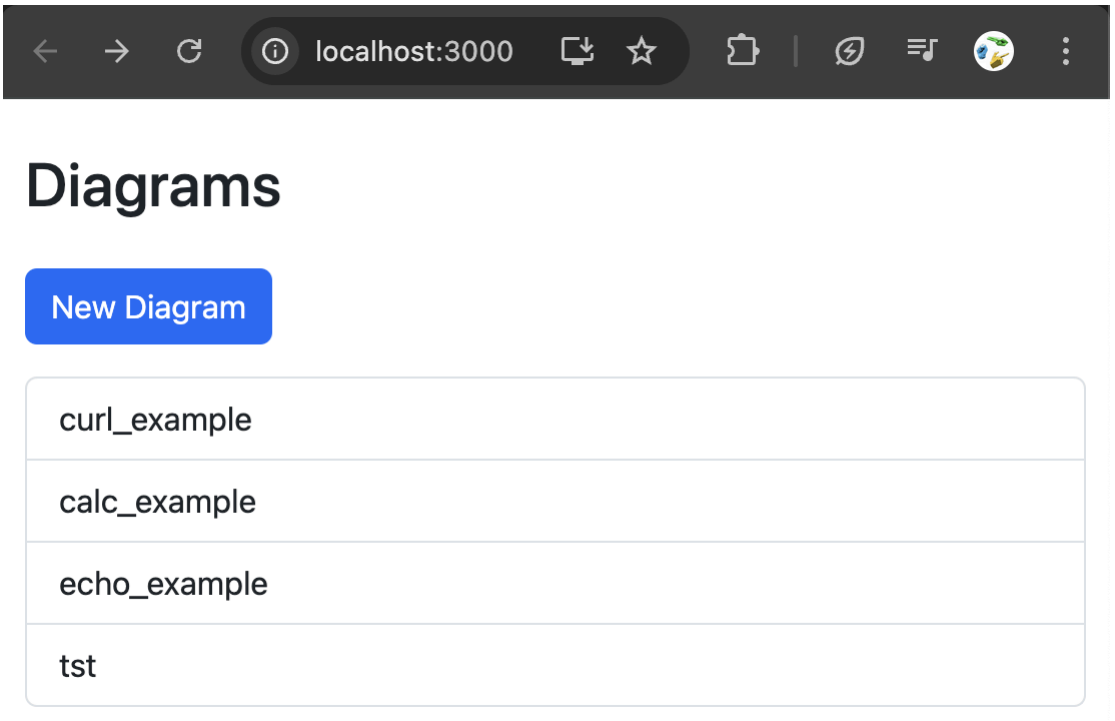

Рисунок 9.7 – Сторінка список діаграм

На рисунку 9.8 зображено код вилучення конкретної діаграми для подальшого її редагування. На рисунку 9.9 зображено вигляд сторінки редагування діаграми.

```
useEffect( effect: () : void  => {
    const fetchAndLoadDiagram = async () : Promise<void>  => {
        try {
            const response : AxiosResponse<any>  = await axios.get(
                url: `${process.env.REACT_APP_DIAGRAM_BACKEND_URL}/diagrams/${diagramId}`
            );
            const {xml, name} = response.data;
            await modelerInstance.current.importXML(xml);
            setDiagramName(name);
        } catch (error) {
            console.error('Failed to fetch or import diagram:', error);
        }
    };

    if (diagramId) {
        fetchAndLoadDiagram();
    }
}, deps: [diagramId]);
```

Рисунок 9.8 – Код завантаження та відображення конкретної діаграми

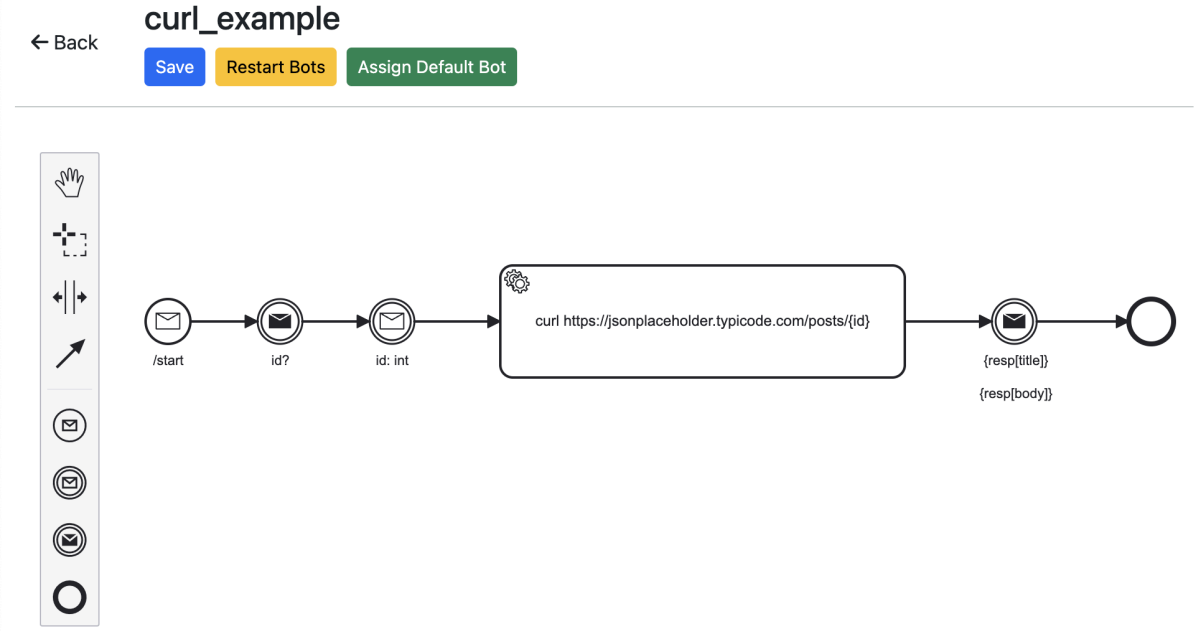

Рисунок 9.9 – Вигляд сторінки редагування діаграми

9.4 Інтеграція Telegram Bot Service

Telegram Bot Service дозволяє адміністраторам перезапустити ботів, пов'язаних з діаграмою, або призначити бота за замовчуванням для робочого процесу. Компонент ДіаграмаРедактор демонструє ці функції з діями,

