Міністерство освіти і науки України
Національний університет «Одеська політехніка»
Інститут комп’ютерних систем
Кафедра інформаційних систем

Семеренко Микита Олегович,
студент групи AI-216

ДИСЦИПЛІНА
Об’єктно-орієнтоване програмування

КУРСОВА РОБОТА
Розробка телеграм-бота для замовлення доставки їжі

Спеціальність:
122 Комп’ютерні науки

Освітня програма:
Комп’ютерні науки

Керівник:
Годовиченко Микола Анатолійович,
кандидат технічних наук, доцент

Одеса – 2023

ЗМІСТ

АНОТАЦІЯ

Курсова робота присвячена розробці телеграм-бота, основною функцією якого є надання можливості замовлення доставки їжі з ресторану. Розробка призначена для замовлення їжі безпосередньо з месенджеру Телеграм, щоб, не відволікаючись від важливих чатів та спілкувань, можна було швидко та зручно замовити доставку своєї улюбленої страви на обідню перерву або вечерю. Основною метою роботи є створення простого, проте ефективного та інтуїтивно зрозумілого телеграм-бота, за допомогою якого, будь-яка людина зможе замовити будь-яку обрану страву всього за декілька хвилин, без необхідності тривалого очікування на дзвінок та багатьох уточнень стосовно створеного замовлення. Для розробки використовуємо мову програмування Java, Spring Framework та Telegram API для створення основних елементів бота та його управління, а також Java Persistence API для створення зв’язку з базою даних.

ABSTRACT

The course work is devoted to the development of a Telegram bot, the main function of which is to provide the ability to order food delivery from a restaurant. The development is designed to order food directly from the Telegram messenger, so that without being distracted from important chats and conversations, you can quickly and conveniently order the delivery of your favorite meal for lunch or dinner. The main goal of the work is to create a simple, but effective and intuitive Telegram bot, with the help of which anyone can order any dish of their choice in just a few minutes, without the need for a long wait for a call and many clarifications regarding the order. For the development, I use Java programming language, Spring Framework, and Telegram API to create the main elements of the bot and its management, as well as Java Persistence API to create a connection with the database.

ВСТУП

В сучасному світі користування різними месенджерами стало повсякденною справою кожної людини. Месенджер є невід'ємною частиною нашого життя: спілкування зі знайомими, друзями та колегами, перегляд актуальних новин, читання різних статей, перегляд відео та прослуховування музики — все це зосереджено в одному застосунку, через що виконання всіх цих дій одночасно не завдає незручності. Таким чином, відпадає гостра необхідність завантажувати різні додатки для кожної з цих потреб, значно зручніше використовувати лише один додаток для більшості випадків.

З урахуванням цього, на мою думку, є дуже доцільним створення телеграм-боту, за допомогою якого, можна всього за декілька хвилин замовити доставку їжі, без необхідності встановлювати окремі додатки, шукати різні сайти, та взагалі, відволікатися від важливих чатів та спілкувань. Мета роботи полягає в створенні простого та інтуїтивно-зрозумілого телеграм-боту, щоб будь-який користувач, не відволікаючись від своїх справ, міг швидко та зручно замовити доставку їжі.

Розробка телеграм-боту передбачає ознайомлення з роботою Telegram API, за допомогою якого відбувається створення, налаштування та управління роботою самого бота, а також наявність навичок роботи з такими технологіями як Spring Framework та Java Persistence API. Такий набір технологій та фреймворків дозволить створити ефективного та простого, проте в той же час зручного та стабільного бота, який буде мати гнучкий функціонал та інтуїтивно-зрозумілий інтерфейс, що в майбутньому буде сприяти використанню саме цього телеграм-бота.

Таким чином, ми визначили, що розробка телеграм-бота для замовлення їжі є актуальним рішенням, оскільки це відкидає необхідність використання сторонніх додатків та веб-сайтів для оформлення доставки. Саме тому, метою курсової роботи є створення, проектування та розробка телеграм-бота, за

допомогою якого будь-який користувач зможе з легкістю за доволі короткий час зробити замовлення. Для досягнення цієї мети, необхідно:

- оглянути схожі телеграм-боти або застосунки, які пропонують послуги з доставки їжі;
- проаналізувати та дослідити технології, які допоможуть нам при створенні бота;
- провести проектування телеграм-бота;
- виконати програмну реалізацію телеграм-бота.

