Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційних радіотехнологій і технічного захисту інформації
(повна назва)

Кафедра Радіотехнологій інформаційно–комунікаційних систем
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти **перший (бакалаврський)**

СИСТЕМА ОБРОБКИ ЗАМОВЛЕНЬ В ЗАКЛАДАХ ХАРЧУВАННЯ
(тема)

Виконала:
студентка **4** курсу, групи **ІТІР–20–1**
Хаустова В. С.
(прізвище, ініціали)

Спеціальність **126 Інформаційні системи та технології**
(код і повна назва спеціальності)

Тип програми освітньо–професійна

Освітня програма **Інформаційні технології інтернету речей**
(повна назва освітньої програми)

Керівник **ст. викл. каф. РТІКС Ганшин Д. Г.**
(посада, прізвище, ініціали)

Допускається до захисту

В.о.зав. кафедри _____ **Зарудний О.А.**
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційних радіотехнологій і технічного захисту інформації

Кафедра Радіотехнологій інформаційно–комунікаційних систем

Рівень вищої освіти перший (бакалаврський)

Спеціальність 126 Інформаційні системи та технології

(код і повна назва)

Тип програми освітньо–професійна

Освітня програма інформаційні системи інтернету речей

(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____

(підпис)

«____»_____ 2024 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові **ХАУСТОВІЙ ВІКТОРІЇ СЕРГІЇВНИ**

(прізвище, ім'я, по батькові)

1. Тема роботи **Система обробки замовлень в закладах харчування**

затверджена наказом університету від **27 травня 2024 р. № 500 Ст**

2. Термін подання студентом роботи до екзаменаційної комісії **10 червня 2024 р.**

3. Вихідні дані до роботи

3.1 Провести аналіз існуючих месенджерів для впровадження боту в систему автоматизованого замовлення та обслуговування.

3.2 Розглянути основні можливості та функціонал Telegram Bot API для розробки Телеграм бота.

3.3 Вивчити характеристики, особливості програмування та можливості мікроконтролера ESP-32 для взаємодії з сенсорним дисплеєм та іншими компонентами системи .

3.4 Дослідити технології та фреймворки, такі як Spring Boot, використані для створення серверної частини, їхні особливості, переваги та недоліки.

3.5 Розробити програмне забезпечення для роботи системи.

4. Перелік питань, що потрібно опрацювати в роботі

Вступ. 1. Автоматизація в сфері харчування. 2. Технологічні аспекти розробки системи. 3. Розробка та реалізація системи. Висновки. Перелік джерел посилання. Додатки.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів)
Комп'ютерна презентація у форматі Power Point

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Основна частина	ст. викл. Ганшин Д. Г.		

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Вступ	06.05.2024	виконано
2	Автоматизація в сфері харчування	08.05 – 12.05.2024	виконано
3	Технологічні аспекти розробки системи	12.05 – 17.05.2024	виконано
4	Розробка та реалізація системи	17.05 – 01.06.2024	виконано
5	Аналіз результатів і ефективності системи	01.06.–04.06.2024	виконано
6	Висновки	04.06.2024	виконано
7	Оформлення пояснювальної записки	05.06.2024	виконано
8	Оформлення презентаційного матеріалу	05.06.2024	виконано
9	Представлення роботи на кафедру	10.06.24	виконано

Дата видачі завдання 06 травня 2024 р.

Студентка _____ Хаустова В. С
(підпис) (прізвище, ініціали)

Керівник роботи _____ ст. викл. Ганшин Д. Г.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Кваліфікаційна робота бакалавра складається з пояснювальної записки, що містить 100 сторінок тексту, 59 рисунків, 2 таблиці, 10 джерел посилання і 5 додатків.

TELEGRAM, ЧАТ-БОТ, ОБСЛУГОВУВАННЯ, ЗАМОВЛЕННЯ, КЛІЄНТ, ДИСПЛЕЙ

Об’єктом розробки є система автоматизованого обслуговування клієнтів у закладі харчування за допомогою Телеграм бота.

Метод дослідження – описово-аналітичний.

Метою даної кваліфікаційної роботи бакалавра є схемотехнічна та програмна розробка автоматизованої системи обслуговування клієнтів у сфері харчування.

В рамках дослідження була розроблена система на основі Телеграм бота, що дозволяє клієнтам робити замовлення, отримувати інформацію про меню, акції та спеціальні пропозиції. Для реалізації цього була використана мова програмування Java та ряд технологічних рішень, таких як Spring Boot, Telegram Bot API, MySQL та інші.

Процес взаємодії між клієнтом та системою відбувається у реальному часі: клієнт сканує QR-код, переходить до Телеграм боту, робить замовлення, сервер обробляє запит, зберігає дані в базі даних і відправляє підтвердження клієнту. Додатково, для оптимізації взаємодії співробітників кухні було впроваджено мікроконтролер ESP-32 для взаємодії з дисплеєм Raspberry Pi.

ABSTRACT

The qualification work of the bachelor consists of an explanatory note containing 100 pages of text, 59 figures, 2 tables, 10 literary sources and 5 appendices.

TELEGRAM, CHAT-BOT, SERVICE, ORDER,
CUSTOMER, DISPLAY

The object of development is a system of automated customer service in a catering establishment using a Telegram bot.

Research method - descriptive and analytical.

The purpose of this bachelor's thesis is the circuitry and software development of an automated customer service system in the field of catering.

As part of the study, a system based on a Telegram bot was developed that allows customers to place orders, receive information about menus, promotions, and special offers. To implement this, the Java programming language and a number of technological solutions such as Spring Boot, Telegram Bot API, MySQL, and others were used.

The process of interaction between the client and the system takes place in real time: the client scans a QR code, goes to the Telegram bot, places an order, the server processes the request, saves the data in the database, and sends a confirmation to the client. Additionally, to optimize the interaction of kitchen staff, an ESP-32 microcontroller was implemented to interact with the Raspberry Pi display.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧОК, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API (Application Programming Interface) – інтерфейс програмування застосунків;

HTTP (HyperText Transfer Protocol) – протокол передачі гіпертексту;

JDBC (Java Database Connectivity) – підключення до баз даних Java;

JSON (JavaScript Object Notation) – формат обміну даними;

JPA (Java Persistence API) – інтерфейс управління постійними об'єктами в Java;

OOP (Object-Oriented Programming) – об'єктно-орієнтоване програмування;

QR (Quick Response) - двовимірний штрих-код, який використовується для швидкого зчитування інформації;

SQL (Structured Query Language) – декларативна мова програмування для взаємодії користувача з базами даних;

URL (Uniform Resource Locator) – адреса сторінки в Інтернеті;

Wi–Fi (Wireless Fidelity) – технологія бездротової локальної мережі;

БД – база даних;

ОС – операційна система;

ПЗ – програмне забезпечення;

ПК – персональний комп’ютер.

ВСТУП

У сучасному світі роль технологій в галузі харчування постійно зростає, надаючи закладам нові можливості для покращення обслуговування та залучення клієнтів. Швидкий технологічний розвиток призвів до того, що клієнти стають все вимогливішими до якості послуг і зручності обслуговування. Відповідно, закладам харчування потрібно адаптуватися до цих змін, пропонуючи інноваційні рішення, що задовольняють потреби сучасного споживача.

Однією з актуальних тенденцій є використання месенджерів для взаємодії з клієнтами. Телеграм, як один з найпопулярніших месенджерів, надає унікальні можливості для створення автоматизованих рішень, таких як Телеграм боти. Ці боти можуть оптимізувати процеси обслуговування, забезпечуючи зручну та ефективну взаємодію з клієнтами, що може підвищити конкурентоспроможність закладу на ринку.

Метою даної дипломної роботи була розробка система на основі Телеграм боту для автоматизації обслуговування клієнтів закладу харчування. Система, яка була розроблена, мала на меті покращити якість обслуговування, збільшити зручність для клієнтів та оптимізувати робочі процеси. Вона спрямована на вдосконалення взаємодії клієнтів з закладом, зокрема, у замовленні та отриманні інформації про статус замовлення.

1 АВТОМАТИЗАЦІЯ В СФЕРІ ХАРЧУВАННЯ

1.1 Визначення автоматизації та роль в сфері харчування

Автоматизація – це процес застосування технологій, механізмів та систем, які забезпечують автоматичне виконання робіт та операцій без значущого втручання людини. У контексті галузі харчування, автоматизація може охоплювати широкий спектр функцій, від приготування їжі до управління клієнтськими замовленнями та складанням розкладів роботи персоналу.

Автоматизація в галузі харчування відіграє ключову роль у покращенні ефективності, збільшенні продуктивності та підвищенні якості обслуговування. Ось деякі аспекти ролі автоматизації в галузі харчування:

– покращення ефективності процесів: автоматизовані системи можуть значно збільшити швидкість виконання рутинних операцій, таких як обробка замовлень, підготовка інгредієнтів або розрахунок запасів, дозволяючи персоналу зосередитися на більш складних завданнях;

– мінімізація помилок: автоматизовані системи дозволяють покращити точність та надійність процесів, зменшуючи ризик помилок, пов'язаних з людським фактором;

– зменшення витрат: хоча впровадження автоматизованих систем вимагає певних витрат на початковому етапі, в майбутньому це може призвести до зменшення загальних витрат завдяки ефективнішому використанню ресурсів та скороченню часу на виконання операцій;

– підвищення задоволення клієнтів: завдяки автоматизованим системам клієнти можуть отримувати швидше та точніше обслуговування, а також зможуть з легкістю розміщувати замовлення або звертатися за допомогою до підтримки;

– адаптація до технологічних тенденцій: в сучасному світі технології швидко розвиваються, і автоматизація дозволяє закладам харчування легко адаптуватися до нових технологічних рішень та інновацій.

1.2 Історія та еволюція автоматизації в закладах харчування

Автоматизація в галузі харчування не є новітнім явищем, але її роль та масштабність впровадження з часом значно зросли. Вивчення історії автоматизації в закладах харчування дозволяє нам краще розуміти, як технології впливали на сферу, а також прогнозувати її майбутній розвиток.

Гарним практичним досвідом використання подібної системи у сфері харчування являється мережа McDonald's. Ще у 1983 році мережа вперше використовувала самообслуговування в деяких ресторанах США, де були встановлені спеціальні термінали для замовлення, що дозволяли клієнтам самостійно вибирати та оплачувати свої блюда.

У 2015 р. McDonald's в Україні вирішив впровадити самообслуговування. Конструкція терміналів самообслуговування створювалася на світовому рівні і є стандартною для всіх країн - їх розробником стала міжнародна компанія Wincor Nixdorf.

Відвідувачі McDonald's отримали можливість здійснювати покупки без участі касира. Тепер покупці за допомогою спеціалізованих терміналів самообслуговування можуть самостійно вибрати та сплатити покупку банківською картою. Термінал пропонує інтерактивний доступ до меню, вибір типу замовлення (з собою або на місці), типу оплати (крім безготівкового розрахунку через сам термінал, замовлення можна також оплатити готівкою при його отриманні), та передає інформацію про замовлення співробітникам McDonald's, які відразу ж починають його збірку. Отримання замовлення відбувається в спеціально обладнаній зоні видачі [1].

Ознайомитись із прикладом такого терміналу можна на рисунку 1.1.

Рисунок 1.1 – Термінал самообслуговування у мережі McDonald's

На початку 2023 року в ТРЦ «LAVINA MALL» у Києві з’явився інтерактивний фуд-корт, який надає широкий спектр можливостей для комунікацій з клієнтами. Технологічну складову забезпечує інноваційна компанія Kodisoft.

Інтерактивний стіл дає можливість органічно впроваджувати різноманітні сервіси для відвідувачів: від замовлення, доставки та оплати їжі з кафе або ресторану до інформування користувачів про діючі акції, пропозиції різноманітних ігор, друку фото, замовлення музики та привітань (рисунки 1.2 і 1.3) [2].

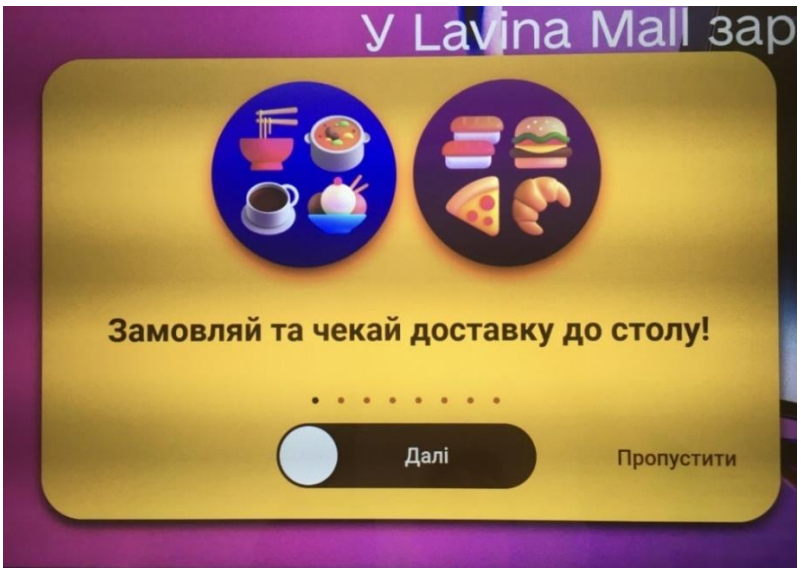


Рисунок 1.2 – Інтерактивний стіл, етап замовлення їжі

Рисунок 1.3 – Інтерактивний стіл, розважальні функції

Впровадження технологій самообслуговування, подібних тим, що запроваджувалися McDonald's, та інноваційних рішень, які з'являються в сучасних інтерактивних фуд-кортах, визначає не лише способи обслуговування клієнтів, але й їх очікування та способи взаємодії з харчовими закладами. Такий прогрес в автоматизації відображає важливість подальших досліджень та розвитку технологій для оптимізації обслуговування клієнтів у галузі харчування [2].

1.3 Переваги та недоліки автоматизації для закладів харчування

Переваги автоматизованих систем для закладів харчування:

– автоматизовані системи дозволяють оптимізувати робочі процеси, зменшуючи час на обробку замовлень та підвищуючи продуктивність персоналу;

– автоматизація дозволяє уникнути помилок під час обробки замовлень та забезпечити більш точну інформацію про замовлення;

– зручність та швидкість обслуговування за допомогою автоматизованих систем позитивно впливають на задоволеність клієнтів та забезпечують їм позитивний досвід від відвідування закладу;

– впровадження автоматизованих систем дозволяє закладам харчування відповідати сучасним вимогам ринку та використовувати нові технології для поліпшення бізнесу.

Недоліки автоматизованих систем для закладів харчування:

– початкові витрати на придбання та налаштування автоматизованих систем можуть бути значними, особливо для невеликих закладів;

– впровадження нової технології часто потребує часу та ресурсів на навчання персоналу, що може бути складним у разі великої текучості кадрів;

– автоматизовані системи можуть бути вразливими до збоїв чи перебоїв з електроживленням, що може призвести до негативних наслідків у роботі.

1.4 QR-сервіси автоматизованої обробки замовлень

QR-сервіси автоматизованої обробки замовлень набирають популярності серед закладів харчування, пропонуючи інноваційні рішення для покращення обслуговування клієнтів. Використання таких сервісів дозволяє не лише прискорити процес замовлення та оплати, але й забезпечити більш зручну та безпечну взаємодію між закладом та відвідувачами. Два провідних QR-сервіси в Україні — Expirenza 2.0 by Mono та Choice QR.

1.4.1 Expirenza 2.0 by Mono

Expirenza 2.0 by Mono – це інноваційна платформа для управління досвідом клієнтів, розроблена компанією Mono. Ця система надає ресторатору та його клієнтам зручні інструменти для покращення взаємодії та обслуговування, використовуючи технології безконтактних послуг. Приклад сканування QR-коду закладу, що використовує цю платформу зображено на рисунку 1.4.

Рисунок 1.4 – Сканування QR-коду

Можливості Expirenza 2.0 [3]:

– користувачі можуть переглядати їжлисти — тематичні добірки страв, створені відомими ресторатори та шеф-кухарями, або ж створювати і ділитися власними добірками (рисунок 1.5);

– клієнти можуть переглядати меню ресторанів, зберігати страви, які вони хочуть скуштувати в майбутньому, що робить процес вибору страв простішим і зручнішим;

– за допомогою кількох натискань клієнти можуть бронювати столи, уникаючи зайвого очікування та забезпечуючи собі місце в улюбленому ресторані (рисунок 1.6);

– відвідувачі можуть сплачувати рахунок і залишати чайові без необхідності чекати на офіціанта, що значно спрощує процес розрахунку та робить його більш зручним (рисунок 1.7);

– клієнти можуть переглядати свої минулі замовлення в історії їжі, що дозволяє їм легко повторити замовлення або згадати, що їм сподобалось у минулому.