викликаними кнопками. Виклики API здійснюються за допомогою запитів HTTP POST і PUT для виконання цих завдань.

На рисунку 9.10 зображено код призначення діаграми на виконання.

```
const assignDefaultBot = async () : Promise<void>  => {
    try {
        console.log('Assigning default bot', diagramId)
        await axios.put( url: `${process.env.REACT_APP_TG_BOT_BACKEND_URL}/bot/tg/assign_default/${diagramId}`);
    } catch (error) {
        console.error('Failed to assign default bot:', error);
    }
};
```

Рисунок 9.10 – Код призначення діаграми на виконання

Цей код відображає логіку призначення діаграми боту за замовчуванням. Це забезпечує надійність і простоту, абстрагуючи складнощі взаємодії бекенда від адміністратора.

Хоча перезапуск ботів також підтримується, основна увага приділяється забезпеченню динамічної конфігурації ботів для узгодження з оновленнями на діаграмах. Ця можливість необхідна для підтримки ефективних та адаптивних робочих процесів, що дозволяє адміністраторам керувати завданнями ботів з мінімальними зусиллями.

Інтеграція між інтерфейсом і Telegram Bot Service спрощує процес підтримки ботів у відповідності зі змінами робочого процесу, підвищуючи загальну гнучкість і зручність роботи системи.

9.5 Інтерфейс кінцевого користувача

Для кінцевих користувачів взаємодія відбувається повністю в додатку Telegram. Бот Telegram, оснащений бекендом і визначений діаграмами BPMN, направляє користувачів через різні процеси. Бот реагує на входи користувачів, збирає дані та виконує дії, зазначені на діаграмі BPMN. Цей підхід використовує знайомство і зручність Telegram, надаючи користувачам доступний і зручний для користувача досвід, не вимагаючи від них установки додаткових додатків. Приклад взаємодії з ботом показаний на рисунку 9.11.

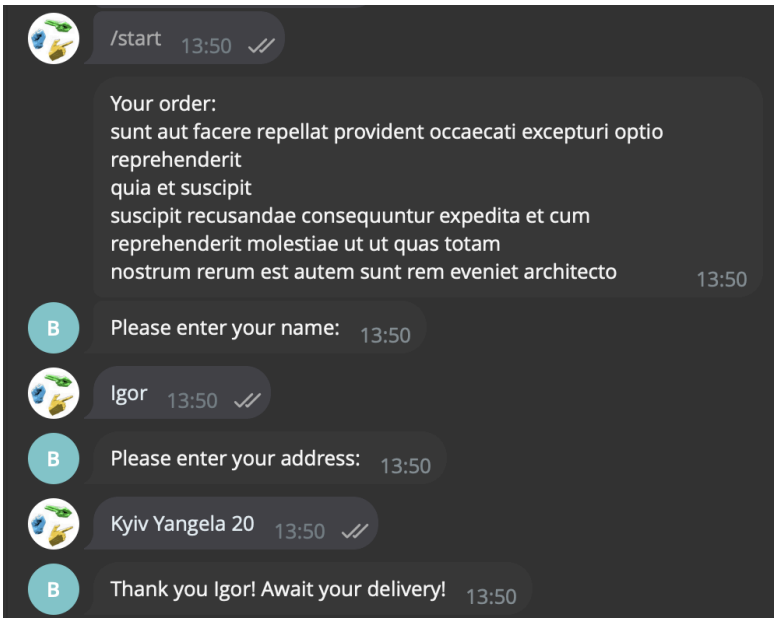

Рисунок 9.11 – Приклад взаємодії юзера з ботом

Висновки до розділу

Даний розділ демонструє реалізацію динамічного та адаптивного інтерфейсу за допомогою React, TypeScript та bpmn-js. Інтегруючи серверні служби, інтерфейс дозволяє адміністраторам керувати діаграмами BPMN та ефективно налаштовувати Telegram-ботів. Налаштування bpmn-js за допомогою customModule забезпечують індивідуальну функціональність, підвищуючи зручність редагування діаграм.

У додатку Л наведено посилання на репозиторій коду на сайті GitHub.

10 РОЗРОБЛЕННЯ СТАРТАП-ПРОЕКТУ

10.1 Опис ідеї проекту

Проект спрямований на створення інформаційної системи підтримки процесів доставки. Ця система дозволить оптимізувати процес доставки їжі шляхом автоматизації рутинних завдань, управління робочими процесами та забезпечення безперебійної комунікації між користувачами, кур'єрами та адміністраторами. Його ключовою особливістю є використання діаграм BPMN для управління робочими процесами доставки, забезпечуючи гнучкість і прозорість для бізнесу.