1 ОГЛЯД СИСТЕМ-АНАЛОГІВ ТА ТЕХНОЛОГІЙ ЇХ РОЗРОБКИ

1.1 Особливості використання чат-боту для оформлення замовлень з доставки їжі

Використання різноманітних чат-ботів у месенджерах стало невід'ємною частиною нашого життя. За їх допомогою, ми можемо робити велику кількість різних дій та дізнаватися значну кількість нової інформації, не залишаючи одного додатку. Перегляд актуальних новин, обробка документів та фото, робота з електронною поштою — це все, та багато іншого дозволяють роботи чат-боти але й це не є межею їх можливостей. Серед особливостей використання телеграм-боту, саме для замовлення їжі, можна виділити:

– зручний доступ – необхідно всього-на-всього скористатися месенджером Телеграм, який є доволі популярним та встановлений на телефон у багатьох людей, після чого можна з легкістю замовити свою улюблену їжу;

– легкість використання – всього за декілька натискань можна зробити замовлення, не відволікаючись від важливих повідомлень та обговорень. Необхідно обрати з меню страви, які зацікавили, після чого всього за декілька кліків оформити замовлення та очікувати на його доставку;

– інтуїтивно-зрозумілий інтерфейс – в основному, в чат-ботах для навігації та виконання операцій використовують різного типу кнопки з коротким але змістовним описом того, що дозволяє зробити саме ця кнопка. Таким чином досягається зручність, легкість та швидкість використання чат-боту;

– персоналізація та історія замовлень – зазвичай, при оформленні замовлення зберігається особиста інформація про користувача, наприклад, адреса доставки, номер телефону та інше, завдяки чому в подальшому оформлення замовлення стає значно зручнішим, оскільки пропонується використання даних, отриманих раніше. Також наявна можливість перегляду попередніх замовлень, що спрощує повторні замовлення та надає зручну взаємодію з ботом.

Таким чином, можна підсумувати, що використання чат-боту для замовлення їжі з ресторану є актуальним та зручним рішенням, яке дозволяє не відволікаючись від важливих справ та обговорень, швидко та просто зробити замовлення своїх улюблених страв.

1.2 Огляд додатків для ведення заміток

Для ефективної розробки телеграм-боту призначеного для замовлення їжі необхідно дослідити та визначити функціонал, який має бути доступним в ньому. Найкращим рішенням для цього є перегляд та аналіз інших чат-ботів, які пропонують схожі послуги. За допомогою цього ми зможемо встановити їх особливості, переваги та недоліки, що стане доволі корисним при розробці власного варіанту. Здійснивши пошук в інтернеті, знаходимо боти, які відповідають поставленим умовам та будуть використовуватися як аналоги в даній роботі:

– Чат-бот «М’ясторія»;
– Чат-бот «Roll Club».

Наступним кроком, ми досліджуємо та аналізуємо оформлення, функціонал та особливості роботи цих чат-ботів, що в майбутньому може стати в пригоді при проектуванні та розробці власного боту.

1.2.1 Чат-бот «М’ясторія»

Чат-бот «М’ясторія» створений для ресторану «М’ясторія», він є зручним у використанні ботом, який представляє для замовлення різні страви з меню ресторану.