Рисунок 1.5 – Тематичні добірки страв

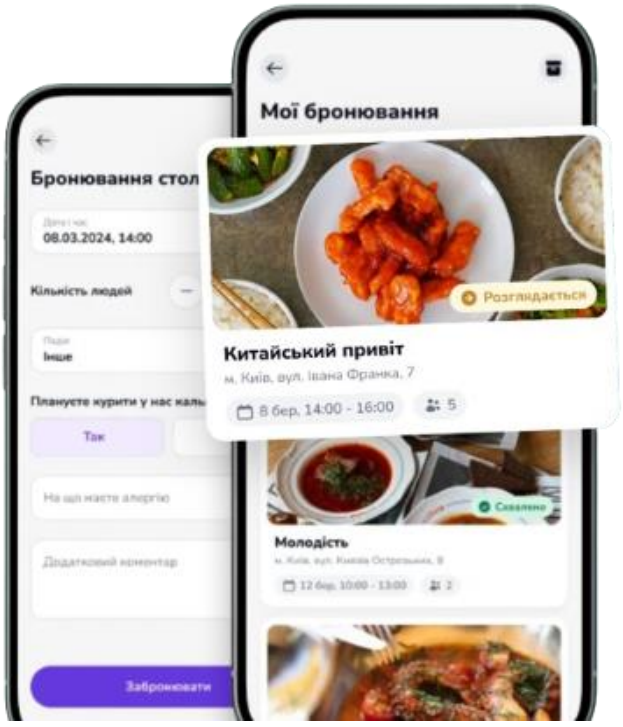


Рисунок 1.6 – Бронювання стола

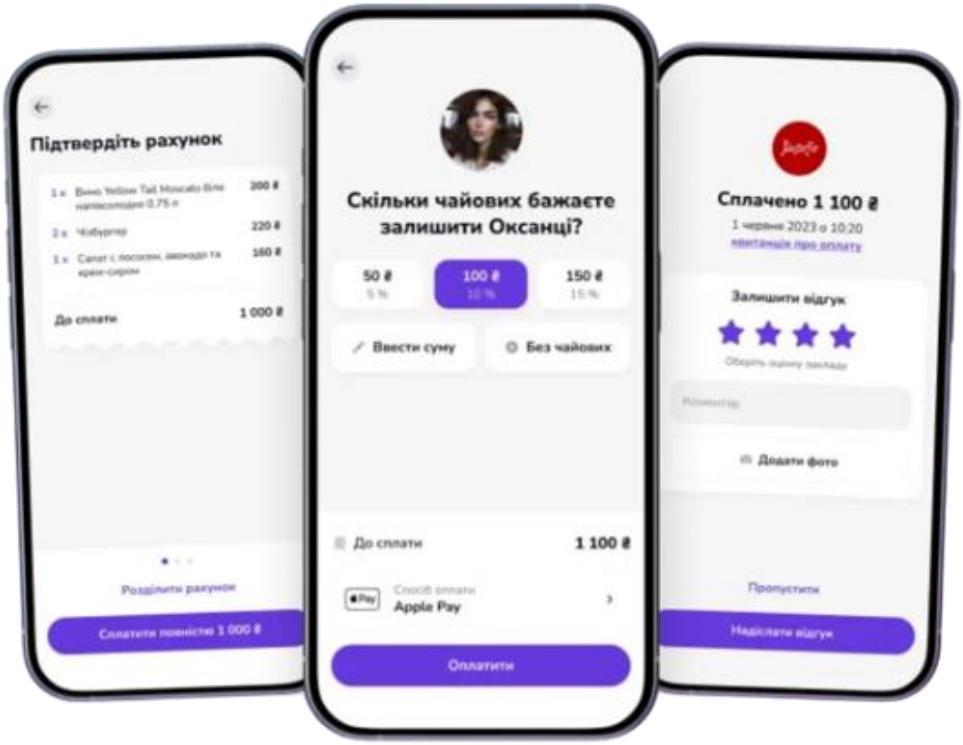


Рисунок 1.7 – Оплата за рахунок

Сервіс Expirenza 2.0 використовує єдиний QR-код, який об'єднує кілька важливих функцій, а саме: безконтактне меню, оплату і відгуки. Ознайомитись з інтерфейсом сервісу можна на рисунку 1.8 [3]:

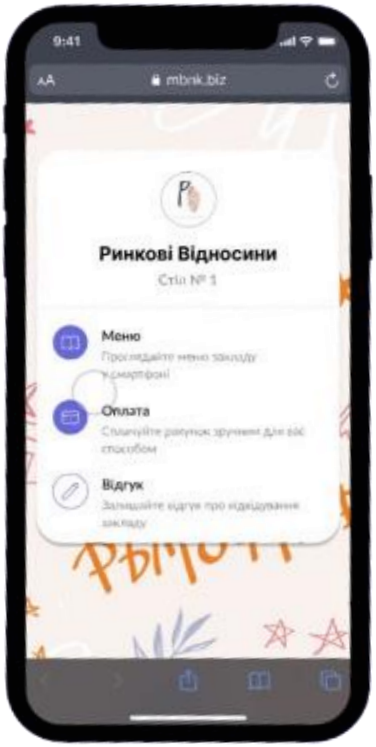


Рисунок 1.8 – Інтерфейс сервісу Expirenza 2.0

Сервіс Expirenza 2.0 by Mono доступний клієнтам усіх банків, що робить його універсальним рішенням для широкого кола користувачів. Завдяки цьому сервісу заклади харчування можуть значно покращити якість обслуговування, роблячи процес взаємодії з клієнтами більш зручним, швидким та ефективним.

1.4.2 Choice QR

Choice QR пропонує інноваційні рішення для покращення сервісу та підвищення ефективності роботи підприємств. Ця передова платформа дозволяє закладам громадського харчування, роздрібним магазинам та іншим бізнесам використовувати можливості QR-кодів для оптимізації процесів замовлення, оплати та отримання зворотного зв'язку від клієнтів.

Переваги використання для гостей [4]:

– близько 20-30% часу гостя витрачається саме на очікування, через це псується враження. Через QR-код гість оплачує, коли йому зручно (рисунок 1.9);

– з оплатою біля столу за QR-кодом гість завжди може залишити зручну для нього суму через онлайн-платіж. Заклад та персонал отримують більше чайових, а гість не потрапить у ситуацію, коли просто немає готівки залишити на знак подяки;

– меню працює за QR-кодом або посиланням – гостям не потрібні додатки. Вони відразу бачать усі ваші пропозиції з яскравими фото, відео, детальним описом (рисунок 1.10);

– гостям не потрібно завантажувати програму або реєструватися – отримати рахунок та сплатити можна за 10 секунд;

– після відображення рахунку за весь стіл, гості можуть розділити рахунок та залишити індивідуально чайові (рисунок 1.11);

– wishlist для зручності клієнтів та економії часу офіціанта гість може додати будь-яку страву з меню до «Списку бажань», а потім просто показати офіціанту на своєму смартфоні (рисунок 1.12).

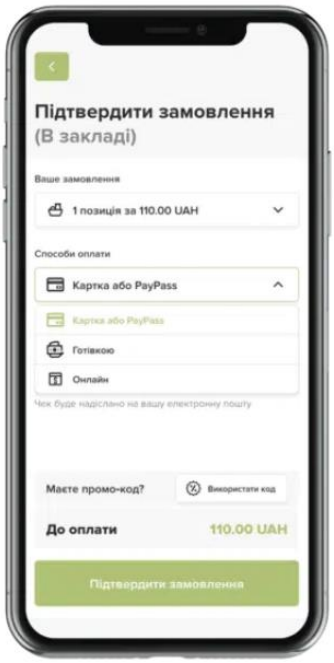


Рисунок 1.9 – Оплата замовлення

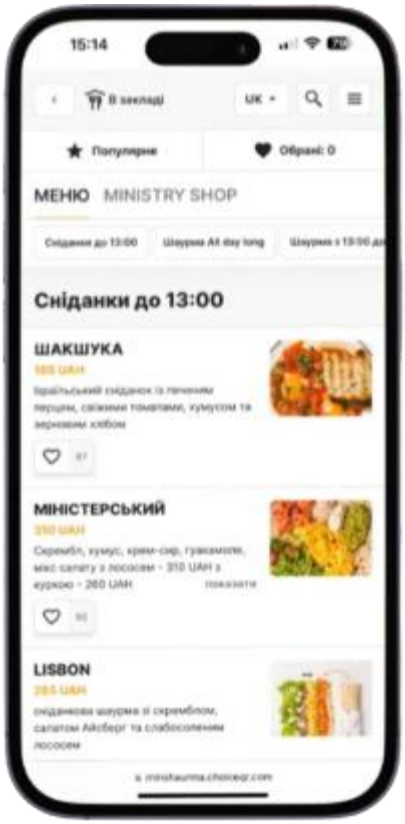


Рисунок 1.10 – Перегляд меню

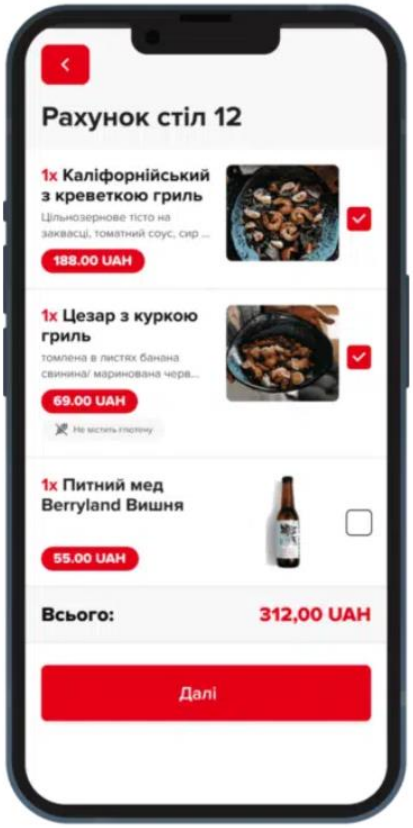


Рисунок 1.11 – Розділення рахунку

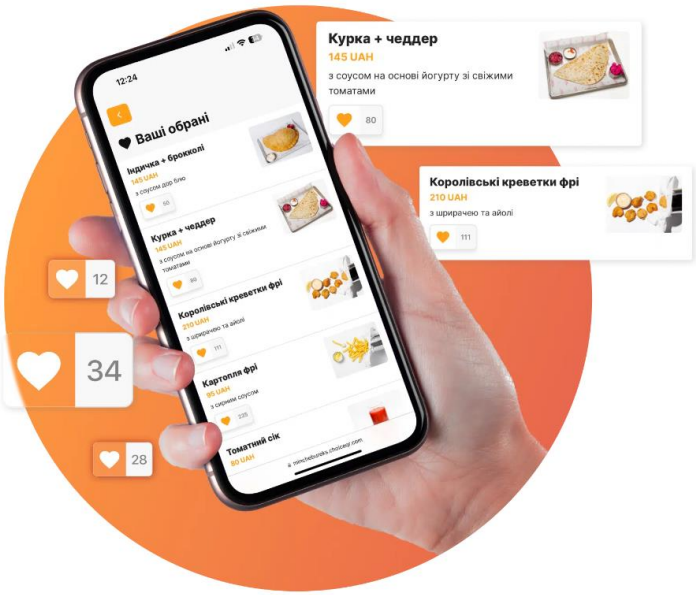


Рисунок 1.12 – Додавання страви до списку бажань

Переваги використання для власників закладу [4]:

– клієнт користується єдиним QR-кодом за столом для меню, відгуків, доставки, оплати. А в POS-системі коректно враховуються всі замовлення. Всі

дані будуть зберігатися в картці клієнта, і ви отримаєте всю історію та деталі в єдиному місці (рисунок 1.13);

– гість сканує QR-код та робить замовлення без офіціанта. Персоналу залишається лише принести замовлення. Таким чином потрібно в 2 рази менше персоналу, щоб обслуговувати всіх клієнтів;

– дозволяє не втратити жодне замовлення, адже надсилається повідомлення про нове замовлення у програмі Choice Business або іншій POS-системі. Це дозволяє оперативно передавати замовлення на кухню та швидко обслуговувати клієнтів;

– після кожного замовлення гість може залишити відгук. Усі відгуки збираються в адмінпанелі, де їх можна легко аналізувати та обробляти. Наприклад, відразу відреагувати на негатив та уникнути поганих відгуків у мережі (рисунок 1.14).

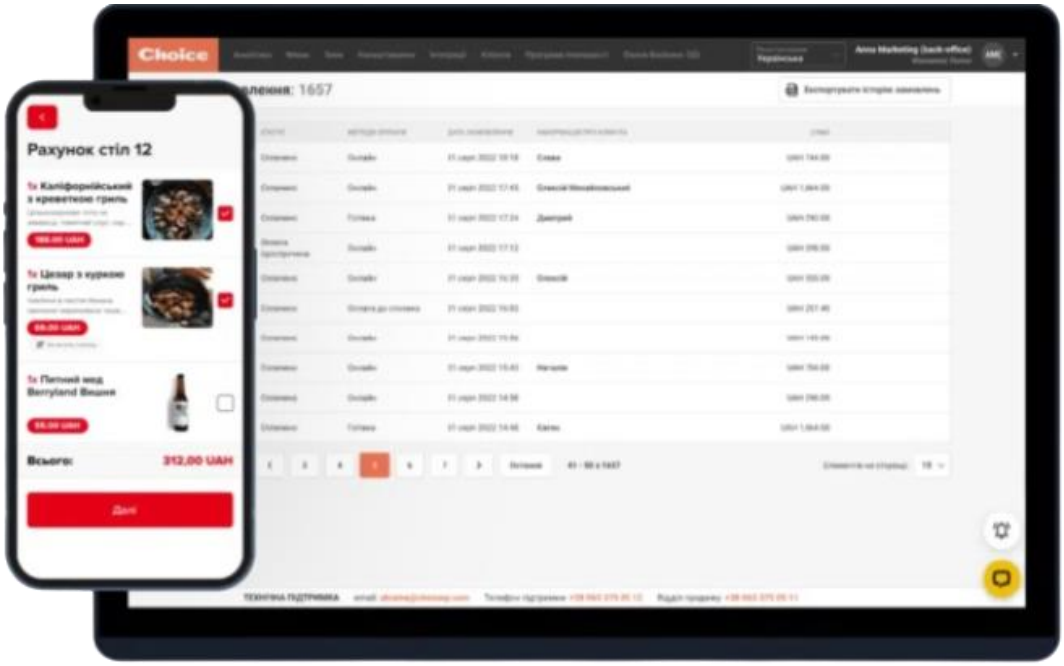


Рисунок 1.13 – Збереження даних про замовлення

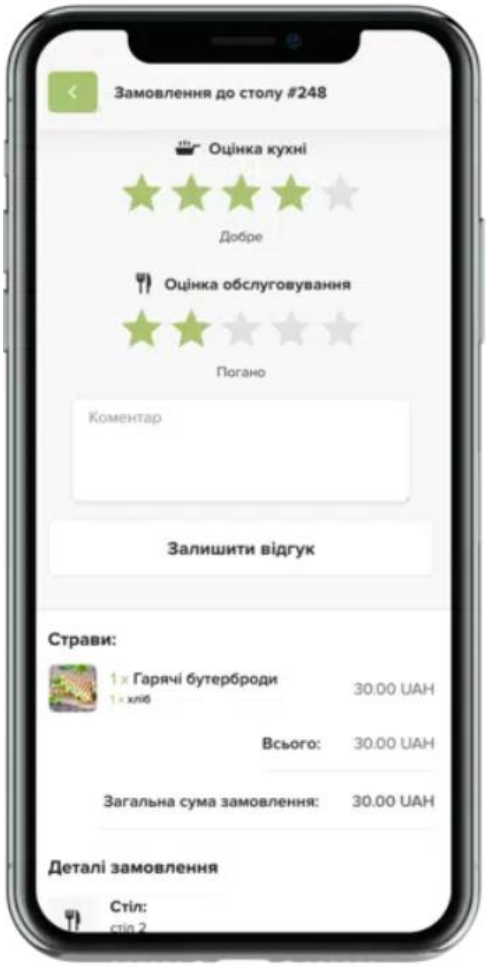


Рисунок 1.14 – Оцінювання кухні й обслуговування

1.5 Вплив соціальних мереж на автоматизацію у галузі харчування

Соціальні мережі відіграють значну роль у сучасному світі комунікації та взаємодії між брендами та споживачами. В останні роки їх вплив на галузь харчування став особливо помітним, викликаючи не лише зміни в маркетингових стратегіях, а й у впровадженні автоматизованих систем у роботу закладів харчування.

Використання соціальних мереж у сфері харчування відкриває широкі можливості для збору даних про споживачів та їх вподобання. Завдяки активності користувачів у соціальних мережах, заклади харчування можуть отримувати цінну інформацію про те, що цікавить їхніх клієнтів, які страви чи сервіси є найбільш популярними, і як можна покращити їхній досвід.

Крім того, соціальні мережі стають ефективним каналом для проведення

рекламних кампаній та акцій, що може значно підвищити популярність закладу. Завдяки цим кампаніям, заклади можуть привернути нових клієнтів, залучити увагу до своїх пропозицій та подій, і відмінно спілкуватися зі своєю аудиторією.

Інтеграція можливостей замовлення через соціальні мережі та месенджери також стає все популярнішою тенденцією. Клієнти можуть легко та зручно робити замовлення прямо через свої обрані платформи, що сприяє збільшенню зручності та швидкості обслуговування.