Таблиця 10.1 – Ідея проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Розробка інформаційної системи підтримки доставки їжі	Компанії з приготування їжі	Підвищення ефективності завдяки централізованому управлінню замовленнями та доставкою
	Послуги доставки їжі	Спрощена автоматизація процесів, відстеження в реальному часі та гнучкий дизайн робочого процесу

Запропонована інформаційна система доставки їжі забезпечує унікальні переваги перед конкурентами шляхом інтеграції гнучкого управління робочим процесом, відстеження в режимі реального часу та безперебійної обробки замовлень. Цей аналіз порівнює техно-економічні характеристики системи з конкуруючими продуктами, виділяючи сильні (S), нейтральні (N) і слабкі (W) аспекти для кожної функції.

Таблиця 10.2 – Визначення сильних, слабких та нейтральних характеристик ідеї проекту.

№ п/п	Техніко-економічні характеристики ідеї	(Потенційні) товари/концепції конкурентів	W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
1	Використання BPMN-діаграм для моделювання процесів	Традиційні системи управління замовленнями	Не всі клієнти знайомі з BPMN	Може потребувати навчання персоналу	Гнучкість і адаптивність процесів
2	Інтеграція з платформами доставки	Інші системи для керування логістикою	Складність налаштувань інтеграції	Відповідає стандартам інтеграції	Підтримка популярних платформ доставки
3	Централізоване управління замовленнями та звітністю	Ручне або напівавтоматизоване управління	Можлива висока початкова вартість	Стандартний функціонал для звітності	Скорочення часу на управління та підвищення точності

10.2 Технологічний аудит ідеї проекту

Технологічний аудит ідеї проекту має на меті оцінити доцільність реалізації концепції інформаційної системи підтримки процесів доставки продуктів

харчування. Це передбачає аналіз технологій, необхідних для реалізації, визначення їх доступності та оцінку того, чи доступні вони авторам проекту. Основна увага приділяється технологіям побудови веб-інтерфейсу, управління бізнес-процесами та інтеграції з платформами доставки.

Результати аналізу представлені в таблиці 10.3, в якій узагальнюються технології, пов'язані з кожним компонентом системи, їх поточним станом і їх доступністю для команди розробників.

Таблиця 10.3 – Технологічна здійсненність ідеї проекту

№ п/п	Ідея проєкту	Технології її реалізації	Наявність технологій	Доступність технологій
1	Реалізація веб-інтерфейсу для управління замовленнями	React, TypeScript, Bootstrap	Наявні	Доступні
2	Моделювання та виконання бізнес-процесів	BPMN-js, custom modules for integration	Наявні	Доступні
3	Інтеграція з платформами доставки	REST API, HTTP libraries (Axios, HTTPx)	Наявні	Доступні
4	Централізоване збереження даних	PostgreSQL, MongoDB	Наявні	Доступні
5	Мікросервісна архітектура для інтеграції компонентів	FastAPI, Docker, Kubernetes	Наявні	Доступні

Аудит демонструє, що всі необхідні технології доступні команді проекту. Обраний стек – React і TypeScript для front-end, BPMN-js для моделювання процесів і PostgreSQL для централізованого зберігання даних – забезпечує надійну і масштабовану розробку системи. Крім того, використання FastAPI і Docker полегшує мікросервісну архітектуру, що дозволяє ефективно інтегрувати всі компоненти.

Таким чином, проект є технологічно здійсненним, з необхідними технологіями, як легко доступними, так і здатними підтримувати функціональні можливості передбачуваної системи.

10.3 Аналіз ринкових можливостей запуску стартап-проекту

Аналіз ринкових можливостей оцінює доцільність впровадження системи підтримки доставки продуктів харчування на ринок. Досліджено попит, динаміку ринку, бар'єри входу та показники рентабельності для визначення привабливості та життєздатності ринку цього проекту. Ця інформація спрямовує стратегічне планування розробки проекту та визначає умови, необхідні для успішної реалізації.

Аналіз попиту включає оцінку кількості ключових гравців ринку, розміру ринку, динаміки та існуючих обмежень. Таблиця 10.4 узагальнює основні ринкові показники та забезпечує якісну оцінку потенційних можливостей та викликів.

Таблиця 10.4 – Попередня характеристика потенційного ринку стартап-проекту.

№ п/п	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, од	5–7 ключових компаній, що пропонують подібні послуги
2	Загальний обсяг продаж, грн/ум.од	Орієнтовно 500 млн грн/рік
3	Динаміка ринку (якісна оцінка)	Зростає

№ п/п	Показники стану ринку (найменування)	Характеристика
4	Наявність обмежень для входу (вказати характер)	Низькі: доступність технологій та відносно невисокі стартові інвестиції
5	Специфічні вимоги до стандартизації та сертифікації	Відповідність нормам безпеки даних та інтероперабельності API
6	Середня норма рентабельності в галузі (або по ринку), %	40–50%, що перевищує середній банківський відсоток

Аналіз показує, що ринок систем підтримки доставки продуктів харчування є привабливим для входу через зростаючий попит і відносно низькі бар'єри входу. Сектор демонструє стійку динаміку зі зростаючою тенденцією і значною прибутковістю. Ці фактори роблять ринок придатним для запуску проекту за умови, що рішення пропонує конкурентні переваги, такі як унікальні функції, зручність використання та рентабельність.

Потенційні групи клієнтів визначаються на основі їх потреб і поведінкових характеристик. Ці групи потім використовуються для формулювання попереднього списку вимог до продукту, щоб забезпечити його успіх на ринку. Таблиця 10.5 містить огляд цих груп клієнтів, їх поведінки та відповідних вимог до продукту.