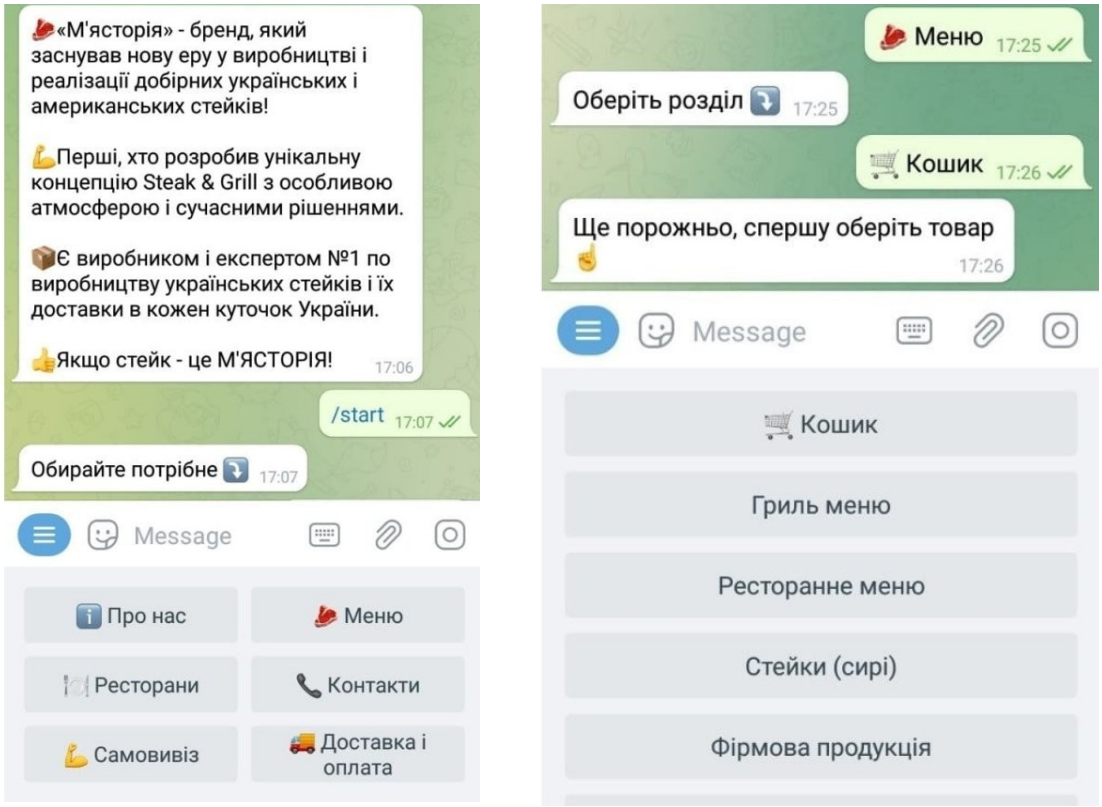

Рисунок 1.1 – Знімки екрану чат-боту «М’ясторія»