Взаємодія з клієнтами через коментарі, відгуки та прямі повідомлення у соціальних мережах має великий вплив на удосконалення сервісу та підвищення лояльності. Реакція на відгуки дозволяє закладам надати кращий сервіс, виправити помилки та недоліки, а також позитивно впливати на сприйняття бренду клієнтами.

1.6 Використання чат-ботів у соціальних мережах

Чат-боти у соціальних мережах можуть відігравати ключову роль у підтримці комунікації між закладами харчування та їх клієнтами. Вони забезпечують можливість клієнтам звертатися з запитаннями, резервувати столи, дізнаватися про акції та спеціальні пропозиції, а також швидко отримувати відповіді на будь-які питання щодо меню, годин роботи та інших послуг. Це сприяє підвищенню рівня задоволеності клієнтів та збільшенню їх залученості до закладу харчування.

Більше того, чат-боти у соціальних мережах можуть бути використані для збору важливої інформації про клієнтів, такої як їхні вподобання, історія замовлень та фідбек щодо обслуговування. Ця інформація може бути використана для персоналізації пропозицій та послуг для кожного клієнта, що сприяє покращенню його досвіду відвідування закладу та збільшенню лояльності.

Нарешті, чат-боти у соціальних мережах можуть стати ефективним

інструментом для збору відгуків від клієнтів про якість обслуговування та продукції. Швидка та зручна форма зворотнього зв'язку дозволяє клієнтам висловлювати свої думки та пропозиції, що допомагає закладам харчування виявляти сильні та слабкі сторони своєї роботи і вносити відповідні зміни для поліпшення якості обслуговування.

Гарним прикладом використання чат-ботів є сервіс CoinyPay, який запустили у столиці у закладі «Dogs & Tails». Це чат-бот, який дозволяє оплачувати рахунок у ресторані через Facebook Messenger. CoinyPay працює таким чином. Гостю потрібно знайти CoinyPay в Facebook Messenger і натиснути «Почати». Далі клієнт прив’язує свою банківську карту до чат-боту і таким чином отримує CoinyID — особистий ідентифікаційний код. При виконанні замовлення він говорить CoinyID офіціантові. Рахунок гостю прийде прямо в Messenger, де йому залишиться лише підтвердити транзакцію – оплата відбудеться автоматично через півгодини [5].

2 ТЕХНОЛОГІЧНІ АСПЕКТИ РОЗРОБКИ СИСТЕМИ

2.1 Вимоги до системи

Визначення вимог для майбутньої системи є критичним етапом у процесі розробки будь-якого програмного продукту. Чіткі та конкретні вимоги дозволяють не лише спрямовувати роботу розробників у потрібному напрямку, але й забезпечують високу якість та ефективність системи, що дозволяє забезпечити зручність для клієнтів та оптимізацію робочих процесів для працівників закладу. Тому правильно сформульовані технічні і функціональні вимоги відіграють ключову роль у досягненні успіху проекту та задоволенні потреб його користувачів.

Технічні вимоги для системи обробки замовлень:

– розробка чат-бота: цей етапу включає в себе не лише створення самого бота, але й визначення його функціональності. Це означає визначення списку команд, які буде розуміти бот, структуру даних, що буде використовуватися для обробки замовлень, а також логіку обробки та відповідей на різноманітні запити користувачів;

– серверна частина: підготовка серверної частини включає розгортання сервера та налаштування середовища виконання, такого як встановлення веб-сервера, вибір серверного програмного забезпечення (наприклад, Apache, Nginx), а також встановлення та налаштування необхідних фреймворків та бібліотек для обробки запитів від бота та взаємодії з базою даних;

– інтеграція з базою даних: для забезпечення надійного та ефективного зберігання даних про замовлення необхідно створити структуру бази даних, яка буде включати таблиці для зберігання інформації про клієнтів, замовлення, меню тощо. Цей етап включає в себе вибір та налаштування бази даних (наприклад, MySQL, PostgreSQL).

Функціональні вимоги визначають способи, за допомогою яких користувачі будуть взаємодіяти з системою та як система буде реагувати на їхні

запити.

– система повинна забезпечувати можливість клієнтам легко та зручно робити замовлення через чат-бот. Це включає в себе можливість перегляду меню, вибір страв та напоїв, а також можливість додавання їх до замовлення та зручний процес оплати;

– можливість перевірки статусу замовлення в будь-який момент. Клієнти повинні мати змогу швидко отримувати інформацію про те, на якому етапі знаходиться їхнє замовлення, чи вже прийняте, готується чи вже готово.

– система повинна надавати доступ до інформації про меню, акції та спеціальні пропозиції закладу. Це дозволяє клієнтам бути в курсі останніх новин та отримувати сповіщення про будь-які зміни або нові можливості, що стосуються закладу. Такий підхід сприяє залученню клієнтів та підвищенню їхньої лояльності до закладу харчування.

2.2 Вибір мови програмування

2.2.1 Основні відомості про мову

Java — це одна з найпопулярніших мов програмування. Її розробила компанія Sun Microsystems під керівництвом Джеймса Гослінга в 1995 році. Гослінг створив інструмент, який дає змогу розробникам писати код один раз і запускати його на будь-якій платформі без необхідності перекомпіляції (принцип WORA – Write Once and Run Anywhere).

Java — це об’єктно-орієнтована мова програмування загального призначення з простим і зрозумілим синтаксисом, яка підходить для різних платформ. Найчастіше нею пишуть Backend (серверну частину софту).

Завдяки широким можливостям, бібліотекам і портативності Java дозволяє створювати ПЗ для різних компаній і сфер: ігри, мобільні застосунки, корпоративні рішення тощо [6].

Ключові особливості Java:

– жорстка типізація. У Java всі змінні мають бути оголошені із зазначенням їхнього типу. Це запобігає помилкам типізації, дає змогу писати зрозумілий код і знаходити баги на етапі компіляції, а не виконання;

– автоматичне керування пам’яттю. У мові програмування Java реалізовано автоматичне збирання сміття. Вона звільняє пам’ять, зайняту об’єктами, що більше не використовуються. Розробникам не потрібно робити це власноруч;

– об’єктно-орієнтованість. Java ґрунтується на концепції об’єктно-орієнтованого програмування. Це означає, що все в Java є об’єктом, який має свої властивості та методи. Об’єктно-орієнтований підхід дає можливість Java-розробникам створювати модульні, гнучкі та безпечні застосунки [6].

2.2.2 Переваги та недоліки мови Java

У Java багато плюсів. Нею пишуть ПЗ у Netflix, Spotify, Google, LinkedIn та інших великих компаніях. Ось основні переваги, на які варто звернути увагу:

– простота використання та освоєння Java, зрозумілий синтаксис і семантика роблять її привабливою для початківців;

– портативність: код, написаний на Java, працює на будь-якій ОС, що підтримує JVM, спрощуючи розробку та розгортання;

– велика бібліотека: стандартна бібліотека Java містить багато класів і методів для різних завдань, що полегшує програмування;

– кібербезпека: завдяки перевірці типів та іншим механізмам безпеки, Java дозволяє розробляти безпечні застосунки;

– багатопоточність і масштабованість: вбудована підтримка багатопоточності дозволяє ефективно опрацьовувати великі обсяги даних та масштабувати програми;

– продуктивність: Java забезпечує гарну продуктивність завдяки своїй віртуальній машині та оптимізації коду;

– підтримка та активна спільнота: велика спільнота розробників забезпечує підтримку та легкий доступ до матеріалів для вивчення;

– обробка винятків: вбудовані механізми дозволяють ефективно керувати помилками та винятковими ситуаціями;

– застосування в різних сферах: Java використовується у мобільних застосунках, вебсервісах, іграх, фінансових та наукових системах тощо;

– рефлексія: можливість програмного аналізу та модифікації структури програми під час її виконання.

Хоча Java має свої переваги, варто також враховувати деякі недоліки:

– швидкість: у порівнянні з іншими мовами, такими як C++, Java може виконувати деякі завдання повільніше, що може бути проблемою для застосунків, що потребують високої продуктивності або низької затримки;

– споживання пам'яті: використання Java вимагає більше пам'яті для програм, через механізм автоматичного керування пам'яттю та додаткові структури даних у JVM;

– відсутність нативного доступу до низькорівневих функцій: деякі типи застосунків можуть бути обмежені через відсутність прямого доступу до операційної системи. Хоча в Java існують механізми для роботи з операційною системою через Java Native Interface (JNI), це вимагає додаткової роботи та знань.

2.2.3 Області застосування Java

На Java розробляють різноманітне програмне забезпечення, від мобільних додатків до наукових і промислових застосунків. Ось деякі основні області, де використовується ця мова:

– веб-застосунки: Java надає потужні інструменти для створення веб-застосунків і серверних компонентів, і безліч сайтів і додатків розроблено саме на цій мові;

– мобільна розробка: Java є однією з основних мов програмування для

платформи Android, що дозволяє розробникам створювати різноманітні додатки для цієї платформи;

– корпоративне ПЗ: багато великих корпорацій використовують Java для створення масштабованих і надійних систем, а також застосунків у хмарі;

– ігри: Java дозволяє розробникам створювати ігри за допомогою різноманітних бібліотек, таких як Lightweight Java Game Library і OpenJFX;

– обробка великих даних: Java використовується в аналітиці та машинному навчанні через бібліотеки, такі як Apache Spark і Weka;

– інтернет речей (IoT): Java використовується для програмування апаратного забезпечення і датчиків периферійних пристроїв, що підключаються до Інтернету;

– наукові та дослідницькі програми: використовується для наукових обчислень, створення математичних моделей, симуляцій та аналізу даних;

– розробка додатків доповненої реальності (AR) і віртуальної реальності (VR): Java використовується для створення AR/VR-застосунків через спеціалізовані фреймворки та інструменти.

Це далеко не весь список того, що можна писати на Java. Цю мову використовують для автоматизації задач, управління обладнанням, створення спецефектів, космічних розробок і багато чого іншого.

2.2.4 Використання мови Java для розробки Telegram ботів

Java — це об'єктно-орієнтована мова програмування загального призначення, яка здобула широку популярність завдяки своїй надійності, кросплатформенності та простоті використання. Вона часто використовується для розробки різноманітних програм та сервісів, включаючи і чат-ботів для платформи Telegram.

Для розробки телеграм ботів на мові Java використовується Telegram Bot API (Application Programming Interface). Це набір інструментів, який надає Telegram для розробників, щоб створювати та взаємодіяти з ботами. Telegram

Bot API дозволяє створювати ботів, які можуть відправляти та отримувати повідомлення, взаємодіяти з користувачами та виконувати різні завдання.

Один з найпопулярніших фреймворків для розробки телеграм ботів на Java — це TelegramBots. Цей фреймворк надає зручний інтерфейс для роботи з Telegram Bot API, спрощуючи процес створення та налаштування бота. Він забезпечує різні можливості, такі як обробка вхідних повідомлень, взаємодія з клавіатурою, робота з файлами та багато іншого.

Ще однією важливою складовою розробки телеграм ботів на Java є робота з базами даних для зберігання і обробки інформації. Для цього можна використовувати різні технології, такі як JDBC (Java Database Connectivity), JPA (Java Persistence API) або різноманітні ORM (Object-Relational Mapping) фреймворки, які спрощують взаємодію з базою даних.

Завдяки Java та Telegram Bot API розробники можуть легко створювати потужні та ефективні телеграм боти, які відповідають на потреби користувачів та виконують різноманітні завдання з високою надійністю.

2.3 Апаратне забезпечення для системи замовлення

Розроблена система автоматизованого замовлення у закладі харчування має на меті спростити процес обробки та виконання замовлень для забезпечення швидкого та ефективного обслуговування клієнтів. Після того як клієнт робить замовлення через Телеграм бота, інформація автоматично надходить на сервер, де вона обробляється та зберігається у базі даних. Потім співробітники кухні отримують доступ до замовлень через мікроконтролер з дисплеєм, де вони можуть переглядати деталі замовлень та оновлювати їхній статус. Після виконання замовлення статус автоматично оновлюється, що повідомляється клієнту через Телеграм бота.

2.3.1 Вибір мікроконролера

Система автоматизації замовлень у сфері харчування потребує мікроконтролера, який забезпечить надійну і швидку обробку даних, ефективну роботу з бездротовими мережами, низьке споживання енергії, а також можливість взаємодії з різними пристроями. Однією з ключових вимог є підтримка бездротового зв'язку Wi-Fi для отримання замовлень і передачі даних між клієнтами, сервером і кухонним персоналом. Також важливо мати можливість взаємодіяти з Bluetooth для підключення до різних пристроїв, наприклад, дисплея для відображення замовлень та статусів. Мікроконтролер повинен бути достатньо потужним для обробки інформації швидко та ефективно, а також мати достатню кількість входів/виходів для підключення різноманітних датчиків та інших пристроїв.

ESP-32 — це мікроконтролер китайського виробника Espressif з двоядерним 32-розрядним процесором Tensilica Xtensa LX6 та 520 Кб пам'яті SRAM. Контролер має тактову частоту до 240 МГц залежно від режиму споживання енергії. Подивитись зовнішній вигляд мікроконтролера можна на рисунку 2.1.

Мікроконтролери цієї серії широко використовуються в проєктах розумного будинку, промислової автоматики, побутовій електроніці, робототехніці, камерах для потокового відео, розпізнаванні мови й зображень та ін [7].

Цей мікроконтролер відповідає зазначеним вище необхідним вимогам, оскільки він має вбудовані модулі Wi-Fi та Bluetooth, низьке споживання енергії, потужну архітектуру з двома ядрами, а також багато входів/виходів для підключення до різних пристроїв. Він надійно працює у різних умовах і володіє широким функціоналом, що робить його ідеальним вибором для реалізації системи обробки замовлень.

Рисунок 2.1 – Мікроконтролер ESP-32

2.3.2 Вибір дисплею

Співробітники кухні мають можливість отримувати доступ до замовлень через спеціальний мікроконтролер з дисплеєм. На цьому дисплеї вони можуть переглядати деталі кожного замовлення, включаючи вміст та інформацію про статус. Вони можуть легко оновлювати статус кожного замовлення, натискуючи відповідні кнопки на дисплеї та обираючи один із доступних статусів, таких як "Готується" або "Готово".

Після виконання замовлення, статус автоматично оновлюється, що дозволяє клієнту отримати повідомлення через Телеграм бота про те, що його замовлення готове до видачі. Такий процес дозволяє забезпечити найкращу якість обслуговування та забезпечити позитивний досвід відвідування для клієнтів вашого закладу харчування.

Для успішної реалізації системи автоматизованого замовлення у закладі харчування необхідно мати дисплей, який забезпечить співробітникам кухні зручний доступ до детальної інформації про замовлення. Основні вимоги до дисплею включають великий розмір та високу роздільну здатність для зручного відображення тексту та графічних елементів. Крім того, дисплей повинен бути легко читабельним навіть в умовах поганого освітлення, щоб забезпечити

комфортну роботу співробітників кухні протягом усього робочого дня.

Розглядаючи різноманітні варіанти дисплеїв, одним з оптимальних виборів є використання міні-комп'ютера Raspberry Pi з підключеним до нього дисплеєм. Raspberry Pi відомий своєю надійністю, простотою в налаштуванні та гнучкістю в застосуванні. Однією з переваг Raspberry Pi є його можливість виводити графічний інтерфейс на підключений дисплей, що дозволяє співробітникам кухні легко переглядати інформацію про замовлення та оновлювати їх статус.

Окрім того, Raspberry Pi підтримує широкий вибір дисплеїв різних розмірів та характеристик, що дозволяє вибрати оптимальний варіант для конкретних потреб проекту. Враховуючи те, що використовується мікроконтролер ESP-32, використання Raspberry Pi як дисплею виявиться ідеальним варіантом для забезпечення зручного та ефективного взаємодії замовлень на кухні закладу харчування.

7 дюймовий сенсорний монітор від Raspberry Pi, зображений на рисунку 2.2, надає користувачу можливість створювати проекти "все-в-одному", наприклад, планшети, інформаційно-розважальні системи або вбудовані проекти. Дисплей приєднується через плату-адаптер, який регулює живлення і перетворює сигнал. Ознайомитись з характеристиками можна у таблиці 2.1 [8].

Рисунок 2.2 – Дисплей Raspberry Pi 7" Touch Screen Display

Таблиця 2.1 – Характеристики дисплею

Назва	Дисплей Raspberry Pi 7" Touch Screen Display
Тип дисплею	TFT/IPS
Підключеня	DSI
Touch Screen	Capacitive
Розмір в дюймах	7.0 inch
Роздільна здатність	800 x 480
Сумістність	Raspberry Pi
Країна виробника	Великобританія

Для поєднання сенсорного дисплею Raspberry Pi 7" Touch Screen Display з мікроконтролером ESP-32 можна використовувати будь-яку модель Raspberry Pi, яка має роз'єми GPIO для підключення та підтримує роботу з сенсорним дисплеєм. Однак, однією з найпопулярніших і підходящих моделей для цього є Raspberry Pi 3 Model B+ або Raspberry Pi 4 Model B.