Таблиця 10.5 – Характеристика потенційних клієнтів стартап-проекту

№ п/п	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1	Оптимізація	Ресторани,	Необхідність інтеграції	Зручний інтерфейс,

№ п/п	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
	процесу управління доставками	кафе, кухні з доставкою	з POS-системами, швидкий доступ до звітності	швидке впровадження, адаптація до існуючого програмного забезпечення
2	Покращення логістики та відстеження замовлень	Компанії з доставки їжі	Пріоритетність швидкої обробки замовлень, інтеграція з системами відстеження водіїв	Висока швидкодія системи, підтримка мобільних пристроїв, можливість масштабування
3	Централізація управління замовленнями	Мережеві заклади громадського харчування	Можливість роботи з декількома філіями, потреба у стандартизації обробки даних	Централізоване сховище даних, безпека інформації, можливість налаштувань під корпоративні потреби

Аналіз показує три ключові цільові аудиторії: окремі ресторани, компанії з доставки їжі та мережеві заклади. Кожна група має конкретні вимоги, на які впливає їх операційний масштаб, технологічна екосистема та пріоритети клієнтів. Вирішення цих різноманітних вимог через індивідуальні функції має важливе значення для успіху продукту.

Після виявлення потенційних груп клієнтів проводиться аналіз ринкового середовища. Це включає в себе складання факторів, які сприяють впровадженню проекту на ринок, а також факторів, які заважають йому. Ці фактори представлені в таблиці 10.6 (загрози) і таблиці 10.7 (можливості) в порядку зменшення значущості.

Таблиця 10.6 – Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1	Зростання конкуренції	Поява нових гравців на ринку зі схожим функціоналом	Унікальний функціонал та вдосконалення продукту
2	Зміни регуляторного середовища	Нові закони чи регуляції, що ускладнюють вихід на ринок	Постійний моніторинг законодавства та адаптація до вимог
3	Технічні обмеження	Складність інтеграції із застарілими системами клієнтів	Розробка універсальних інтерфейсів для інтеграції

Таблиця 10.7 – Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1	Розвиток ринку доставки їжі	Зростання популярності послуг доставки серед населення	Прискорення розробки та вихід на ринок
2	Технологічний розвиток	Нові технології, які полегшують	Використання сучасних рішень для оптимізації

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
		впровадження продукту	процесів
3	Розширення середнього бізнесу	Збільшення кількості потенційних клієнтів	Розробка продукту, орієнтованого на потреби середнього бізнесу

Аналіз пропозиції передбачає визначення загальних особливостей конкуренції на ринку, що допомагає визначити стратегії підтримки конкурентоспроможності. Цей крок класифікує конкурентне середовище та його вплив на проект. Таблиця 10.8 описує аналіз конкуренції на ринку.

Таблиця 10.8 – Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
Тип конкуренції	Монополістична конкуренція: декілька сильних гравців	Унікалізація функціоналу та активне просування на ринок
Рівень конкурентної боротьби	Національний	Вихід на локальні та регіональні ринки для побудови бази клієнтів
Галузева ознака	Внутрішньогалузева: конкуренція серед сервісів доставки їжі	Вивчення сильних та слабких сторін конкурентів, створення конкурентоспроможної ціни
Конкуренція за видами товарів	Товарно-видова: схожі платформи, але різний рівень сервісу	Вдосконалення інтерфейсу користувача та функціоналу
Характер конкурентних переваг	Нецінова: акцент на якості та зручності	Забезпечення високої якості роботи платформи, впровадження нових

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
		функцій
Інтенсивність	Не марочна: відсутність великих брендів у ніші	Активне просування бренду через маркетингові кампанії

Аналіз підкреслює, що ринок працює в умовах монополістичної конкуренції з сильними національними гравцями. Конкуренція в першу чергу не базується на цінах, підкреслюючи якість послуг та досвід користувачів. Щоб залишатися конкурентоспроможними, проект повинен зосередитися на створенні унікальної ціннісної пропозиції, активно брендуючи себе та орієнтуючись на недосвідчені регіональні ринки.

На основі аналізу конкуренції, характеристик проекту, вимог замовника та факторів маркетингового середовища, ми обґрунтовано перелік факторів, що забезпечують конкурентну перевагу проекту. Таблиця 10.9 деталізує обгрунтування кожного фактора.

Таблиця 10.9 – Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентоспроможності	Обґрунтування
1	Зручність використання платформи	Інтуїтивний інтерфейс і спрощена навігація залучають більшу кількість користувачів.
2	Швидкість обробки замовлень	Висока швидкодія забезпечує задоволення клієнтів, зменшуючи час очікування.
3	Унікальні функції для компаній	Можливість кастомізації під потреби бізнесу дозволяє обслуговувати різні цільові аудиторії.
4	Надійність і стабільність	Високий рівень доступності платформи з

№ п/п	Фактор конкурентоспроможності	Обґрунтування
	системи	мінімальними збоями створює довіру серед користувачів.
5	Розширений аналітичний функціонал	Забезпечує бізнес-клієнтів інструментами для аналізу ефективності доставки та клієнтських даних.
6	Підтримка мультиплатформенності	Доступність на різних пристроях (мобільні додатки, веб-версія) розширює охоплення аудиторії.

Виявлені фактори конкурентоспроможності демонструють потенціал проекту, щоб виділитися на ринку шляхом вирішення зручності користувача, продуктивності, надійності та налаштування. Ці атрибути відповідають вимогам клієнтів та тенденціям ринку, забезпечуючи стійкість та успіх проекту.