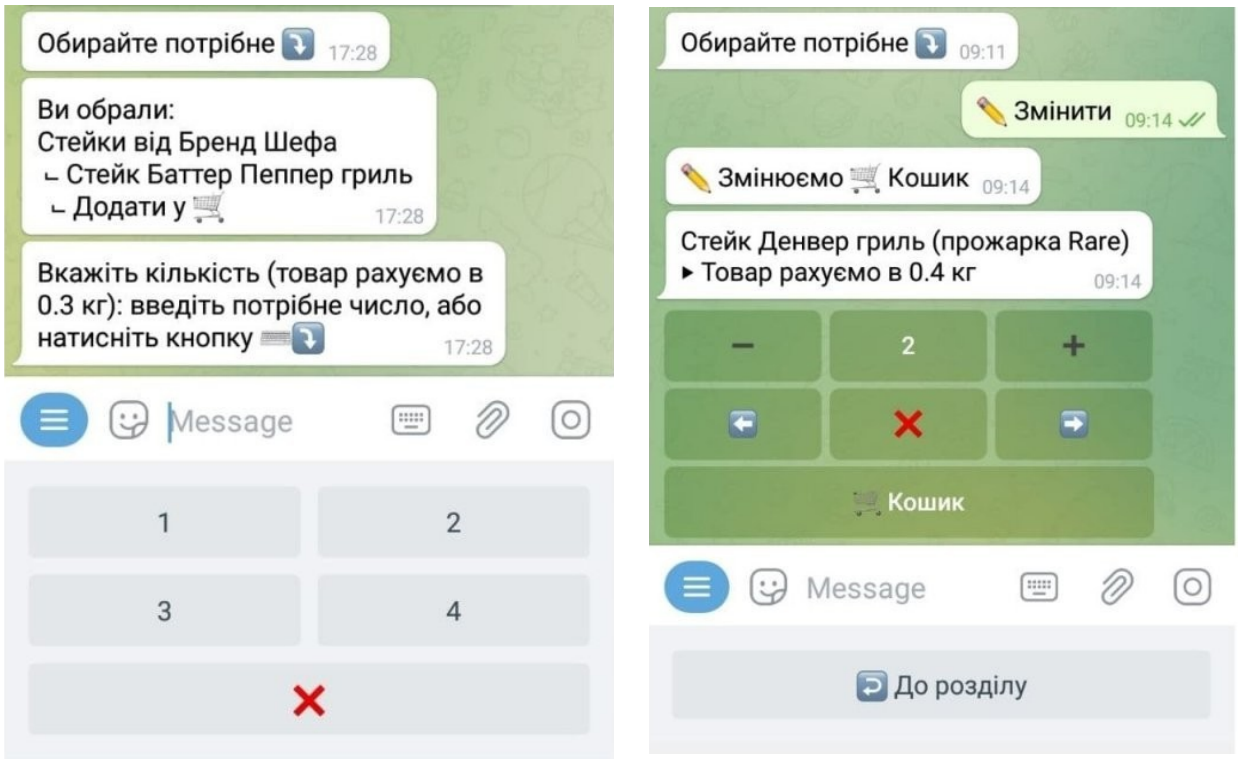

Рисунок 1.2 – Знімки екрану чат-боту «М’ясторія»

Основний функціонал чат-бота:

- перегляд меню;
- додавання страви до кошика;
- зміна кількості страв у кошику;
- перегляд страв у меню за різними категоріями;
- оформлення замовлення;
- оплата замовлення;
- отримання додаткової інформації про замовлення;
- можливість зміни замовлення;
- можливість скасування замовлення;
- можливість оформлення замовлення на зазначений час;
- можливість зв’язку з менеджером закладу.

Таким чином, ми можемо побачити, що чат-бот закладу «М’ясторія» має доволі зрозумілий інтерфейс, а також основний функціонал, який має передбачати кожен бот для замовлення їжі.

1.2.2 Чат-бот «Roll Club»

Чат-бот «Roll Club» створений для ресторану японської кухні, з відповідною назвою. Бот містить в собі наступний функціонал:

- перегляд меню закладу;
- перегляд страв за різними категоріями;
- додавання страв до кошика;
- перегляд збереженої інформації про користувача;
- оформлення замовлення;
- перегляд попередніх замовлень;
- зміна кількості страв у кошику;
- можливість зворотнього зв'язку з менеджером закладу;
- оплата замовлення;

– можливість вибору методу оплати;
– можливість оформлення доставки на вказаний час;
– можливість скасування замовлення.