Якщо вибирати між ними, Raspberry Pi 4 Model B може бути кращим варіантом, оскільки він має більш потужний процесор, більше оперативної пам'яті та покращені можливості підключення. Більш детально ознайомитись з характеристиками мікроконтрелера можна у таблиці 2.2.

Одноплатний мікрокомп'ютер Raspberry Pi 4 Model B, зображений на рисунку 2.3 — це новий продукт в популярному ряді мікрокомп'ютерів Raspberry Pi [9].

Рисунок 2.3 – Мікрокомп'ютер Raspberry Pi 4 Model B 8GB

Таблиця 2.2 – Характеристики мікроконтролеру

Назва	Мікрокомп'ютер Raspberry Pi 4 Model B 8GB
Форм-фактор	Raspberry Pi 4
Процесор	Cortex-A72 @ 1.5GHz
Кількість ядер	quad-core
Об'єм ОЗП	8GB
Об'єм ПЗП	Micro SD card
Bluetooth	Bluetooth 5.0
Ethernet	10/100/1000 Ethernet
HDMI	2 × micro-HDMI
USB	2 x USB 3.0 / 2 x USB 2.0
Wi-Fi	2.4 GHz / 5.0 GHz
Країна виробника	Великобританія

2.4 Вибір бази даних

2.4.1 Вимоги до бази даних

У процесі розробки системи автоматизованого замовлення для закладу харчування важливим аспектом є вибір відповідної бази даних. База даних відіграє ключову роль у зберіганні, обробці та доступі до інформації про замовлення, клієнтів та інші важливі дані, необхідні для ефективного функціонування системи. У цьому проекті було обрано MySQL як основну базу даних, і цей вибір обґрунтований низкою переваг та відповідності вимогам проекту.

База даних повинна бути надійною, забезпечувати збереження даних без втрат і помилок та стабільно працювати під навантаженням. Також система повинна мати можливість розширення та адаптації до зростаючих обсягів даних та збільшення кількості користувачів.

Не менш важливим є безпека, адже дані про клієнтів та замовлення повинні бути захищені від несанкціонованого доступу та втрат. Варто також звернути увагу на те, що база даних повинна легко інтегруватися з іншими компонентами системи, такими як серверна частина на Spring Boot та клієнтський інтерфейс на основі Telegram Bot API.

2.4.2 Переваги MySQL

MySQL було обрано як базу даних для цього проєкту завдяки ряду переваг:

– забезпечує швидку обробку запитів, що дозволяє системі оперативно обробляти замовлення та надавати клієнтам актуальну інформацію про їх статус;

– підтримує вертикальне та горизонтальне масштабування, що дозволяє легко розширювати систему у разі збільшення обсягу даних або навантаження;

– відома своєю стабільністю та надійністю у промислових умовах, що забезпечує збереження даних без втрат;

– пропонує розширені можливості для захисту даних, включаючи управління доступом користувачів та шифрування даних;

– легко інтегрується з іншими технологіями, які використовуються в проекті, такими як Java, Spring Boot, та інші інструменти для розробки веб-додатків;

– має велику спільноту користувачів та розробників, а також доступну документацію та технічну підтримку, що полегшує вирішення будь-яких проблем та питань, які можуть виникнути в процесі розробки та експлуатації системи.

Враховуючи всі ці фактори, MySQL є оптимальним вибором для створення надійної, продуктивної та масштабованої системи автоматизованого замовлення для закладу харчування.

3 РОЗРОБКА ТА РЕАЛІЗАЦІЯ СИСТЕМИ

3.1 Написання Телеграм боту

3.1.1 Структура проєкту

Проєкт складається з кількох основних пакетів і класів, що організовані в певну ієрархію для забезпечення функціональності Телеграм бота для автоматизації обробки замовлень у закладі харчування. Зі складовими проєкту можна ознайомитись на рисунку 3.1.

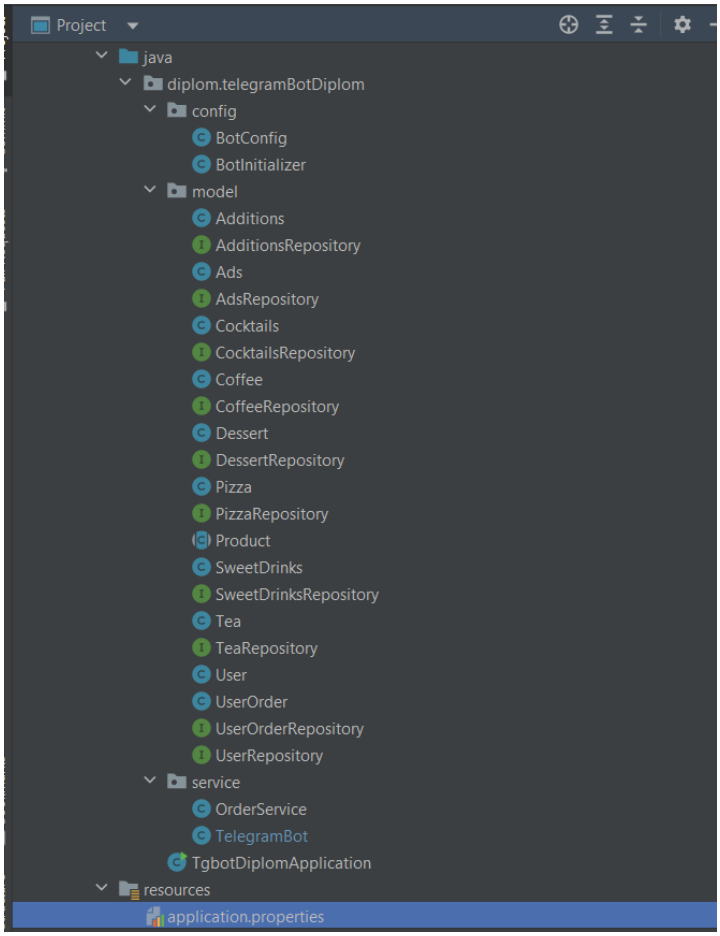


Рисунок 3.1 – Структура проєкту

Основні компоненти проєкту:

Пакет diplom.telegramBotDiplom — це основний пакет проєкту, який містить кілька підпакетів для різних аспектів реалізації:

– пакет config — цей пакет містить конфігураційні класи для налаштування Телеграм бота, а саме: BotConfig (клас для налаштування конфігурації бота, наприклад, токену доступу та інших параметрів) і BotInitializer (клас для ініціалізації та запуску бота);

– пакет model містить класи, що представляють модель даних, і відповідні репозиторії для доступу до бази даних: Additions (клас, що представляє додаткові елементи меню,наприклад, соуси), AdditionsRepository (репозиторій для доступу до даних про додаткові елементи), Ads (клас, що представляє рекламні оголошення), AdsRepository (репозиторій для доступу до даних про рекламні оголошення), Cocktails (клас, що представляє коктейлі), CocktailsRepository (репозиторій для доступу до даних про коктейлі), Coffee (клас, що представляє кавові напої), CoffeeRepository (репозиторій для доступу до даних про кавові напої), Dessert (клас, що представляє десерти), DessertRepository (репозиторій для доступу до даних про десерти), Pizza (клас, що представляє піци), PizzaRepository (репозиторій для доступу до даних про піци), Product (базовий клас для різних продуктів), SweetDrinks (клас, що представляє солодкі напої), SweetDrinksRepository (репозиторій для доступу до даних про солодкі напої), Tea (клас, що представляє чайні напої), TeaRepository (репозиторій для доступу до даних про чайні напої), User (клас, що представляє користувача), UserOrder (клас, що представляє замовлення користувача), UserOrderRepository (репозиторій для доступу до даних про замовлення користувачів), UserRepository (репозиторій для доступу до даних про користувачів);

– пакет service містить сервіси, що забезпечують основну бізнес-логіку додатка. А саме це: OrderService (сервіс для обробки замовлень) і TelegramBot (клас, що реалізує основну логіку взаємодії з Телеграм ботом).

Кореневий клас TgbotDiplomApplication — це головний клас, що запускає додаток, використовуючи Spring Boot.

3.1.2 Підключення залежностей і фреймворків

Для створення ефективного і надійного програмного забезпечення важливо правильно організувати процес керування проектом і його збірки. У цьому контексті одним із найбільш популярних і потужних інструментів є Apache Maven. Завдяки своїй здатності автоматизувати та стандартизувати різні аспекти розробки, Maven став незамінним інструментом для багатьох розробників і команд.

Maven – інструмент для управління та збирання проектів. Він полегшує життя розробнику на всіх стадіях роботи: від створення структури проекту та підключення необхідних бібліотек до розгортання продукту на сервері. Maven забезпечує стандартизовану структуру проекту, спрощуючи процеси компіляції, тестування, створення пакетів і розгортання програмного забезпечення [10].

Для організації залежностей і керування збіркою проєкту використовується файл pom.xml, що належить до системи керування збірками Apache Maven. Цей файл визначає всі необхідні залежності, плагіни та властивості для проєкту, дозволяючи автоматизувати процеси компіляції, тестування і розгортання. Нижче наведено частини файлу pom.xml.

Перша і обов'язкова складова - заголовок XML і визначення проекту, з нею можна ознайомитись на рисунку 3.2. Цей блок коду є початковою частиною файлу pom.xml, який використовується для налаштування проекту в системі керування збірками Apache Maven.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3           xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
4      <modelVersion>4.0.0</modelVersion>
```

Рисунок 3.2 – Заголовок XML і визначення проекту

Ці рядки відповідають за визначення версії XML та кодування символів, встановлення кореневого елемента для POM-файлу Maven, визначення

простору імен для моделі об'єкта проекту (POM), дозволяє використання атрибутів XML Schema в документі, та вказує на місце розташування XML-схеми для перевірки структури та вмісту POM-файлу.

Наступний, не менш важливий блок коду, зображений на рисунку 3.3, відповідає за визначення основних характеристик і конфігурацій проекту. Він встановлює тип упаковки для проекту як WAR, що використовується для веб-додатків. Також, він визначає батьківський проект spring-boot-starter-parent, від якого успадковуються налаштування, включаючи груповий ідентифікатор, артефактний ідентифікатор та версію батьківського проекту. Він задає груповий ідентифікатор, артефактний ідентифікатор, версію, назву та опис для поточного проекту, що допомагає ідентифікувати проект і його версію в системі управління збірками.

```
5       <packaging>war</packaging>
6       <parent>
7           <groupId>org.springframework.boot</groupId>
8           <artifactId>spring-boot-starter-parent</artifactId>
9           <version>2.7.1</version>
10          <relativePath/>
11      </parent>
12      <groupId>io.proj3ct</groupId>
13      <artifactId>TelegramBotDiplom</artifactId>
14      <version>0.0.1-SNAPSHOT</version>
15      <name>tg_bot_diplom</name>
16      <description>tg_bot_diplom</description>
```

Рисунок 3.3 – Визначення основних характеристик і конфігурацій проекту

Основним призначенням файлу pom.xml є організація залежностей. Наступний блок коду, з яким можна ознайомитись на рисунку 3.4, відповідає за визначення властивостей проекту, залежностей і плагінів. У розділі властивостей задаються версії Java і бібліотеки Telegram. У розділі залежностей перераховуються всі необхідні для проекту бібліотеки, із зазначенням відповідних версій та інших параметрів:

– telegrambots (бібліотека для роботи з Telegram Bot API);
– spring-boot-starter (основний стартер для додатків Spring Boot);
– spring-boot-starter-test (інструменти для тестування додатків Spring

Boot);

– lombok (генерація коду, зокрема геттери, сеттери, конструктори);

– spring-boot-starter-data-jpa (підтримка JPA для роботи з базами даних);

– mysql-connector-j (JDBC-драйвер для підключення до бази даних MySQL);

– emoji-java (підтримка емодзі в Java-додатках);

– javax.persistence-api (стандартні API для JPA).

У розділі збірки визначається плагін spring-boot-maven-plugin, який забезпечує інтеграцію з Spring Boot і налаштування його виконуваності.

```
<dependencies>

    <dependency>
        <groupId>org.telegram</groupId>
        <artifactId>telegrambots</artifactId>
        <version>${telegram.version}</version>
    </dependency>

    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter</artifactId>
    </dependency>

    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>

    <dependency>
        <groupId>org.projectlombok</groupId>
        <artifactId>lombok</artifactId>
        <optional>true</optional>
    </dependency>
```

Рисунок 3.4 – Підключення залежностей

3.1.3 Налаштування конфігурації

Ефективна конфігурація відіграє критичну роль у розробці програмного забезпечення, особливо у великих або складних проектах. Клас BotConfig, зображений на рисунку 3.5, є центральним елементом управління налаштуваннями бота для додатку. Цей клас дозволяє централізовано зберігати та управляти конфігураційними даними, такими як ім'я бота, його токен та

ідентифікатор власника. Він також забезпечує простий та зручний доступ до цих даних у всьому додатку через використання анотацій Spring та Lombok. Це дозволяє легко налаштовувати та керувати параметрами бота без необхідності редагування вихідного коду. @PropertySource("application.properties") - ця анотація вказує на файл з налаштуваннями додатку. У цьому випадку, властивості бота (ім'я, токен, власник) зберігаються в файлі application.properties.

```java
package diplom.telegramBotDiplom.config;

import lombok.Data;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.PropertySource;
import org.springframework.scheduling.annotation.EnableScheduling;

@Configuration 3 usages  VikaKhaustova
@EnableScheduling
@Data

@PropertySource("application.properties")
public class BotConfig {

    @Value("${bot.name}")
    String botName;

    @Value("${bot.token}")
    String token;

    @Value("${bot.owner}")
    Long ownerId;
}
```

Рисунок 3.5 – Клас BotConfig

Файл application.properties використовується для налаштування конфігурації Spring Boot-додатка. Він містить різноманітні властивості, які визначають поведінку додатка, підключення до бази даних, параметри серверу та інші важливі налаштування. У даному випадку файл application.properties, зображений на рисунку 3.6 включає наступні налаштування:

– server.port=${PORT:8080}: вказує порт, на якому запускається сервер. Якщо змінна PORT не визначена, використовується порт 8080;

– bot.name, bot.token, bot.owner: визначають ім'я бота, токен доступу та ID власника бота в Telegram відповідно.

Налаштування БД:

– spring.jpa.hibernate.ddl-auto=update: вказує Hibernate автоматично оновлювати схему БД при запуску додатка;

– spring.datasource.url=jdbc:mysql://localhost:3306/tg_bot: вказує URL для підключення до БД MySQL;

– spring.datasource.username=root: вказує ім'я користувача для підключення до БД;

– spring.datasource.password=****: вказує пароль для підключення БД;

– spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver: вказує драйвер JDBC для MySQL;

– spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect: вказує діалект Hibernate для MySQL 8;

– spring.jpa.show-sql=true: вказує, що SQL-запити повинні бути виведені в консоль для налагодження.

```
server.port=${PORT:8080}

bot.name=vikaPizzeriaBot
bot.token=6524074864:AAHzhJ30xUe03mdessPbkeR8BgRfGBC2bP0
bot.owner=347016249

#db related settings
spring.jpa.hibernate.ddl-auto=update
spring.datasource.url=jdbc:mysql://localhost:3306/tg_bot
spring.datasource.username=root
spring.datasource.password=******
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect
spring.jpa.show-sql=true
```

Рисунок 3.6 – Налаштування конфігурації

3.1.4 Опис класів, що являють собою сутності страв і напоїв

Пакет «model» містить класи, що представляють модель даних, і відповідні репозиторії для доступу до БД.

Клас Product, наведений на рисунку 3.7, є абстрактним класом, який слугує базовою моделлю для різних продуктів. Він визначає загальні властивості, які є спільними для всіх продуктів, і використовується для створення конкретних підкласів (наприклад, Dessert, Tea, Coffee тощо), що представляють конкретні типи продуктів. Ось основні деталі класу Product:

– анотація @Data від Lombok генерує стандартні методи, такі як гетери, сетери, toString, equals і hashCode, що значно скорочує обсяг коду;

– анотація @Entity вказує, що цей клас є сутністю JPA і буде відображатися в базі даних;

– анотація @Inheritance(strategy = InheritanceType.TABLE_PER_CLASS) визначає стратегію наслідування, за якою кожен підклас зберігається в окремій таблиці бази даних;

– поле id, анотоване як @Id, вказує на те, що це поле є ідентифікатором сутності;

– анотація @GeneratedValue(strategy = GenerationType.AUTO) задає стратегію автоматичного генерування значень для поля id;

– поля name, price і quantity визначають основні властивості продукту: назва, ціна і кількість.

Клас Product виступає базовою сутністю, що дозволяє забезпечити уніфіковану структуру для всіх продуктів у додатку. Це спрощує роботу з різними типами продуктів, забезпечуючи загальні поля та поведінку, які можуть бути розширені в конкретних підкласах.

```java
package diplom.telegramBotDiplom.model;

import lombok.Data;

import javax.persistence.*;
@Data  7 inheritors  VikaKhaustova *
@Entity
@Inheritance(strategy = InheritanceType.TABLE_PER_CLASS)
public abstract class Product {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int id;
    private String name;
    private double price;
    private int quantity;
}
```

Рисунок 3.7 – Клас Product

Всі класи продуктів наслідують клас Product і описані за аналогією з класом Dessert, з яким можна ознайомитись на рисунку 3.8.