Використовуючи фактори конкурентоспроможності, визначені в таблиці 10.3.6, ми провели порівняльний аналіз сильних і слабких сторін стартап-проекту по відношенню до конкурентів. Ця оцінка висвітлює сфери, де проект перевершує або вимагає поліпшення. Таблиця 10.10 передбачає детальну розбивку порівняння.

Таблиця 10.10 – Порівняльний аналіз сильних та слабких сторін проекту

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів
1	Зручність використання платформи	18	+2
2	Швидкість обробки замовлень	17	+1
3	Унікальні функції для	19	+3

№ п/п	Фактор конкурентоспроможності компаній	Бали 1-20	Рейтинг товарів-конкурентів
4	Надійність і стабільність системи	16	-1
5	Розширений аналітичний функціонал	14	0
6	Підтримка мультиплатформенності	15	+1

Заключний етап аналізу ринкових можливостей реалізації проекту передбачає створення SWOT-матриці. Ця матриця визначає сильні сторони, слабкі сторони, можливості та загрози (SWOT) на основі раніше окреслених ринкових факторів, а також сильних і слабких місць проекту. В таблиці 10.11 наведено SWOT- аналіз стартап-проекту.

Таблиця 10.11 – SWOT- аналіз стартап-проекту

Сильні сторони	Слабкі сторони
Зручність користування платформою для різних категорій бізнесу; унікальні функції для покращення логістичних процесів; швидка інтеграція з уже існуючими системами клієнтів; висока продуктивність і швидкість обробки замовлень	Відсутність популярності на ринку (низька впізнаваність бренду); обмеженість аналітичного функціоналу у порівнянні з деякими конкурентами; залежність від стабільного інтернет-з'єднання
Можливості	Загрози

Зростання попиту на автоматизацію логістичних процесів; розширення ринку через інтеграцію з платформами доставок; залучення партнерів для розвитку нових функціоналів; використання сучасних технологій для зниження витрат	Посилення конкуренції з боку великих гравців; можливі економічні кризи або зниження купівельної спроможності; зміни у законодавстві, що впливають на логістичну сферу; технічні ризики

На основі SWOT-аналізу виявлено потенційні альтернативи поведінки на ринку. Ці альтернативи включають стратегічні дії для виведення стартап-проекту на ринок та оцінку оптимальних термінів розгляду проектів потенційних конкурентів. Таблиця 10.12 узагальнює запропоновані альтернативи, їх ймовірність придбання ресурсів та приблизні терміни реалізації.

Таблиця 10.12 – Альтернативи ринкового впровадження стартап-проекту

№ п/п	Альтернатива (орієнтовний комплекс заходів)	Ймовірність отримання ресурсів	Строки реалізації
1	Запуск продукту з базовим функціоналом, орієнтованим на локальні малі й середні підприємства	Висока	3 місяці
2	Розробка й впровадження додаткових функцій для інтеграції з великими платформами доставки (API, модулі аналітики)	Середня	6 місяців
3	Партнерство з іншими IT-компаніями для крос-продажів та спільної промоції	Низька	9 місяців

№ п/п	Альтернатива (орієнтовний комплекс заходів)	Ймовірність отримання ресурсів	Строки реалізації
4	Розширення платформи до міжнародних ринків через локалізацію продукту та маркетингові кампанії	Середня	12 місяців

Серед запропонованих альтернатив найбільш життєздатним для початкового виходу на ринок був обраний перший варіант – запуск продукту з основними функціями для малого та середнього бізнесу.

Доступність ресурсів: ця альтернатива має найвищу ймовірність забезпечення ресурсів через менші початкові інвестиції та простіші вимоги до впровадження.

Короткі терміни: 3-місячне вікно реалізації дозволяє швидко встановити присутність на ринку та отримати відгуки користувачів.

Масштабованість: запуск малого забезпечує основу для розширення функціональності та охоплення ринку на більш пізніх етапах, поєднуючи другу та четверту альтернативи зростання.

Цей покроковий підхід забезпечує баланс між ефективністю використання ресурсів та оперативністю реагування на ринок.

10.4 Розроблення ринкової стратегії проекту

Перший крок у розробці стратегії ринку передбачає виявлення цільових груп споживачів на основі їх профілів, готовності ринку, потенціалу попиту, інтенсивності конкуренції та простоти виходу на ринок. Таблиця 10.13 узагальнює аналіз.

Таблиця 10.13 – Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Малі та середні підприємства в сфері доставки продуктів харчування (локальний бізнес)	Висока	Високий	Помірна	Висока
2	Великі мережі доставки їжі (національні бренди)	Середня	Середній	Висока	Низька
3	Ресторани та кафе, які організовують доставку їжі	Висока	Високий	Помірна	Середня

Розробка ринкової стратегії встановила чітке розуміння цільових груп, їх потреб і ринкового потенціалу. Аналіз показав, що малий та середній бізнес у секторі доставки продуктів харчування та ресторанах є основним напрямком через високий попит та готовність прийняти запропоноване рішення.

Обрана диференційована маркетингова стратегія дозволяє індивідуально підходити до вирішення унікальних вимог цих груп, забезпечуючи масштабованість і релевантність. Ця стратегія позиціонує проект для успішного виходу на ринок і довгострокового зростання.

Розробка основної ринкової стратегії передбачає вибір оптимального підходу для захоплення цільових сегментів ринку, визначених у попередньому аналізі. Виходячи з конкретних потреб потенційних клієнтів і конкурентного середовища, проект вимагає чіткого фокусування на своїх унікальних сильних сторонах, ефективно задовольняючи вимоги клієнтів.