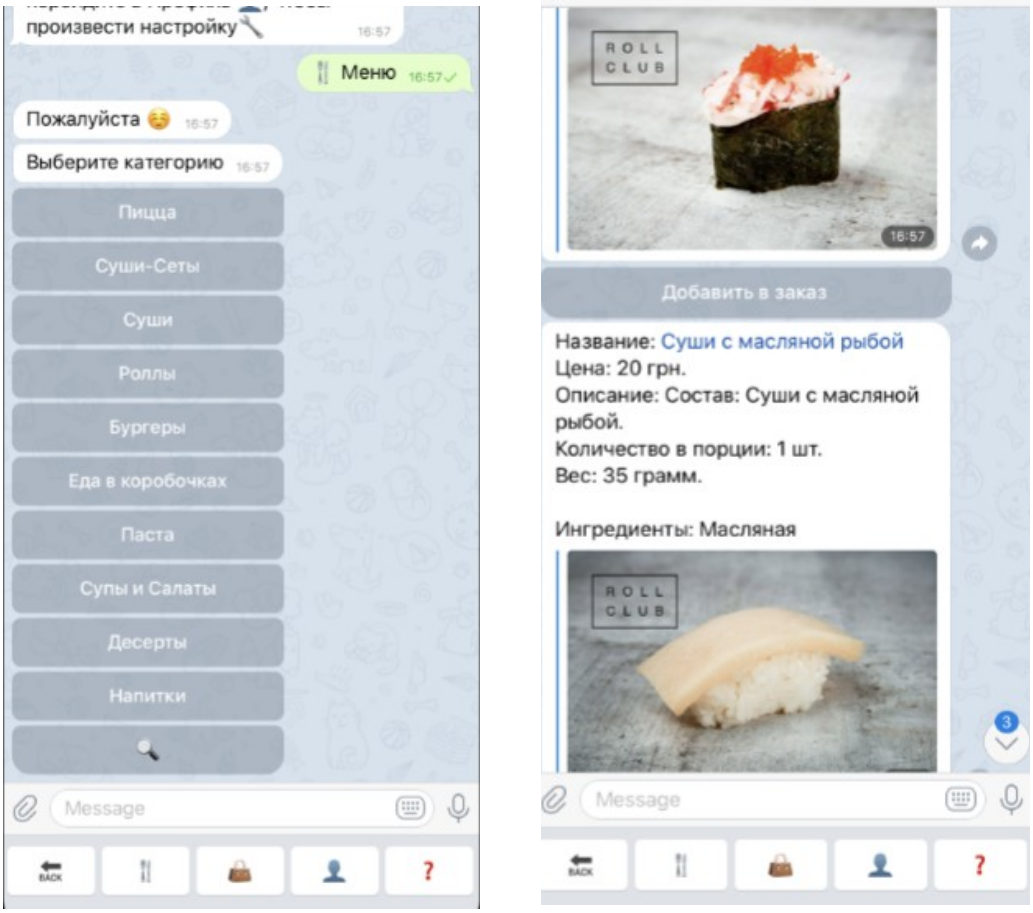

Рисунок 1.3 – Знімки екрану чат-боту «Roll Club»

Таким чином, ми можемо побачити, що чат-бот закладу японської кухні «Roll Club», на вигляд є доволі простим варіантом реалізації чат-боту. Він має максимально простий інтерфейс, а його основний функціонал відповідає основним вимогам до такого роду ботів.

1.3 Формування вимог до основних функцій телеграм-бота

Провівши дослідження та аналіз аналогічних телеграм-ботів для замовлення їжі, можна сформувати функціонал та вимоги до телеграм-бота, який створюється.

Основний функціонал телеграм-бота для замовлення їжі має включати наступні можливості:

– перегляд меню закладу – користувач переглядає різні категорії меню, страви з нього, таким чином вирішуючи що він бажає замовити. Бот має надавати зручне та зрозуміле для користувача представлення інформації про страви;

– додавання страви до кошику – користувач, обравши бажану страву, додає її до кошику, з якого згодом зможе оформити замовлення;

– оформлення замовлення – користувач, сформувавши повний перелік страв, які бажає замовити, може оформити замовлення, вказавши всі необхідні дані, такі як адреса доставки, номер телефону для зв’язку, додаткова інформація до замовлення тощо;

– зміна кількості страв в кошику та видалення страв з кошику – користувач може вирішити додати ще декілька страв до кошику, або навпаки видалити зайві страви, тому чат-бот має містити цей функціонал, який є невід’ємним;

– перегляд попередніх замовлень – всі попередні замовлення в боті зберігаються в базі даних, тому отримати інформацію про них не стає проблемою. Тому користувач може переглянути свої попередні замовлення, щоб він мав можливість зручніше та швидше роботи повторні замовлення;

– збереження даних під час оформлення замовлення – ми вже з’ясували, що під час оформлення замовлення користувачу необхідно вказати свої особисті дані, тобто адресу доставки замовлення, номер телефону для зв’язку, ім’я отримувача тощо. Тому для користувача буде значно зручніше мати можливість, під час оформлення замовлення, зазначити ті самі дані, що й минулого разу, щоб повторно не вводити одну й ту саму інформацію;

– вказання додаткової інформації при оформленні замовлення – користувач, при оформленні замовлення, може вказати побажання до замовлення, або додати деталі стосовно його доставки;

– зручний метод оплати – під час оформлення замовлення, чат-бот пропонує користувачу обрати зручний для нього спосіб оплати замовлення.

1.4 Огляд інформаційних технологій для розробки телеграм-бота

1.4.1 Фреймворк Spring

Spring — це фреймворк із відкритим вихідним кодом. Він спрощує та прискорює програмування на Java, а також є дуже популярним Java-фреймворком. Основні особливості Spring Framework можуть бути використані будь-яким застосунком Java, але також є розширення для створення веб-додатків на платформі Java EE. [2]

Особливості використання Spring Framework:

– фреймворк Spring дозволяє швидко реалізувати нові можливості у робочому середовищі завдяки мікросервісам, які можна розвивати незалежно один від одного;

– код можна розгорнути в будь-якій хмарі, а сервіси можна пов’язувати один з одним і масштабувати на будь-якій платформі;

– spring надає фреймворки для розробки швидких і безпечних застосунків, які можуть підключатися до будь-якого сховища даних;

– spring можна використовувати для розробки програм, які масштабуються за потребою залежно від вимог;

– програми на Spring можуть обробляти потокові дані в режимі реального часу;

– spring можна використовувати для автоматизації завдань та офлайн-обробки даних у зручний час.