```java
package diplom.telegramBotDiplom.model;
import lombok.Data;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
@Data  24 usages  VikaKhaustova *
@Entity(name = "dessertsTable")
public class Dessert  extends  Product{
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int id;
    private String name;
    private String description;
    private int weight;
    private String imageUrl;
    private double price;

    public Dessert() {  1 usage  VikaKhaustova
    }

    public Dessert(int id, String name,  String description,  int weight, String imageUrl, double price) {
        this.id = id;
        this.description = description;
        this.name = name;
        this.weight = weight;
        this.imageUrl = imageUrl;
        this.price = price;
    }
}
```

Рисунок 3.8 – Клас Dessert

Клас Dessert є прикладом сутності, яка представляє десерти в БД. Аналогічно цьому класу створені всі інші класи, що представляють різні продукти, такі як піццу, доповнення, кавові напої, чаї, коктейлі, солодкі напої

тощо. Ці класи містять відповідні поля для зберігання інформації про конкретні продукти і використовують анотації для інтеграції з JPA.

Анотація @Data від Lombok генерує стандартні методи, такі як гетери, сетери, toString, equals і hashCode, спрощуючи код і позбавляючи необхідності вручну писати ці методи. Це допомагає зменшити обсяг коду та підвищити його читабельність.

Анотація @Entity вказує, що клас є сутністю JPA і буде відображатися в БД як таблиця. @Id визначає поле id як ідентифікатор сутності, а @GeneratedValue(strategy = GenerationType.AUTO) задає стратегію автоматичної генерації значень для цього поля.

Клас Dessert представляє десерти з властивостями id, name, description, weight, imageUrl та price, тобто ідентифікаційний номер в таблиці БД, назва десерту, опис, вага, посилання на зображення і ціну, відповідно. Він також містить конструктор без параметрів і конструктор з параметрами для створення об'єктів із заданими значеннями.

3.1.5 Опис інтерфейсів, що забезпечують доступ до БД

Клас DessertRepository, зображений на рисунку 3.9, і аналогічні йому репозиторії є інтерфейсами, що забезпечують доступ до БД для відповідних сутностей. Вони розширюють інтерфейс JpaRepository від Spring Data JPA, який надає набір стандартних CRUD-операцій (Create, Read, Update, Delete) для роботи з базою даних.

```
1  package diplom.telegramBotDiplom.model;
2
3  import org.springframework.data.jpa.repository.JpaRepository;
4  import org.springframework.stereotype.Repository;
5
6  @Repository  no usages  VikaKhaustova
7  public interface DessertRepository extends JpaRepository<Dessert, Integer> {
8
9  }
```

Рисунок 3.9 – Інтерфейс DessertRepository

Анотація @Repository вказує, що цей інтерфейс є репозиторієм і слугує для виконання операцій з базою даних. Вона також допомагає Spring розпізнати цей клас як компонент, що має бути зареєстрований у контексті додатка для забезпечення належної роботи залежностей.

Інтерфейс JpaRepository приймає два параметри: тип сутності (у даному випадку, Dessert) і тип ідентифікатора сутності (у даному випадку, Integer). Це забезпечує автоматичне створення основних методів для роботи з БД, таких як збереження, оновлення, видалення та пошук об'єктів.

Подібні класи-репозиторії, такі як AdditionsRepository, TeaRepository, CocktailsRepository, CoffeeRepository, PizzaRepository, SweetDrinksReposotory тощо, забезпечують аналогічний доступ до бази даних для своїх відповідних сутностей. Вони допомагають ізоляції логіки доступу до даних від бізнес-логіки, роблячи код більш структурованим і підтримуваним.

3.1.6 Опис класів, що відповідають за дані користувача

У проєкті було необхідним розробити можливість забезпечити основну функціональність для управління користувачами та їхніми замовленнями в системі, дозволяючи зберігати, витягувати і обробляти відповідну інформацію в базі даних. Тому необхідно було розробити ряд класів і інтерфейсів для вирішення поставленої задачі. До них належать: клас User, який представляє користувача; інтерфейс UserRepository для доступу до даних про користувачів; клас UserOrder, який представляє замовлення користувача; і інтерфейс UserOrderRepository для доступу до даних про замовлення користувачів.

Клас User представляє користувача системи з усіма необхідними атрибутами, такими як ідентифікатор чату, ім'я, прізвище, ім'я користувача, час реєстрації та стан підписки на новини. Цей клас зберігається в БД як сутність JPA, що дозволяє легко зберігати і витягувати інформацію про користувачів.

Інтерфейс UserRepository розширює CrudRepository, надаючи основні

операції для роботи з користувачами, такі як створення, читання, оновлення та видалення. Це спрощує доступ до бази даних і управління записами користувачів без необхідності писати SQL — запити вручну, завдяки чому розробка стає швидшою та ефективнішою.

Клас UserOrder і відповідний інтерфейс UserOrderRepository служать для управління замовленнями користувачів. UserOrder містить список продуктів, які користувач додав до замовлення, та методи для додавання продуктів і отримання списку замовлених продуктів. Інтерфейс UserOrderRepository також розширює CrudRepository, забезпечуючи базові CRUD-операції для замовлень користувачів, що дозволяє зберігати, витягувати та маніпулювати замовленнями в БД.

3.1.7 Опис класу, що відповідає за замовлення

Клас OrderService відіграє ключову роль у керуванні замовленнями користувачів у Телеграм боті. Його загальне призначення полягає у наданні функціоналу для додавання, оновлення та очищення замовлень користувачів. Також у класі реалізовано можливість оновлення статусу замовлення і надсилання повідомлення користувачу. Він використовується для обробки даних про продукти, які додаються в кошик користувача, та зберігає інформацію про замовлення і його статус у вигляді об'єктів класу UserOrder. Ознайомитись з цим класом можна на рисунку 3.10.

Клас надає методи для додавання або оновлення продуктів у замовлення користувача (addOrUpdateProductInOrder()), а також для очищення його кошика (clearUserCart()). Використовуючи колекцію Map, він зберігає зв'язок між ідентифікаторами користувачів та їхніми замовленнями.

Оновлення статусу замовлення реалізується через метод updateOrderStatus(), який дозволяє змінювати статус конкретного продукту в замовленні користувача. Цей метод перевіряє наявність замовлення користувача і продукту в ньому, після чого оновлює статус продукту та логує

відповідні зміни. Це забезпечує гнучкість у керуванні статусами замовлень, дозволяючи відстежувати процес виконання замовлень у режимі реального часу.

Крім того, використання анотацій @Component та @Slf4j дозволяє Spring автоматично керувати цим компонентом і надає можливість логувати події, пов'язані з обробкою замовлень. Таким чином, OrderService відповідає за ефективне управління замовленнями користувачів у додатку, що робить його невід'ємною складовою системи обробки замовлень у телеграм-боті.

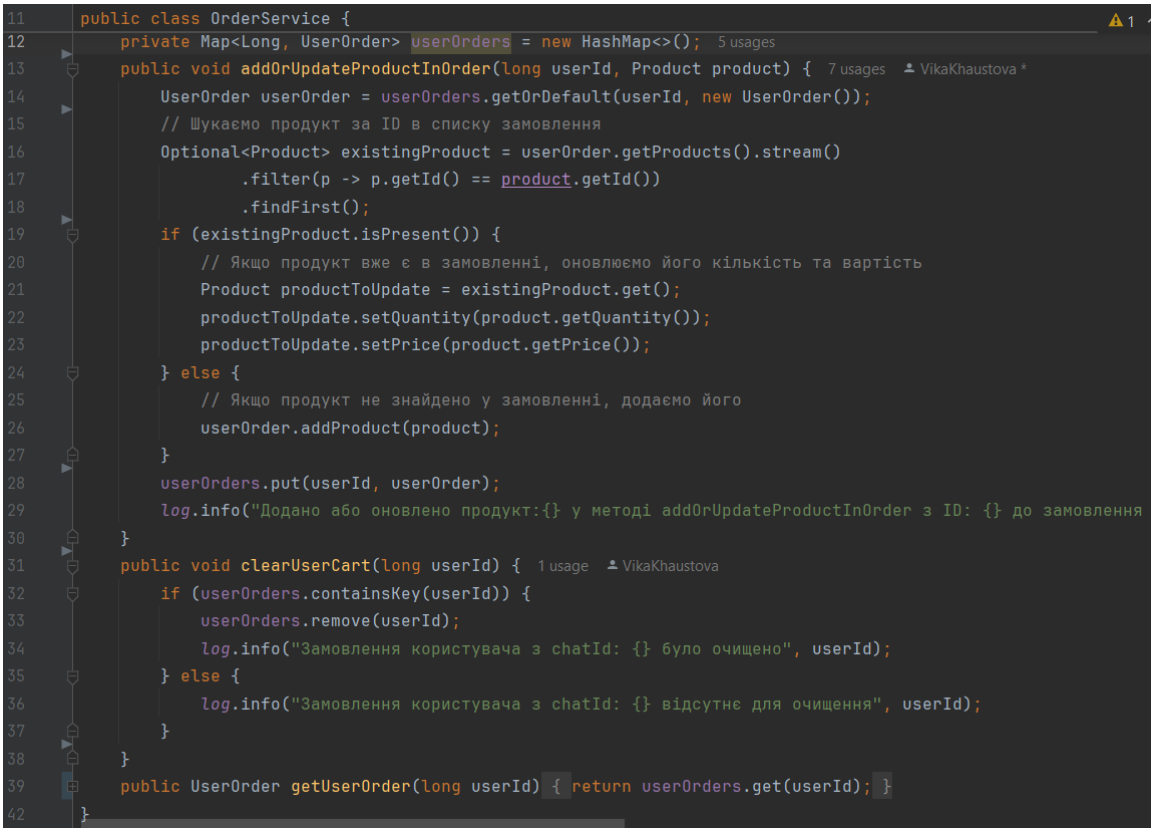
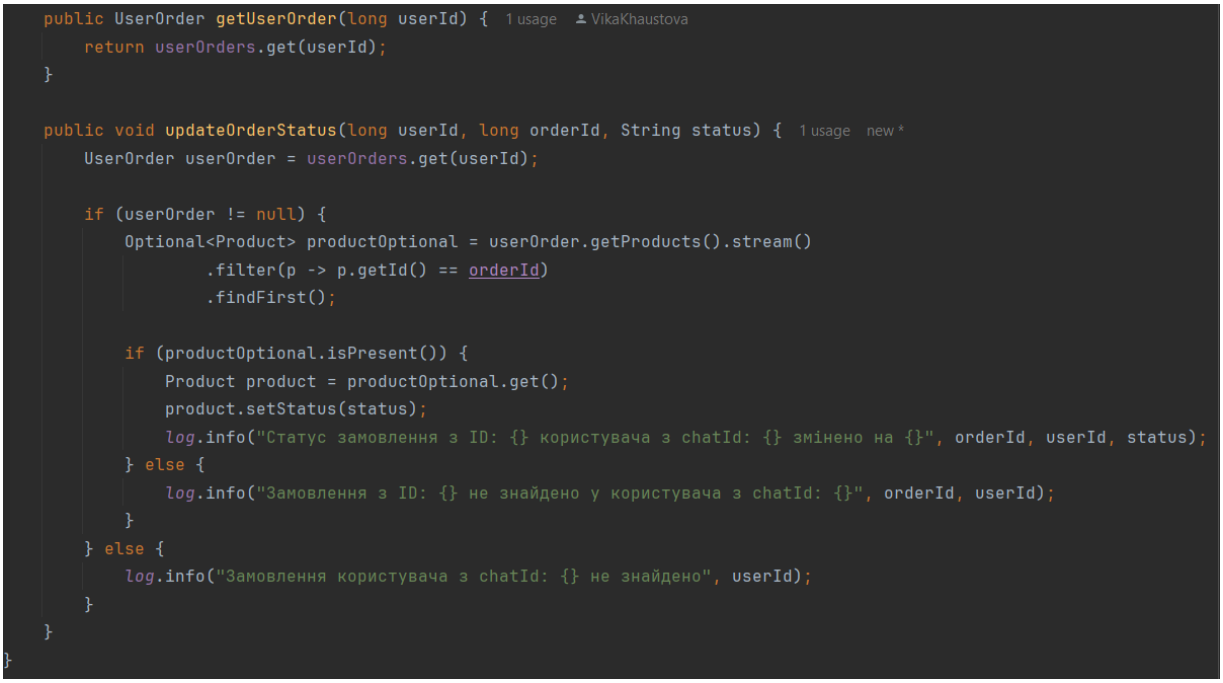


Рисунок 3.10 – Клас OrderService, додавання або прибирання продуктів



```
    public UserOrder getUserOrder(long userId) {  1 usage   VikaKhaustova
        return userOrders.get(userId);
    }

    public void updateOrderStatus(long userId, long orderId, String status) {  1 usage   new *
        UserOrder userOrder = userOrders.get(userId);

        if (userOrder != null) {
            Optional<Product> productOptional = userOrder.getProducts().stream()
                    .filter(p -> p.getId() == orderId)
                    .findFirst();

            if (productOptional.isPresent()) {
                Product product = productOptional.get();
                product.setStatus(status);
                log.info("Статус замовлення з ID: {} користувача з chatId: {} змінено на {}", orderId, userId, status);
            } else {
                log.info("Замовлення з ID: {} не знайдено у користувача з chatId: {}", orderId, userId);
            }
        } else {
            log.info("Замовлення користувача з chatId: {} не знайдено", userId);
        }
    }
}
```

Рисунок 3.11 – Клас OrderService, оновлення статусу замовлення

3.1.8 Опис класу TelegramBot

Цей клас забезпечує управління користувачами, обробку замовлень, надсилання повідомлень, управління підписками на новини та розсилку рекламних повідомлень. Він дозволяє ефективно взаємодіяти з користувачами та керувати їх замовленнями, підтримуючи зручний інтерфейс через Telegram-бота.

Метод registerUser(), зображений на рисунку 3.12, реєструє нового користувача, якщо він ще не збережений у базі даних. Використовується інформація з повідомлення для збереження даних користувача (ім'я, прізвище, ім'я користувача та час реєстрації).

```
2237  @    private void registerUser(Message msg) {  1 usage  VikaKhaustova
2238           long chatId = msg.getChatId();
2239           Optional<User> userOptional = userRepository.findById(chatId);
2240
2241           if (userOptional.isEmpty()) {
2242               User user = new User();
2243               user.setChatId(chatId);
2244               var chat = msg.getChat();
2245               user.setFirstName(chat.getFirstName());
2246               user.setLastName(chat.getLastName());
2247               user.setUserName(chat.getUserName());
2248               user.setRegisteredAt(new Timestamp(System.currentTimeMillis()));
2249               userRepository.save(user);
2250               log.info("New user saved: " + user);
2251           }
2252       }
```

Рисунок 3.12 – Метод registerUser()

Метод startCommandReceived() надсилає вітальне повідомлення новому користувачу, очищає його кошик та видаляє інформацію про кількість замовлених товарів. Ознайомитись з цим методом можна на рисунку 3.13.

```
2254      private void startCommandReceived(long chatId, String name) {  1 usage  VikaKhaustova
2255          String answer = EmojiParser.parseToUnicode( input: "Вітаємо, " + name + "!" + " :blush:");
2256          log.info("Replied to user " + name);
2257          sendMessage(chatId, answer, BotState.MAIN_MENU);
2258          orderService.clearUserCart(chatId);
2259          userPizzaQuantity.remove(chatId);
2260          userAdditionsQuantity.remove(chatId);
2261          userTeaQuantity.remove(chatId);
2262          userCoffeeQuantity.remove(chatId);
2263          userSweetDrinksQuantity.remove(chatId);
2264          userCocktailQuantity.remove(chatId);
2265      }
```

Рисунок 3.13 – Метод startCommandReceived()

Метод sendMessage() відповідає за надсилання повідомлення користувачу та оновлення стану бота. Він отримує поточний стан користувача, додає його до стека станів, після чого встановлює новий стан. Метод створює об'єкт повідомлення з заданим текстом, формує клавіатуру з відповідними кнопками і надсилає повідомлення користувачу, використовуючи бібліотеку Telegram API. Переглянути код метода можна на рисунку 3.14.