Обрана стратегія документується в таблиці 10.14, деталізуючи обгрунтування обраного шляху розвитку.

Таблиця 10.14 – Визначення базової стратегії розвитку

№ п/п	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
1	Створення унікального функціоналу та UX/UI	Диференційований маркетинг	Унікальний дизайн, інтеграція з платформами замовника, високий рівень техпідтримки	Стратегія диференціації

Стратегія диференціації вибирається як найбільш підходяща через необхідність унікальних функцій, які вирішують специфічні для клієнта проблеми в галузі доставки їжі. Такий підхід позиціонує продукт як преміальне рішення на конкурентному ринку.

Вибір стратегії конкурентної поведінки передбачає аналіз ринкової позиції проекту, виявлення його сильних і слабких сторін, а також визначення найбільш ефективного способу захоплення і підтримки частки ринку. Конкурентна стратегія повинна вирішувати, чи має проект на меті започаткувати свій сегмент ринку, кинути виклик конкурентам або слідувати існуючим лідерам.

Виходячи з особливостей проекту та конкурентного ландшафту, таблиця 10.15 окреслює обрану стратегію конкурентної поведінки.

Таблиця 10.15 – Визначення базової стратегії конкурентної поведінки

№ п/п	Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки
1	Ні	Компанія шукатиме нових споживачів та забере частку конкурентів	Базовий функціонал, який є критично важливим для клієнтів	Стратегія виклику лідера

Була обрана стратегія виклику лідера. Це передбачає націлювання на слабкі сторони конкурентів, такі як обмежена функціональність або зниження задоволеності клієнтів, одночасно підвищуючи критичні функції, такі як зручність використання та інтеграція. Цей підхід спрямований на створення проекту як конкурентної альтернативи, поступово захоплюючи частку ринку у існуючих гравців.

Позиціонування продукту на ринку передбачає узгодження його особливостей і можливостей з конкретними вимогами цільових сегментів клієнтів. Ця стратегія базується на вимогах споживачів, обраній стратегії розвитку та стратегії конкурентної поведінки. Позиціонування спрямоване на створення чіткого та незабутнього образу продукту, що відображає його конкурентні переваги та основні цінності.

Стратегія позиціонування формулюється як сукупність асоціацій, які визначають, як продукт сприймається клієнтами. Таблиця 10.16 узагальнює стратегію позиціонування.

Таблиця 10.16 – Визначення стратегії позиціонування

№ п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
1	Зручність, швидкодія, інтеграція	Стратегія диференціації	Висока якість обслуговування, простота у використанні, інноваційність	«Простота», «Ефективність», «Інновації»

Розроблена стратегія позиціонування гарантує, що проект виділяється на ринку, підкреслюючи його простоту використання, ефективність та інноваційний підхід. Ці елементи резонують з потребами цільової аудиторії та узгоджуються з більш широкими маркетинговими та конкурентними стратегіями, забезпечуючи чіткий напрямок ринкової поведінки компанії.

10.5 Розроблення маркетингової програми стартап-проекту

Першим кроком у розробці маркетингової програми є визначення концепції продукту, яка буде надавати цінність для споживача. Це передбачає узагальнення конкурентних переваг, визначених у попередньому аналізі, та узгодження їх із потребами цільової аудиторії. Концепція продукту повинна сформулювати основні переваги та унікальні точки продажу, які відрізняють пропозицію від конкурентів.

Таблиця 10.17 надає короткий огляд основних переваг продукту, узагальнюючи його відповідність потребам клієнтів, цінність, яку він забезпечує, і його конкурентну перевагу.

Таблиця 10.17 – Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)
1	Оптимізація доставки їжі	Швидкість і зручність процесу замовлення доставки	Інтуїтивний інтерфейс, автоматизація процесів, масштабованість
2	Підвищення ефективності роботи компаній	Зниження часу на організацію та координацію	Інтеграція з CRM, аналітичні звіти, налаштовувані функції
3	Забезпечення якості обслуговування	Централізоване управління і контроль	Можливість персоналізації, висока швидкодія, багатофункціональність

Маркетингова концепція визначає здатність продукту задовольняти критичні потреби клієнтів, підкреслюючи його переваги в автоматизації, інтеграції та продуктивності. Ці атрибути не тільки вирішують існуючі прогалини ринку, але й забезпечують основу для майбутніх удосконалень, забезпечуючи конкурентну перевагу на динамічному ринку.

Розробка трирівневої маркетингової моделі уточнює концепцію продукту, його відчутні елементи та особливості доставки. Ця модель висвітлює основну ціннісну пропозицію продукту, функціональні та фізичні атрибути на його сучасному етапі та додаткові послуги, які підвищують загальну пропозицію. Мета полягає в тому, щоб забезпечити структуровану основу для розуміння позиціонування продукту та його унікальних елементів, забезпечуючи відповідність потребам клієнтів та вимогам ринку.

Таблиця 10.18 описує три рівні моделі продукту, підкреслюючи його концептуальні, фізичні та допоміжні атрибути. Крім того, особлива увага

приділяється захисту продукту від реплікації через права інтелектуальної власності та унікальні функції.