Підсумовуючи, можемо сказати, що Spring Framework спрощує та прискорює розробку застосунків на Java та складається з безлічі міні-сервісів, бібліотек і розширень. Його розширення Spring Boot, яке здебільшого й буде використовуватися при розробці телеграм-бота, ще більше прискорює та спрощує розробку застосунків, хоча теоретично й дає менше гнучкості через автоматичні налаштування.

1.4.2 Telegram API

Telegram API є бібліотекою, яка дозволяє розробникам взаємодіяти з Telegram-платформою у своїх Java-проектах. Вона надає доступ до різноманітних функцій, таких як надсилання повідомлень, створення чат-груп, керування користувачами і ботами, отримання оновлень тощо. Це потужний інструмент, який дозволяє розробникам створювати різноманітні телеграм-боти і додатки, які інтегруються з цією платформою. Документація Telegram API надає детальну інформацію про доступні методи, класи і приклади використання, що допомагає розробникам швидко почати роботу з цією бібліотекою. Крім базової функціональності, Telegram API також підтримує розширені можливості, такі як робота з медіафайлами, створення кнопок і інтерактивних меню, робота з клавіатурами, обробка вхідних команд та багато іншого. Це дозволяє розробникам створювати ботів і додатки зі складнішою логікою, включаючи можливість взаємодії з користувачами через вкладені елементи і варіанти відповідей. Загалом, Telegram API є потужним інструментом, який допомагає розробникам створювати різноманітні та функціональні додатки для платформи Telegram.

1.5 Висновки до першого розділу

В першому розділі курсової роботи був проведений огляд та аналіз телеграм-ботів для замовлення їжі, виділені особливості їх роботи, функціонал тощо.

Було розглянуто доцільність розробки чат-боту з можливістю замовлення доставки їжі. Було визначено, що через значну інтеграцію месенджерів у наше повсякденне життя, можливість замовлення своїх улюблених страв, не відволікаючись від важливих чатів та спілкувань, є доволі доцільним рішенням, оскільки відпадає необхідність використання сторонніх застосунків або веб-сайтів для цього.

Після чого було проведено дослідження та аналіз аналогічних чат-ботів для доставки їжі, з чого було встановлено їх основний функціонал, а також особливості їх створення та використання. Завдяки цьому, були сформовані та детально описані основні вимоги до власного телеграм-бота.

Насамкінець, був проведений огляд інформаційних технологій для розробки телеграм-бота замовлення їжі. В якості яких виступають Spring фреймворк, а також Telegram API. Фреймворк Spring спрощує та прискорює розробку застосунків на Java, а Telegram API надає зручний доступ для роботи з платформою Telegram.

2 ПРОЕКТУВАННЯ ТЕЛЕГРАМ-БОТА ДЛЯ ЗАМОВЛЕННЯ ЇЖІ

2.1 Мета та задачі телеграм-бота

Мета телеграм-бота для замовлення доставки їжі полягає в спрощенні та збільшенні зручності для користувача в процесі замовлення улюблених страв з ресторану. Головною метою є надання зручних послуг замовлення їжі, щоб будь-який користувач Telegram міг всього за декілька хвилин обрати бажані страви з меню та оформити замовлення, при цьому не відволікаючись від важливих справ, чатів та спілкувань.