```java
private void sendMessage(long chatId, String textToSend, BotState nextState) {  1 usage  VikaKhaustova *
    BotState currentState = userStates.getOrDefault(chatId, BotState.MAIN_MENU);
    log.info("currentState in sendMessage "+ currentState);
    Stack<BotState> stateStack = userStateStack.computeIfAbsent(chatId, k -> new Stack<>());
    stateStack.push(currentState);
    userStates.put(chatId, nextState);
    log.info("Stack after adding to stack in sendMessage: {}", stateStack+"/n/n");
    SendMessage message = new SendMessage();
    message.setChatId(String.valueOf(chatId));
    message.setText(textToSend);
    ReplyKeyboardMarkup keyboardMarkup = new ReplyKeyboardMarkup();
    List<KeyboardRow> keyboardRows = new ArrayList<>();
    KeyboardRow row = new KeyboardRow();
    row.add("переглянути меню");
    row.add("зробити замовлення");
    keyboardRows.add(row);
    row = new KeyboardRow();
    row.add("підписатись на розсилку новин");
    keyboardRows.add(row);
    keyboardMarkup.setKeyboard(keyboardRows);
    message.setReplyMarkup(keyboardMarkup);
    executeMessage(message);
}
```

Рисунок 3.14 – Метод sendMessage()

Методи executeMessage() і prepareAndSendMessage() забезпечують функціональність надсилання повідомлень користувачам через бібліотеку Telegram API. Ознайомитись з цими методами можна на рисунку 3.15.

Метод executeMessage() безпосередньо надсилає повідомлення через бібліотеку Telegram API. Він приймає об'єкт SendMessage, що містить параметри повідомлення, і викликає метод execute для його відправлення. У випадку виникнення помилки метод фіксує її в логах.

Метод prepareAndSendMessage() створює повідомлення для надсилання. Він ініціалізує об'єкт SendMessage, встановлює ідентифікатор чату та текст повідомлення, після чого викликає executeMessage для відправлення цього повідомлення користувачу.

```
2292        //НАДСИЛАННЯ ПОВІДОМЛЕНЬ ЧЕРЕЗ БІБЛІОТЕКУ TELEGRAM API.
2293        private void executeMessage(SendMessage message) {  20 usages  VikaKhaustova
2294            try {
2295                execute(message);
2296            } catch (TelegramApiException e) {
2297                log.error(ERROR_TEXT + e.getMessage());
2298            }
2299        }
2300
2301        //НАДСИЛАННЯ ПОВІДОМЛЕНЬ
2302        private void prepareAndSendMessage(long chatId, String textToSend) {  16 usages  VikaKhaustova
2303            SendMessage message = new SendMessage();
2304            message.setChatId(String.valueOf(chatId));
2305            message.setText(textToSend);
2306            executeMessage(message);
2307        }
```

Рисунок 3.15 – Методи executeMessage() і prepareAndSendMessage()

Для забезпечення інтерактивної взаємодії користувачів з чат-ботом, дозволяючи їм переглядати меню, вибирати продукти, отримувати детальну інформацію про них і додавати їх до замовлення потрібні відповідні методи, які реалізують основні функції, необхідні для створення повноцінного замовлення, включаючи навігацію по меню, вибір продуктів, підтвердження кількості та оформлення замовлення. Вони роблять процес замовлення зручним і зрозумілим для користувачів.

Частина коду, зображена на рисунку 3.16, налаштовує основні параметри та поведінку Telegram бота для створення замовлень. Він визначає текст допомоги для користувачів, можливі стани бота, зберігання станів користувачів, режими роботи та конфігурацію бота. Конструктор бота встановлює команди для взаємодії користувачів з ботом, такі як /start і /help, а також обробляє можливі винятки, що виникають під час налаштування команд. Таким чином, цей блок забезпечує базову функціональність і структуру бота для подальшої розробки та використання.

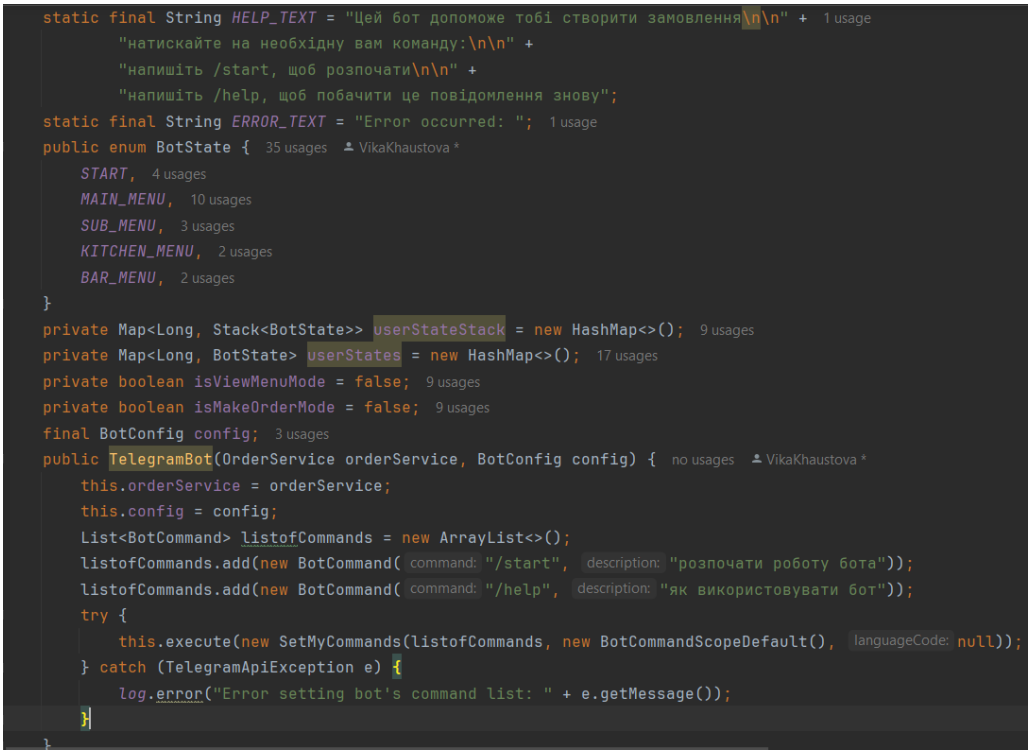


Рисунок 3.16 – Основні параметри Телеграм бота

Метод onUpdateReceived() обробляє всі оновлення, які надходять до Telegram-бота. Він перевіряє тип оновлення (текстове повідомлення або запит зворотного виклику) і визначає відповідний чат та ідентифікатор повідомлення. Потім метод обробляє різні типи повідомлень та запити зворотного виклику відповідно до поточного стану користувача. Основні функції методу включають:

– ідентифікація користувача та поточного стану, тобто отримання чат-ID та ідентифікатора повідомлення для обробки, визначення поточного стану користувача з мапи userStates (рисунок 3.17);

– збереження інформації про чат-ID, поточний стан та отримані повідомлення за допомогою логування для відстеження роботи бота та діагностики (рисунок 3.17);

– можливість повернення назад до попереднього меню реалізується за допомогою управління стеком станів користувача (рисунок 3.18);

– обробка текстових повідомлень, а саме виконання різних дій залежно від отриманого тексту повідомлення, наприклад: команди /start, /help,

переглянути меню (рисунок 3.19);

– обробка запитів зворотнього виклику - виконання відповідних дій при натисканні кнопок у інтерфейсі бота, наприклад: вибір піци, десертів, збільшення/зменшення кількості товарів (рисунок 3.20).

Цей метод забезпечує динамічну взаємодію користувачів з ботом, дозволяючи їм переходити між різними меню, робити замовлення та отримувати відповідні відповіді на їхні дії.

```
public void onUpdateReceived(Update update) {  no usages   VikaKhaustova *
    long chatId;
    Integer messageId = null;
    if (update.hasMessage()) {
        chatId = update.getMessage().getChatId();
        messageId = update.getMessage().getMessageId();
    } else if (update.hasCallbackQuery()) {
        chatId = update.getCallbackQuery().getMessage().getChatId();
        messageId = update.getCallbackQuery().getMessage().getMessageId();
    } else {
        return;
    }
```

Рисунок 3.17 – Частина методу onUpdateReceived(), реалізація ідентифікації користувача та поточного стану

```
BotState currentState = userStates.getOrDefault(chatId, BotState.START);
log.info("Chat ID: " + chatId + ", Current State: " + currentState);
if (update.hasMessage() && update.getMessage().hasText()) {
    String messageText = update.getMessage().getText();
    log.info("Received message: " + messageText);
    if (messageText.equals("Повернутися назад")) {
        // Отримуємо стек станів користувача
        Stack<BotState> stateStack = userStateStack.get(chatId);
        if (stateStack != null && !stateStack.isEmpty()) {
            // Видаляємо поточний стан
            stateStack.pop();
            // Отримуємо попередній стан
            BotState previousState = stateStack.isEmpty() ? BotState.MAIN_MENU : stateStack.peek();
            userStates.put(chatId, previousState);

            // Обробка логіки повернення назад
            handleBackButtonPressed(chatId);
        } else {
            // Якщо стек порожній, переводимо користувача в головне меню
            userStates.put(chatId, BotState.MAIN_MENU);
        }
        return;
    }
```

Рисунок 3.18 – Частина методу onUpdateReceived(), можливість повернення назад до попереднього меню

```
switch (messageText) {
    case "/start":
        registerUser(update.getMessage());
        startCommandReceived(chatId, update.getMessage().getChat().getFirstName());
        break;
    case "/help":
        prepareAndSendMessage(chatId, HELP_TEXT);
        break;
    case "переглянути меню":
        isViewMenuMode = true;
        isMakeOrderMode = false;
        sendSubMenu(chatId);
        break;
    case "зробити замовлення":
        isViewMenuMode = false;
        isMakeOrderMode = true;
        sendSubMenu(chatId);
        break;
    case "підписатись на розсилку новин":
        subscribeToNewsletter(chatId);
        break;
    case "відписатись від розсилки новин":
        subscribeToNewsletter(chatId);
        break;
    case "кухня":
        handleKitchenMenu(chatId);
        break;
```

Рисунок 3.19 – Частина методу onUpdateReceived(), обробка текстових повідомлень

```
if (callbackData.startsWith("dessert_")) {
    int dessertsId = Integer.parseInt(callbackData.substring("dessert_".length()));
    handleDessertSelection(chatId, dessertsId);
    return;
}
if (callbackData.startsWith("dessertsOrder_")) {
    int dessertId = Integer.parseInt(callbackData.substring("dessertsOrder_".length()));
    handleDessertSelectionToOrder(chatId,dessertId, messageId);
    return;
}

if (callbackData.startsWith("increase_dessert_")) {
    int dessertId = Integer.parseInt(callbackData.substring("increase_dessert_".length()));
    handleIncreaseDessertQuantity(chatId, dessertId, messageId);
    return;
}

if (callbackData.startsWith("decrease_dessert_")) {
    int dessertId = Integer.parseInt(callbackData.substring("decrease_dessert_".length()));
    handleDecreaseDessertQuantity(chatId, dessertId, messageId);
    return;
}

if (callbackData.startsWith("add_to_order_continue_dessert_")) {
    int dessertId = Integer.parseInt(callbackData.substring("add_to_order_continue_dessert_".length()));
    handleAddToOrderContinueForDessert(chatId, dessertId, messageId);
    return;
}
```

Рисунок 3.20 – Частина методу onUpdateReceived(), обробка запитів зворотнього виклику

Нижче наведено опис методів для замовлення і перегляду меню десертів: handleDessertsMenu(), handleDessertSelection(), handleDessertMenuToOrder(), getDessertById(), createDessertsMenu(), handleDessertSelectionToOrder(), createQuantityChangeKeyboardForDessert(), handleIncreaseDessertQuantity(), handleIncreaseDessertQuantity(), handleAddToOrderContinueForDessert(), updateDessertMessage(). Методи для замовлення піци, доповнень, чаю, кави, солодких напоїв та коктейлів є аналогічними за структурою і функціональністю.

За відображення списку десертів користувачеві відповідає метод handleDessertsMenu(). Він створює повідомлення з текстом "Список десертів:" та налаштовує клавіатуру з кнопками для кожного десерту, що дозволяє користувачу вибрати конкретний десерт. Потім це повідомлення відправляється користувачу через метод executeMessage.

Метод handleDessertSelection() виконується після вибору десерту користувачем. Він знаходить обраний десерт за його ідентифікатором, відправляє зображення десерту та надсилає повідомлення з детальною інформацією про десерт, включаючи назву, опис, вагу та вартість.

Ознайомитись з цими методами можна на рисунку 3.21.

```
2311    private void handleDessertsMenu(long chatId) {
2312        SendMessage message = new SendMessage();
2313        message.setChatId(String.valueOf(chatId));
2314        message.setText("Список десертів:");
2315        InlineKeyboardMarkup keyboardMarkup = new InlineKeyboardMarkup();
2316        List<List<InlineKeyboardButton>> rowsInLine = new ArrayList<>();
2317        List<Dessert> dessertsMenu = createDessertsMenu();
2318        for (Dessert dessert : dessertsMenu) {
2319            InlineKeyboardButton button = new InlineKeyboardButton();
2320            button.setText(dessert.getName());
2321            button.setCallbackData("dessert_" + dessert.getId());
2322            List<InlineKeyboardButton> rowInLine = new ArrayList<>();
2323            rowInLine.add(button);
2324            rowsInLine.add(rowInLine);
2325        }
2326        keyboardMarkup.setKeyboard(rowsInLine);
2327        message.setReplyMarkup(keyboardMarkup);
2328        executeMessage(message);
2329    } 1 usage  VikaKhaustova
2330    private void handleDessertSelection(long chatId, int dessertId) {
2331        Dessert selectedDessert = getDessertById(dessertId);
2332        sendPhoto(chatId, selectedDessert.getImageUrl());
2333        String dessertInfo = "Назва: " + selectedDessert.getName() + "\n";
2334        dessertInfo += "Опис: " + selectedDessert.getDescription() + "\n";
2335        dessertInfo += "Вага: " + selectedDessert.getWeight() + " г"+"\n";
2336        dessertInfo += "Вартість: " + selectedDessert.getPrice() + " грн" + "\n";
2337        prepareAndSendMessage(chatId, dessertInfo);
2338    }
```

Рисунок 3.21 – Методи handleDessertsMenu() і handleDessertSelection()

Здійснює пошук десерту у списку десертів за заданим ідентифікатором метод getDessertById(). Він повертає об'єкт десерту, якщо ідентифікатор знайдено, або null, якщо десерт з таким ідентифікатором не знайдено.

createDessertsMenu() — цей метод створює та повертає список десертів. Він ініціалізує кілька об'єктів десертів із відповідними властивостями, такими як назва, опис, вага, зображення та вартість, і додає їх до списку, який повертається.

Переглянути код методів можна на рисунку 3.22

Рисунок 3.22 – Методи getDessertById() і createDessertsMenu()

Метод handleDessertMenuToOrder() дуже схожий на handleDessertsMenu(). Обидва методи призначені для відображення меню десертів, але handleDessertsMenu() більше орієнтований на загальний перегляд десертів, тоді як handleDessertMenuToOrder() (рисунок 3.23) орієнтований на замовлення десертів з можливістю вибору кількості. Ця відмінність дозволяє системі розрізняти, яка дія має бути виконана після натискання кнопки: показ інформації про десерт чи вибір десерту для замовлення.

```
private void handleDessertMenuToOrder(long chatId) {  1 usage  VikaKhaustova *
    SendMessage message = new SendMessage();
    message.setChatId(String.valueOf(chatId));
    message.setText("Список десертів:");
    InlineKeyboardMarkup keyboardMarkup = new InlineKeyboardMarkup();
    List<List<InlineKeyboardButton>> rowsInLine = new ArrayList<>();
    List<Dessert> dessertMenu = createDessertsMenu();
    for (Dessert desserts : dessertMenu) {
        InlineKeyboardButton button = new InlineKeyboardButton();
        button.setText(desserts.getName());
        button.setCallbackData("dessertsOrder_" + desserts.getId());
        List<InlineKeyboardButton> rowInLine = new ArrayList<>();
        rowInLine.add(button);
        rowsInLine.add(rowInLine);
    }
    keyboardMarkup.setKeyboard(rowsInLine);
    message.setReplyMarkup(keyboardMarkup);
    executeMessage(message);
}
```

Рисунок 3.23 – Метод handleDessertMenuToOrder()

Аналогічна ситуація у методів handleDessertSelection() і handleDessertSelectionToOrder(). Обидва методи обробляють вибір десерту, але з різними цілями: перший показує інформацію про десерт, а другий дозволяє користувачу замовити десерт, вказавши його кількість. Ознайомитись з прикладом програмування методу handleDessertSelectionToOrder() можна на рисунку 3.24.

Для створення клавіатури для зміни кількості десерту, яка дозволятиме користувачу збільшувати або зменшувати кількість десерту, а також додавати його до замовлення — використовується метод createQuantityChangeKeyboardForDessert(), з яким можна ознайомитись на рисунку 3.25.