Таблиця 10.18 – Опис трьох рівнів моделі товару

<table>
<tr><td>Рівні товару</td><td colspan="3">Сутність та складові</td></tr>
<tr><td>I. Товар за задумом</td><td colspan="3">Автоматизація процесів доставки їжі, зменшення витрат часу та ресурсів для компаній з доставки їжі.</td></tr>
<tr><td rowspan="5">II. Товар у реальному виконанні</td><td>Властивості/характеристики</td><td>М/Нм</td><td>Вр/Тх /Тл/Е/Ор</td></tr>
<tr><td>1. Інтуїтивно зрозумілий інтерфейс для користувачів;
2. Низькі витрати на впровадження та обслуговування;
3. Базові функції моніторингу та управління доставками;
4. Налаштовувані звіти для аналізу ефективності;
5. Підтримка різних платформ (мобільних та десктопних пристроїв).</td><td>-</td><td>-</td></tr>
<tr><td colspan="3">Якість: визначається через проведення детального тестування.</td></tr>
<tr><td colspan="3">Пакування: електрона документація</td></tr>
<tr><td colspan="3">Марка: DiagramRunner</td></tr>
<tr><td rowspan="2">III. Товар із підкріпленням</td><td colspan="3">До продажу: пробна версія з обмеженими функціями для демонстрації можливостей.</td></tr>
<tr><td colspan="3">Після продажу: постійні оновлення, технічна підтримка та кастомізація функціоналу під потреби клієнта.</td></tr>
<tr><td colspan="4">Унікальні алгоритми, патенти на ключові функції, захист інтелектуальної власності.</td></tr>
</table>

Трирівнева маркетингова модель гарантує, що продукт розроблений і позиціонується для задоволення потреб клієнтів, захищаючись від реплікації. Поєднання надійної концептуальної основи, чудових функціональних

характеристик та великої післяпродажної підтримки гарантує її конкурентоспроможність на ринку. Захист інтелектуальної власності та власні функції забезпечують додаткові гарантії довгострокового успіху проекту.

Наступний крок у розробці маркетингової програми передбачає визначення цінових меж для керівництва ціноутворенням потенційного продукту. У той час як остаточна ціна буде визначена під час фінансово-економічного аналізу, цей крок спрямований на встановлення початкового цінового діапазону шляхом оцінки цін на замінні та аналогічні продукти, а також рівня доходів цільової аудиторії. Цей аналіз проводиться з використанням експертних методів для забезпечення актуальності та точності.

Результати узагальнені в таблиці 10.19, забезпечуючи структурований огляд цінових міркувань, заснованих на ринкових умовах та доступності для споживачів.

Таблиця 10.19 – Визначення меж встановлення ціни

№ п/п	Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
1	300-600 грн/міс	300-1000 грн/міс	Середній рівень доходів: 20,000-40,000 грн/міс	Нижня межа: 350 грн/міс, Верхня межа: 500 грн/міс

Ця таблиця ілюструє розрахунковий діапазон цін на продукт, балансуючи доступність для цільової аудиторії з конкурентними ринковими умовами. Остаточна стратегія ціноутворення включатиме ці ідеї для оптимізації цінності для клієнтів та стійкості для бізнесу.

Наступний етап передбачає визначення оптимальної системи розподілу продукту. Цей процес включає в себе рішення про те, чи варто покладатися на внутрішні ресурси або залучати сторонніх посередників, оцінюючи оптимальну

глибину каналу збуту, і підбираючи найбільш підходящий тип посередників. Система дистрибуції повинна забезпечувати ефективну доставку на цільовий ринок при мінімізації витрат і максимізації доступності.

Таблиця 10.20 узагальнює стратегію розподілу, виділяючи поведінку покупців, функції розподілу ключів, глибину каналу та рекомендовану систему.

Таблиця 10.20 – Визначення меж встановлення ціни

№ п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
1	Бажання отримати швидкий доступ до послуги через онлайн-канали	Забезпечення онлайн-доступу до сервісу, підтримка клієнтів, інтеграція з платформами клієнтів	Однорівневий	Власна система збуту через веб-платформу
2	Перевага використання послуг через інтегровані партнерські сервіси	Надання API для інтеграції, забезпечення технічної підтримки	Дворівневий	Залучена система через партнерські мережі

Обрана система розподілу балансує необхідність прямого доступу клієнтів з масштабованістю, запропонованою партнерствами. Основний канал буде використовувати платформу компанії, забезпечуючи безперебійну взаємодію з клієнтами, а партнерські відносини розширюють охоплення ринку за допомогою додаткових послуг.

Заключним елементом маркетингової програми є розробка комунікаційної стратегії. Ця стратегія базується на раніше визначеній структурі позиціонування, моделях поведінки клієнтів та найбільш ефективних каналах комунікації. Мета – створити згуртоване повідомлення, яке підкреслює унікальні ціннісні пропозиції продукту та враховує конкретні потреби та переваги цільової аудиторії.

Таблиця 10.21 описує стратегію комунікації, деталізуючи поведінку клієнтів, бажані канали комунікації, позиціонування повідомлень, а також цілі та стиль реклами.

Таблиця 10.21 – Визначення меж встановлення ціни

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
1	Клієнти шукають швидкість та зручність під час замовлення послуг	Соціальні мережі, веб-сайт, мобільні додатки	Автоматизація, швидкість обслуговування	Показати переваги використання системи для швидкого та безпроблемного замовлення	«Ваш ідеальний партнер у швидкій та ефективній доставці — автоматизуйте свої процеси вже зараз!»
2	Прагнення отримати інтегровані рішення для щоденних потреб	Email-розсилки, інтегровані платформи	Комплексність рішень, легкість інтеграції	Підкреслити можливість інтеграції з існуючими системами клієнта	«Легкість інтеграції, повна автоматизація: ми робимо вашу доставку ефективнішою! »
3	Потреба у впевненості та підтримці	Прямі контакти, підтримка	Надійність, підтримка клієнтів	Створити відчуття довіри та	«Ми поруч на кожному етапі: від інтеграції до

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
	після придбання	через чат-боти		впевненості у сервісі	підтримки вашого бізнесу!»