```
private void handleDessertSelectionToOrder(long chatId, int dessertId, int messageId) {  1 usage   VikaKhaustova *
    Dessert selectedDessert = getDessertById(dessertId);
    sendPhoto(chatId, selectedDessert.getImageUrl());
    String dessertInfo = "Назва: " + selectedDessert.getName() + "\n";
    dessertInfo += "Опис: " + selectedDessert.getDescription() +"\n";
    dessertInfo += "Вага: " + selectedDessert.getWeight() +" г"+ "\n";
    dessertInfo += "Вартість: " + selectedDessert.getPrice() + " грн" + "\n";
    double additionPrice = selectedDessert.getPrice();
    int quantity = userAdditionsQuantity
            .computeIfAbsent(chatId, k -> new HashMap<>())
            .getOrDefault(dessertId,  defaultValue: 0);
    dessertInfo += "КІЛЬКІСТЬ: " + quantity + "\n";
    dessertInfo += "Загальна вартість: " + String.format("%.2f", additionPrice * quantity) + " грн" + "\n";
    InlineKeyboardMarkup keyboardMarkup = createQuantityChangeKeyboardForDessert(dessertId);
    sendMessageWithKeyboardAndEdit(chatId, dessertInfo, keyboardMarkup, messageId);
}
```

Рисунок 3.24 – Метод handleDessertSelectionToOrder()

```
private InlineKeyboardMarkup createQuantityChangeKeyboardForDessert(int dessertId) {  3 usages   VikaKhaustova *
    InlineKeyboardMarkup keyboardMarkup = new InlineKeyboardMarkup();
    List<List<InlineKeyboardButton>> rowsInLine = new ArrayList<>();
    List<InlineKeyboardButton> plusMinusRow = new ArrayList<>();
    InlineKeyboardButton plusButton = new InlineKeyboardButton( text: "+");
    plusButton.setCallbackData("increase_dessert_" + dessertId);
    InlineKeyboardButton minusButton = new InlineKeyboardButton( text: "-");
    minusButton.setCallbackData("decrease_dessert_" + dessertId);
    plusMinusRow.add(plusButton);
    plusMinusRow.add(minusButton);
    List<InlineKeyboardButton> addToOrderContinueRow = new ArrayList<>();
    InlineKeyboardButton addToOrderContinueButton = new InlineKeyboardButton( text: "Додати до замовлення");
    addToOrderContinueButton.setCallbackData("add_to_order_continue_dessert_" + dessertId);
    addToOrderContinueRow.add(addToOrderContinueButton);
    rowsInLine.add(plusMinusRow);
    rowsInLine.add(addToOrderContinueRow);
    keyboardMarkup.setKeyboard(rowsInLine);
    return keyboardMarkup;
}
```

Рисунок 3.25 – Метод createQuantityChangeKeyboardForDessert()

У коді реалізовано два методи, які збільшують та зменшують кількість обраного десерту для конкретного користувача та оновлюють відповідне повідомлення в чаті, а саме це методи — handleIncreaseDessertQuantity() і handleIncreaseDessertQuantity(). Ознайомитись із ними можна на рисунку 3.26.

```
private void handleIncreaseDessertQuantity(long chatId, int dessertId, int messageId) {
    int currentQuantity = userDessertQuantity
            .computeIfAbsent(chatId, k -> new HashMap<>())
            .getOrDefault(dessertId,  defaultValue: 0);
    userDessertQuantity
            .computeIfAbsent(chatId, k -> new HashMap<>())
            .put(dessertId, currentQuantity + 1);
    updateDessertMessage(chatId, dessertId, messageId);
}

private void handleDecreaseDessertQuantity(long chatId, int dessertId, int messageId) {
    int currentQuantity = userDessertQuantity
            .computeIfAbsent(chatId, k -> new HashMap<>())
            .getOrDefault(dessertId,  defaultValue: 0);
    if (currentQuantity > 0) {
        userDessertQuantity
                .computeIfAbsent(chatId, k -> new HashMap<>())
                .put(dessertId, currentQuantity - 1);
        updateDessertMessage(chatId, dessertId, messageId);
    }
}
```

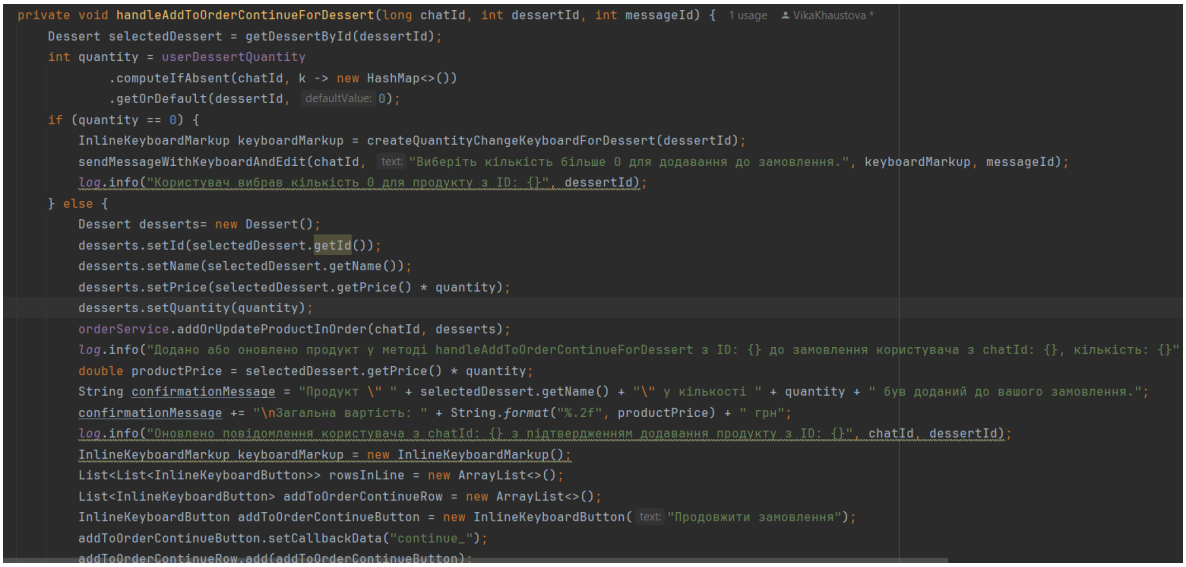
Рисунок 3.26 – Методи handleIncreaseDessertQuantity() і handleIncreaseDessertQuantity()

За оновлення повідомлення з інформацією про десерт та його кількість відповідає метод updateDessertMessage(), зображений на рисунку 3.27.

```
private void updateDessertMessage(long chatId, int dessertId, int messageId) {  2 usages   VikaKhaustova *
    Dessert selectedDessert = getDessertById(dessertId);
    String dessertInfo = "Назва: " + selectedDessert.getName() + "\n";
    dessertInfo += "Опис: " + selectedDessert.getDescription() +"\n";
    dessertInfo += "Вага: " + selectedDessert.getWeight() +" г"+ "\n";
    dessertInfo += "Вартість: " + selectedDessert.getPrice() + " грн" + "\n";
    double dessertPrice = selectedDessert.getPrice();
    int quantity = userDessertQuantity
            .computeIfAbsent(chatId, k -> new HashMap<>())
            .getOrDefault(dessertId,  defaultValue: 0);
    dessertInfo += "КІЛЬКІСТЬ: " + quantity + "\n";
    dessertInfo += "Загальна вартість: " + String.format("%.2f", dessertPrice * quantity) + " грн" + "\n";
    InlineKeyboardMarkup keyboardMarkup = createQuantityChangeKeyboardForDessert(dessertId);
    sendMessageWithKeyboardAndEdit(chatId, dessertInfo, keyboardMarkup, messageId);
}
```

Рисунок 3.27 – Методи updateDessertMessage()

handleAddToOrderContinueForDessert() — цей метод обробляє підтвердження додавання десерту до замовлення. Він перевіряє кількість обраного десерту, додає його до замовлення та оновлює повідомлення з підтвердженням. Переглянути частину цього методу можна на рисунку 3.28.



```
private void handleAddToOrderContinueForDessert(long chatId, int dessertId, int messageId) {  1 usage   VikaKhaustova *
    Dessert selectedDessert = getDessertById(dessertId);
    int quantity = userDessertQuantity
            .computeIfAbsent(chatId, k -> new HashMap<>())
            .getOrDefault(dessertId,  defaultValue: 0);
    if (quantity == 0) {
        InlineKeyboardMarkup keyboardMarkup = createQuantityChangeKeyboardForDessert(dessertId);
        sendMessageWithKeyboardAndEdit(chatId,  text: "Виберіть кількість більше 0 для додавання до замовлення.", keyboardMarkup, messageId);
        log.info("Користувач вибрав кількість 0 для продукту з ID: {}", dessertId);
    } else {
        Dessert desserts= new Dessert();
        desserts.setId(selectedDessert.getId());
        desserts.setName(selectedDessert.getName());
        desserts.setPrice(selectedDessert.getPrice() * quantity);
        desserts.setQuantity(quantity);
        orderService.addOrUpdateProductInOrder(chatId, desserts);
        log.info("Додано або оновлено продукт у методі handleAddToOrderContinueForDessert з ID: {} до замовлення користувача з chatId: {}, кількість: {}"
        double productPrice = selectedDessert.getPrice() * quantity;
        String confirmationMessage = "Продукт \" " + selectedDessert.getName() + "\" у кількості " + quantity + " був доданий до вашого замовлення.";
        confirmationMessage += "\nЗагальна вартість: " + String.format("%.2f", productPrice) + " грн";
        log.info("Оновлено повідомлення користувача з chatId: {} з підтвердженням додавання продукту з ID: {}", chatId, dessertId);
        InlineKeyboardMarkup keyboardMarkup = new InlineKeyboardMarkup();
        List<List<InlineKeyboardButton>> rowsInLine = new ArrayList<>();
        List<InlineKeyboardButton> addToOrderContinueRow = new ArrayList<>();
        InlineKeyboardButton addToOrderContinueButton = new InlineKeyboardButton( text: "Продовжити замовлення");
        addToOrderContinueButton.setCallbackData("continue_");
        addToOrderContinueRow.add(addToOrderContinueButton);
```

Рисунок 3.28 – Метод handleAddToOrderContinueForDessert()

Методи sendSundayAd() і sendThursdayAd() використовуються для автоматичної розсилки рекламних повідомлень користувачам, які підписані на новини, у визначений день та час. Це дозволяє підтримувати регулярний контакт з користувачами та інформувати їх про спеціальні пропозиції чи акції, підвищуючи тим самим зацікавленість та залученість. Методи отримують список усіх користувачів з бази даних через репозиторій userRepository. Для кожного користувача, який підписаний на новини (перевіряється методом user.isSubscribedToNews()), відправляється рекламне повідомлення. Ознайомитись із кодом методів можна на рисунку 3.29.

Метод sendSundayAd() налаштований на запуск кожну неділю о 17:40. Користувачам відправляється рекламне повідомлення "Кожну неділю після 18-00 на другу піццу -50%" через метод prepareAndSendMessage(). Переглянути виконання цього методу можні на рисунку 3.30.

Аналогічно працює метод sendThursdayAd(). У четвер о 18:00 надсилається повідомлення "1+1=3 на піццу".

```java
//РЕКЛАМА ПО НЕДІЛЯХ
@Scheduled(cron = "0 30 17 * * TUE") // О 17:30 неділі  no usages  VikaKhaustova *
private void sendSundayAd() {
    var users = userRepository.findAll();
    var adText = "Кожну неділю після 18-00 на другу піццу -50%";

    for (User user : users) {
        if (user.isSubscribedToNews()) {
            prepareAndSendMessage(user.getChatId(), adText);
        }
    }
}
//РЕКЛАМА ПО ЧЕТВЕРГАХ
@Scheduled(cron = "0 0 10 * * THU") // О 10:00 четверга  no usages  VikaKhaustova
private void sendThursdayAd() {
    var users = userRepository.findAll();
    var adText = "1+1=3 на піццу";

    for (User user : users) {
        if (user.isSubscribedToNews()) {
            prepareAndSendMessage(user.getChatId(), adText);
        }
    }
}
```

Рисунок 3.29 – Методи sendSundayAd() і sendThursdayAd()

Рисунок 3.30 – Виконання методу sendSundayAd()

Клас TelegramBot є ключовим компонентом для роботи Telegram-бота, який забезпечує інтерактивність та зручність користування. Він включає функції для обробки оновлень, управління станами користувачів, обробки повідомлень та зворотних викликів, а також регулярного надсилання рекламних повідомлень. Цей клас інтегрує основні сервіси, такі як OrderService та UserRepository, що дозволяє ефективно керувати замовленнями та взаємодією з користувачами.

3.1.9 Клас TgbotDiplomApplication

Клас TgbotDiplomApplication, що зображений на рисунку 3.31 забезпечує конфігурацію та запуск всього застосунку, що дозволяє Telegram-боту взаємодіяти з користувачами та базою даних, використовуючи можливості Spring Boot та Spring Data JPA.

Рисунок 3.31 – Клас TgbotDiplomApplication

Клас TgbotDiplomApplication є стартовою точкою для запуску Spring Boot застосунку, який реалізує Telegram-бота.

Анотація @SpringBootApplication — це поєднання трьох анотацій: @Configuration, @EnableAutoConfiguration, та @ComponentScan. Вона вказує Spring Boot, що це основний клас конфігурації та запуску застосунку.

Анотація @EnableJpaRepositories вказує на те, що слід сканувати зазначений пакет (diplom.telegramBotDiplom.model) для пошуку JPA репозиторіїв. Це дозволяє Spring Data JPA автоматично генерувати реалізації репозиторіїв на основі інтерфейсів.

Метод main викликає SpringApplication.run, який запускає застосунок. Цей метод ініціює контекст Spring, що дозволяє Spring Boot автоматично налаштовувати всі необхідні компоненти та залежності.

3.2 Взаємодія Телеграм боту із співробітниками кухні

Для ефективної взаємодії між користувачами Telegram боту і співробітниками кухні використовується три основні компоненти: телефон з Telegram ботом, сервер для обробки замовлень і сенсорний дисплей на базі Raspberry Pi. Користувачі надсилають замовлення через Telegram бот, сервер обмінюється інформацією з базою даних для зберігання та отримання даних про замовлення і надсилає їх на дисплей, розташований на кухні, де співробітники можуть оновлювати статус замовлень. Ця система, зображена на рисунку 3.32, дозволяє забезпечити безперебійну і зручну комунікацію між клієнтами і кухнею. Для реалізації цієї системи необхідно виконати кілька кроків, включаючи підключення мікроконтролера ESP-32 до Raspberry Pi, налаштування веб-додатку на Raspberry Pi і інтеграцію його з сервером. Далі представлено детальний алгоритм дій.

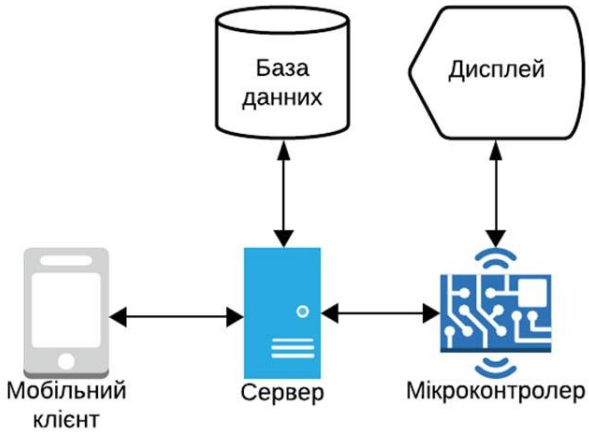


Рисунок 3.32 – Структурна схема системи

Першим кроком є установка ОС на Raspberry Pi. Для цього, по-перше, треба завантажите Raspberry Pi Imager з офіційного сайту Raspberry Pi. По-друге, вставити microSD карту в комп'ютер і запустити Raspberry Pi Imager. По-третє, вибрати операційну систему (наприклад, Raspberry Pi OS). Далі, вибрати microSD карту і натиснути "Write". Після запису ОС вставити microSD карту в Raspberry Pi і підключити його до живлення.

Другим кроком є підключення ESP-32 до одного з USB портів Raspberry

Pi за допомогою USB кабелю, що зображено на рисунку 3.33.

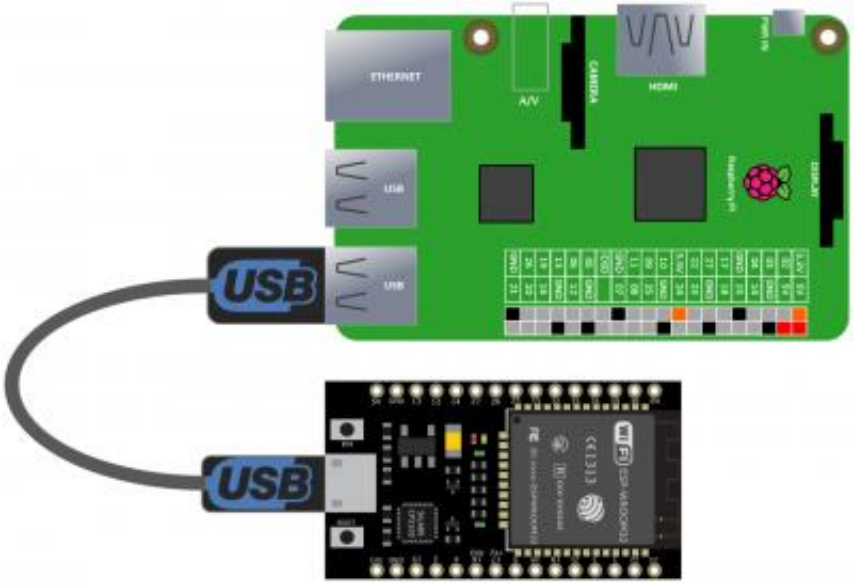


Рисунок 3.33 – Підключення ESP-32 до Raspberry Pi

Третім кроком є налаштування серійного зв'язку. Серійний зв’язок – це метод обміну даними між комп'ютерами та периферійними пристроями, при якому дані передаються по одному біту за раз через один або більше каналів зв'язку. Серійний зв'язок використовується для підключення різноманітних пристроїв, таких як модеми, принтери, сенсори, мікроконтролери, а також для внутрішнього зв'язку між компонентами комп'ютера.

Для початку, необхідно відкрити термінал на Raspberry Pi і встановити необхідні пакети для серійного зв'язку. Для цього в терміналі вводимо рядки, зображені на рисунку 3.34. Для підключення до ESP-32 через серійний порт вводимо рядок зображений на рисунку 3.35.

```
sudo apt update
sudo apt install minicom
```

Рисунок 3.34 – Встановлення необхідних пакетів для серійного зв'язку

```
minicom -b 115200 -o -D /dev/ttyUSB0
```

Рисунок 3.35 – Підключення до ESP-32 через серійний порт

Четвертим кроком є процес завантаження прошивки на ESP-32, який розпочинається з завантаження Arduino IDE на комп'ютер. Після встановлення Arduino IDE необхідно додати підтримку ESP-32. Для цього запустити Arduino IDE, перейти до меню File -> Preferences і у полі Additional Board Manager URLs додати наступний URL: https://dl.espressif.com/dl/package_esp32_index.json. Потім перейти до меню Tools -> Board -> Boards Manager, знайти "esp32" у списку і натиснути Install.

Код для ESP-32, який буде взаємодіяти з сервером і відображати дані на дисплеї з використанням HTTP запиту зображений на рисунках 3.36 і 3.37.

```
#include <WiFi.h>
#include <HTTPClient.h>
#include <ArduinoJson.h>
#include <Adafruit_GFX.h>
#include <Adafruit_ILI9341.h>

const char* ssid = "my_SSID"; //  SSID Wi-Fi мережі
const char* password = "my_PASSWORD"; // пароль до Wi-Fi мережі

#define TFT_CS     15
#define TFT_RST    4
#define TFT_DC     2

Adafruit_ILI9341 tft = Adafruit_ILI9341(TFT_CS, TFT_DC, TFT_RST);

void setup() {
  Serial.begin(115200);

  // Підключення до WiFi
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi...");
  }
  Serial.println("Connected to WiFi");

  // Ініціалізація дисплея
  tft.begin();
  tft.setRotation(3); // Орієнтація дисплея
  tft.fillScreen(ILI9341_BLACK); // Заливка екрану чорним кольором
  tft.setTextColor(ILI9341_WHITE); // Колір тексту білий
  tft.setTextSize(2); // Розмір тексту
}
```

Рисунок 3.36 – Код для взаємодії ESP-32 із сервером, функція setup()

```
void loop() {
  if (WiFi.status() == WL_CONNECTED) {
    HTTPClient http;
    http.begin("http://server.com/api/orders"); //адреса сервера
    int httpResponseCode = http.GET();

    if (httpResponseCode > 0) {
      String payload = http.getString();
      Serial.println(payload); // Відображення отриманих даних у серійному моніторі

      // Парсинг JSON
      DynamicJsonDocument doc(2048); // Збільшено розмір для більшого JSON
      deserializeJson(doc, payload);
      JsonArray orders = doc["orders"];

      // Відображення даних на дисплеї
      tft.fillScreen(ILI9341_BLACK); // Очищення екрану
      tft.setCursor(0, 0); // Встановлення курсору у верхній лівий кут
      for (JsonObject order : orders) {
        tft.print("Order ID: ");
        tft.println(order["id"].as<String>());
        tft.print("Order: ");
        tft.println(order["order"].as<String>());
        tft.print("Status: ");
        tft.println(order["status"].as<String>());
        tft.println(); // Додати порожній рядок між замовленнями
      }
    } else {
      Serial.print("Error on HTTP request: ");
      Serial.println(httpResponseCode);
    }
    http.end();
  }
  delay(10000); // Очікування 10 секунд перед наступним запитом
}
```

Рисунок 3.37 – Код для взаємодії ESP-32 із сервером, функція loop()

Код зазначений на рисунках вище підключає ESP-32 до Wi-Fi мережі, відправляє HTTP-запити до сервера для отримання списку замовлень, парсить отримані дані у форматі JSON, і відображає їх на сенсорному дисплеї. Зокрема, він показує ідентифікатор замовлення, опис замовлення та його статус, відповідно до 3.38. Кожні 10 секунд відбувається оновлення даних шляхом надсилання нового запиту до сервера.

```
{
  "orders": [
    {"id": 1, "status": "Готується", "order": "Піцца Маргарита і чорний чай"},
    {"id": 2, "status": "Готово", "order": "Чізкейк і капучино"},
    {"id": 3, "status": "Готово", "order": "Лимонад полуниця/чизкейк і піцца Капрічоза"}
  ]
}
```

Рисунок 3.38 – Приклад отриманих від сервера даних у форматі JSON

Цей код отримує список замовлень у форматі JSON із сервера та відображає кожне замовлення на дисплеї. Замовлення надходять у вигляді масиву об'єктів JSON.

П’ятим пунктом є налаштування середовища розробки на Paspberry Pi і написання веб-додатку на ньому. Для початку встановимо веб-сервер Apache на Raspberry Pi. Після цього встановимо необхідне ПЗ для розробки Visual Studio Code, який буде використовуватись для написання та редагування коду.

Для відображення на дисплеї списку замовлень зробимо веб-додаток. Почнемо з файлу index.html в кореневій папці веб-сервера. Цей файл містить структуру веб-сторінки. Ознайомитись з кодом файлу index.html можна на рисунку 3.39.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Order Management</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <div id="orders-container">
        <!-- Інформація про замовлення буде динамічно додаватися сюди -->
    </div>
    <script src="script.js"></script>
</body>
</html>

```

Рисунок 3.39 – Код файлу index.html

Створимо файл styles.css для оформлення сторінки. Цей файл міститиме стилі для веб-додатка, відповідно до рисунку 3.40.

```
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    background-color: #f4f4f4;
}
#orders-container {
    width: 80%;
    margin: 20px auto;
    padding: 10px;
    background-color: #fff;
    border: 1px solid #ccc;
    box-shadow: 2px 2px 12px rgba(0, 0, 0, 0.1);
}
.order {
    padding: 10px;
    border-bottom: 1px solid #ccc;
}
.order:last-child {
    border-bottom: none;
}
.order-status {
    font-weight: bold;
}
```

Рисунок 3.40 – Код файлу styles.css

Створимо файл script.js для отримання та відображення даних замовлень. Цей файл містить логіку для динамічного відображення даних на сторінці і налаштовує періодичне оновлення даних кожні 5 секунд, забезпечуючи постійне відображення актуальної інформації про замовлення. Переглянути код цього файлу можна на рисунку 3.41.

Останнім кроком залишається підключити Raspberry Pi до локальної мережі. Для цього необхідно відредагувати файл конфігурації Wi-Fi /etc/wpa_supplicant/wpa_supplicant.conf, додавши дані необхідної мережі, як вказано на рисунку 3.42. Після цього перезапустити мережевий інтерфейс, за допомогою рядку «sudo wpa_cli -i wlan0 reconfigure».

```
document.addEventListener('DOMContentLoaded', function() {
    fetchOrders();
    setInterval(fetchOrders, 5000);  // Оновлення кожні 5 секунд
});

function fetchOrders() {
    fetch('/api/orders')  // Запит до вашого API
        .then(response => response.json())
        .then(data => {
            const ordersContainer = document.getElementById('orders-container');
            ordersContainer.innerHTML = '';
            data.orders.forEach(order => {
                const orderElement = document.createElement('div');
                orderElement.className = 'order';
                orderElement.innerHTML = `
                    <p>Order ID: ${order.id}</p>
                    <p>Product: ${order.product}</p>
                    <p class="order-status">Status: ${order.status}</p>
                    <button onclick="updateOrderStatus(${order.id}, 'Готово')">Mark as Ready</button>
                `;
                ordersContainer.appendChild(orderElement);
            });
        });
}

function updateOrderStatus(orderId, status) {
    fetch(`/api/orders/${orderId}`, {
        method: 'PUT',
        headers: {
            'Content-Type': 'application/json',
        },
        body: JSON.stringify({ status }),
    })
    .then(response => response.json())
    .then(() => {
        fetchOrders();  // Оновити список замовлень після зміни статусу
    });
}
```

Рисунок 3.41 – Код файлу script.js

```
network={
    ssid="your_SSID"
    psk="your_PASSWORD"
}
```

Рисунок 3.42 – Додавання даних мережі

Виконавши всі ці кроки, ми забезпечили інтеграцію всієї системи, яка дозволяє обробляти замовлення, відображати їх та оновлювати статуси в реальному часі.

3.3 Аналіз результатів

В рамках даного проекту був розроблений чат-бот, що використовується для автоматизованого обслуговування клієнтів у сфері харчування. Основним завданням було створення ефективної системи замовлення їжі та надання інформації про меню закладу через популярний месенджер Telegram.

Переглянути алгоритм роботи чат боту можна у додатку А. Ці схеми включають в себе послідовність дій для перегляду меню, замовлення страв і можливості підписатись на новини закладу, щоб отримувати вигідні пропозиції. Реалізація цих алгоритмів дозволила створити функціональний та зручний інструмент для клієнтів, а також оптимізувати робочі процеси для персоналу закладу харчування. Ознайомитись з скріншотами функціоналу боту можна у Додатку Б.

Для проведення аналізу результатів і ефективності системи, необхідно врахувати різні аспекти, пов'язані з функціонуванням чат-бота, мікроконтролера ESP-32 і інтеграцією з сервером. Основними критеріями оцінки є швидкість обробки замовлень, точність введення та передачі даних, рівень задоволеності клієнтів і персоналу, а також можливість збирання та аналізу аналітичних даних.

Перш за все, важливо зазначити, що система забезпечує миттєву передачу замовлень від клієнтів на кухню через Telegram бота. Це дозволяє швидко розпочати процес приготування їжі, зменшуючи час очікування для клієнтів. Відгуки клієнтів, зібрані через опитування після використання бота, показують високий рівень задоволеності завдяки зручності та швидкості обслуговування.

Точність введення замовлень також значно покращилася. Використання чат бота мінімізує ризик людських помилок, таких як неправильне записування замовлень, що часто трапляється при усних замовленнях. Автоматизована система дозволяє клієнтам самостійно вносити свої дані, що зменшує ймовірність помилок.

Впровадження мікроконтролера ESP-32 значно спрощує роботу персоналу. Замовлення тепер надходять безпосередньо на дисплей Raspberry Pi, що зменшує час на передачу інформації і покращує організацію роботи на кухні.

Система також забезпечує збір даних про обсяги продажів, популярність певних страв та інші аналітичні показники. Ця інформація може бути використана для оптимізації асортименту, планування закупівель і стратегічного управління закладом. Наприклад, аналіз популярності страв дозволяє швидко реагувати на зміну смаків клієнтів і вносити корективи до меню.

Важливою перевагою є можливість відправлення клієнтам повідомлень про статус їхніх замовлень. Коли замовлення готове, клієнт отримує сповіщення через Telegram, що зменшує їхній час очікування і підвищує рівень обслуговування. Це особливо актуально для закладів з великим потоком клієнтів, де своєчасне інформування є критичним фактором.

Для оцінки ефективності системи також важливо врахувати економічні показники. Використання автоматизованої системи дозволяє зменшити витрати на персонал, адже частина завдань, що виконувалися вручну, тепер автоматизована. Це дозволяє закладу зосередитися на підвищенні якості їжі та

сервісу, що в кінцевому рахунку призводить до збільшення кількості задоволених клієнтів і підвищення прибутковості.

Додатково, система сприяє підвищенню ефективності маркетингових кампаній. За допомогою Telegram бота клієнти можуть підписуватися на розсилку новин і отримувати інформацію про акції та спеціальні пропозиції. Це дозволяє закладу підтримувати контакт з клієнтами, інформувати їх про новинки в меню і стимулювати повторні візити.

Ще одним аспектом є безпека даних. Всі дані про замовлення та клієнтів зберігаються на сервері, що забезпечує їх надійний захист і дозволяє уникнути втрати інформації. Регулярні резервні копії та налаштування безпеки серверу гарантують захист від несанкціонованого доступу і кібератак.

Позитивний досвід клієнтів і персоналу підтверджує, що впровадження системи сприяє загальному покращенню роботи закладу. Клієнти отримують швидке та якісне обслуговування, а персонал - зручний інструмент для роботи. Це створює сприятливу атмосферу для розвитку бізнесу і підтримки високого рівня обслуговування.

Загалом, результати дають зрозуміти, що автоматизована система обслуговування клієнтів є ефективним рішенням для закладів харчування. Вона дозволяє зменшити час обробки замовлень, покращити точність даних, підвищити рівень задоволеності клієнтів і забезпечити зручність для персоналу. Інтеграція мікроконтролера ESP-32 з системою додатково спрощує процеси на кухні, що позитивно впливає на загальну продуктивність.

Використання новітніх технологій дозволяє не лише оптимізувати робочі процеси, але й створити конкурентні переваги на ринку. Це особливо актуально в умовах високої конкуренції, де кожен елемент обслуговування відіграє важливу роль у задоволенні потреб клієнтів. Таким чином, впровадження системи автоматизованого обслуговування на базі Telegram бота та мікроконтролера ESP-32 є важливим кроком до підвищення ефективності та якості обслуговування у сфері харчування. Це рішення дозволяє не лише покращити взаємодію з клієнтами, але й оптимізувати внутрішні процеси

закладу, що в кінцевому результаті призводить до збільшення прибутковості та задоволення клієнтів.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи була створена система автоматизованого обслуговування клієнтів у сфері харчування на базі Telegram чат-бота, мікроконтролера ESP-32 та веб-додатка. Основним завданням даного проекту було розробити ефективний механізм прийому та обробки замовлень, що забезпечує зручність як для клієнтів, так і для персоналу закладу харчування.

Система включає в себе кілька ключових компонентів, що працюють у тісній взаємодії. Telegram бот виступає основним інтерфейсом для клієнтів, надаючи їм можливість переглядати меню, робити замовлення та отримувати інформацію про статус своїх замовлень. Інтеграція мікроконтролера ESP-32 дозволяє кухонному персоналу оперативно отримувати інформацію про нові замовлення та їхні зміни, що значно спрощує робочі процеси.

Основні переваги впровадженої системи були підтверджені у 3 розділі. В першу чергу, це значне підвищення швидкості обслуговування клієнтів. Завдяки автоматизації процесів замовлення, час очікування значно зменшився, що позитивно вплинуло на задоволеність клієнтів. Крім того, знизилась кількість помилок, пов'язаних з людським фактором, оскільки замовлення тепер обробляються автоматично.

На підставі отриманих результатів можна зробити висновок, що використання сучасних технологій, таких як Telegram бот та мікроконтролери, є доцільним та ефективним для автоматизації процесів у сфері обслуговування. Подальший розвиток даної системи може включати впровадження нових функцій, таких як автоматизоване управління запасами, інтеграція з платіжними системами та розширення можливостей аналітики.

Отже, проведена робота не лише вирішила поставлені завдання, але й відкрила нові перспективи для вдосконалення та розвитку автоматизованих систем у сфері харчування. Сучасні технології дозволяють створювати гнучкі та ефективні рішення, що здатні значно покращити якість обслуговування та оптимізувати бізнес-процеси, забезпечуючи стабільний розвиток та зростання

бізнесу.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Термінали самообслуговування в McDonald's: світові технології з українською специфікою. [Електронний ресурс] – URL: https://systemgroup.com.ua/uk/project/terminaly-samoobslugovuvannya-v-mcdonalds (Дата звернення 08.05.2024).

2. Інтерактивний фуд-корт — інноваційна платформа для комунікацій. [Електронний ресурс] – URL: https://mmr.ua/show/interaktyvnyj-fud-kort (Дата звернення 09.05.2024).

3. Один QR для всього [Електронний ресурс] – URL: https://expz.monobank.ua/ (Дата звернення 10.05.2024).

4. Продукти Choice QR [Електронний ресурс] – URL: https://choiceqr.com/uk/products/ (Дата звернення 10.05.2024).

5. У Києві запустили чат-бот для оплати рахунків у ресторанах. [Електронний ресурс] – URL: https://rau.ua/novyni/chat-bot-dlya-oplaty-schetov/ (Дата звернення 11.05.2024).

6. Мова програмування Java: її плюси та мінуси. [Електронний ресурс] – URL: https://life.obozrevatel.com/ukr/section-zhizn/news-yazyik-programmirovaniya-java (Дата звернення 13.05.2024).

7. Мікроконтролер ESP-32 [Електронний ресурс] – URL: https://itmaster.biz.ua/directory/microcontrollers/esp32.html (Дата звернення 14.05.2024).

8. Дисплей Raspberry Pi 7" Touch Screen Display, оригінальний – URL: https://evo.net.ua/raspberry-pi-7-touch-screen-display/ (Дата звернення 15.05.2024).

9. Мікрокомп'ютер Raspberry Pi 4 Model B 2GB – URL: https://evo.net.ua/mikrokomputer-raspberry-pi-4-model-b-2gb/ (Дата звернення 15.05.2024).

10. Основи Maven – URL: https://javarush.com/ua/groups/posts/ (Дата звернення 17.05.2024).