Маркетингова комунікаційна стратегія поєднує в собі різні канали для ефективного досягнення різних сегментів клієнтів. Кампанія прагне встановити довіру та довгострокову взаємодію з цільовою аудиторією.

Висновки до розділу

Аналіз підтверджує, що проект має потужний потенціал для комерціалізації ринку. Очевидний попит на автоматизовані рішення управління доставкою, підкріплені позитивною динамікою ринку і високими показниками прибутковості в цільовому секторі. Це створює чітку можливість для успішного виходу на ринок.

Завдяки сприятливим умовам, таким як мінімальні бар'єри входу, чітко визначена цільова аудиторія та конкурентні переваги, такі як автоматизація, надійність та безшовна інтеграція, проект демонструє перспективи реалізації. Запропонована стратегія масового маркетингу забезпечує ефективний шлях для впровадження ринку.

З огляду на узгодження особливостей проекту з потребами ринку та його здатність вирішувати ключові больові точки клієнтів, настійно рекомендується подальший розвиток та впровадження. Проект готовий досягти успішної комерціалізації та довгострокової прибутковості, забезпечуючи міцну основу для майбутнього зростання та розширення.

ВИСНОВКИ

Дисертація присвячена розробці інформаційної системи, призначеної для підвищення ефективності операцій з доставки продуктів харчування. Зі зростанням попиту на послуги доставки, проект вирішує своєчасну та значну проблему, пропонуючи надійний інструмент для оптимізації процесів доставки.

Аналіз предметної області та існуючих рішень виявив ключові прогалини на сучасному ринку, зокрема в автоматизації, зручності використання та економічності. На основі цієї інформації визначено функціональні і нефункціональні вимоги до інформаційної системи, забезпечуючи узгодження з потребами потенційних користувачів і зацікавлених сторін.

Описано математичну основу для підтримки діяльності служби доставки продуктів харчування.

Проект також використовує модульну архітектуру, використовуючи передові технології, такі як Python для розробки backend і TypeScript для інтерфейсів frontend. Бази даних були розроблені допомогою PostgreSQL і MongoDB.

Система використовує діаграми BPMN для автоматизації процесів доставки, що дозволяє підприємствам моделювати та виконувати свої процеси з точністю. Включивши ці діаграми, система спрощує складні операції доставки, що дозволяє легше адаптуватися до конкретних організаційних потреб.

Дослідження ринку підтвердили доцільність комерціалізації проекту. Виявлена цільова аудиторія, включаючи компанії з доставки їжі та ресторани. Конкурентний аналіз ще більше підтвердив потенціал проекту, підкреслюючи його диференціацію за допомогою таких функцій, як автоматизація та масштабованість.

Ця робота успішно вирішує проблеми проектування та впровадження складної системи з урахуванням потреб сектора доставки. Рішення пропонує відчутні переваги для бізнесу шляхом оптимізації робочих процесів, зменшення помилок та підвищення загальної ефективності. Майбутній розвиток може містити додаткові інтеграції та розширену аналітику для подальшого підвищення якостей системи.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. К.С. Клен. Методи моделювання інформаційних систем. Ресурс доступний за адресою: https://ela.kpi.ua/server/api/core/bitstreams/165549aa-60cc-4eb7-94a8-be3fe391fbdd/content.

2. Hashmap, an NTT DATA Company. The What, Why, and How of a Microservices Architecture. Ресурс доступний за адресою: https://medium.com/hashmapinc/the-what-why-and-how-of-a-microservices-architecture-4179579423a9.

3. Документація FastAPI. Ресурс доступний за адресою: https://fastapi.tiangolo.com/.

4. Документація Aiogram. Ресурс доступний за адресою: https://docs.aiogram.dev/.

5. Документація Pydantic. Ресурс доступний за адресою: https://docs.pydantic.dev/latest/.

6. Martin Fowler. Inversion of Control Containers and the Dependency Injection pattern. Ресурс доступний за адресою: https://martinfowler.com/articles/injection.html.

7. Документація TypeScript. Ресурс доступний за адресою: https://www.typescriptlang.org/docs/.

8. Документація React. Ресурс доступний за адресою: https://react.dev/.

9. Документація BPMN-js. Ресурс доступний за адресою: https://bpmn.io/toolkit/bpmn-js/.

10. Sergii Telenyk, Grzegorz Nowakowski, Yevhenii Vovk, Ihor Anosov Розвиток і реалізація технології створення широкого класу застосувань на зразок чат-ботів на основі формальних моделей. (2022): Наукові записки НаУКМА. Комп’ютерні науки Том 5 с.97-107.

11. Документація PostgreSQL. Ресурс доступний за адресою: https://www.postgresql.org/docs/.

12. Документація MongoDB. Ресурс доступний за адресою: https://www.mongodb.com/docs/.

13. Ignaty Kashnitsky. Building Chatbots with Python and Aiogram 3.x. Ресурс доступний за адресою: https://python.plainenglish.io/5-tips-for-building-chatbots-with-python-and-aiogram-3-x-d34feeb18d2f.

14. Visual Paradigm. BPMN Events. Ресурс доступний за адресою: https://www.visual-paradigm.com/guide/bpmn/bpmn-events/.

15. BPMN Activity Types Explained. Ресурс доступний за адресою: https://www.visual-paradigm.com/guide/bpmn/bpmn-activity-types-explained/.