Як було з’ясовано раніше, задачі чат-бота для замовлення їжі можуть включати:

– перегляд меню закладу – телеграм-бот має надавати користувачам зручне та зрозуміле представлення меню, за допомогою якого, користувач може легко та швидко обрати бажані страви;

– додавання страви до кошику – після обрання бажаних страв, користувач повинен мати можливість додати їх до кошику для подальшого оформлення замовлення;

– оформлення замовлення – після додавання обраних страв до кошика, користувач повинен мати можливість оформити замовлення, вказавши необхідні дані для цього;

– зручний метод оплати – під час оформлення замовлення користувач повинен мати можливість обрати зручний для нього метод оплати замовлення;

– перегляд попередніх замовлень – користувач повинен мати можливість переглядати свої попередні замовлення, це дозволить йому легше роботи схожі або повторні замовлення;

– зміна кількості та видалення страв з кошику – бот має надати користувачу можливість, при необхідності, змінювати кількість страв та видаляти їх з кошику, щоб забезпечити більшу зручність для користувача;

– збереження даних при оформленні замовлення – для більшої зручності користувача, при оформленні нового замовлення, телеграм-бот має запропонувати йому використати дані з попередніх замовлень, що дозволить прискорити процес та позбавить необхідності повторно вводити однакові дані.

Таким чином, використання телеграм-бота для замовлення їжі має зменшити витрати часу на цей процес в інших застосунках, а також надати зручний та зрозумілий у використанні сервіс замовлення їжі, який розташований у звичайному месенджері.

2.2 Визначення функціональних вимог до телеграм-бота

Визначення функціональних вимог є важливим та невід'ємним етапом у процесі створення телеграм-бота для замовлення їжі. Функціональні вимоги визначають, які можливості, опції та функції мають бути реалізовані в чат-боті. Завдяки цьому встановлюється чітка спрямованість розробки та при цьому зменшується кількість непорозумінь.

Також, визначення функціональних вимог дозволяє сфокусуватися на потребах та вимогах користувачів. Завдяки їх визначенню стає зрозуміло, які функції та можливості є найбільш корисними для користувачів.

Крім того, функціональні вимоги є основою для комунікації між усіма учасниками проекту, оскільки вони сприяють розумінню того, що саме потрібно реалізувати та що очікується в результаті.

Таким чином, встановлення функціональних вимог дозволяє визначити загальний обсяг роботи, а також ефективно планувати робочий час, бюджет та необхідні ресурси для проекту.

Для того, щоб визначити як користувачі будуть використовувати телеграм-бота та які вимоги можна для нього сформувати, була створена діаграма сценаріїв для нього (рис. 2.1). Діаграма сценаріїв UML (Unified Modeling Language) - це спосіб графічно показати функції системи та їх зв'язок з користувачами. На

діаграмі зображені актори, сценарії та їх взаємодія. Ця діаграма дозволяє проаналізувати потреби акторів, можливості системи та визначити вимоги до неї. Також ця діаграма наочно демонструє розробникам основні можливості та функції системи.

Телеграм-бот для замовлення їжі має лише одного актора, і це «користувач». Користувач має можливість переглядати меню, обирати страви з нього, отримувати про них більш детальну інформацію, додавати їх до кошика та оформлювати замовлення для доставки, тобто він має доступ до повного функціонала бота.

Рисунок 2.1 – Діаграма варіантів використання телеграм-бота

2.3 Формування користувацьких історій телеграм-бота

За допомогою спроектованої діаграми варіантів використання телеграм-бота, можна визначити такі користувацькі історії чат-бота для замовлення їжі.

US1 Як користувач, я можу переглядати меню, щоб обрати страви для замовлення.

Сценарій користувацької історії:

– користувач відкриває спілкування з ботом в месенджері;

– користувач натискає на кнопку «Меню»;

– після цього на екрані з’являються різні категорії меню;

– обравши необхідну категорію, з’являються страви, що належать до цієї категорії;

– обравши страву, користувач може переглянути інформацію про неї.

US2 Як користувач, я можу переглядати детальну інформацію про страву, щоб краще розуміти чи варто її замовляти.

Сценарій користувацької історії:

– після обрання категорії страв в меню, на екрані з’являється перелік страв, що належать до цієї категорії;

– натиснувши на кнопку з назвою страви, користувач може переглянути детальну інформацію про неї: опис, фото, ціна тощо.

US3 Як користувач, я можу додати обрану страву до кошику, щоб мати змогу оформити замовлення.

Сценарій користувацької історії:

– користувач обрав страву з меню та отримав детальну інформацію про неї;

– користувач натискає на кнопку «Додати до кошику» під описом страви;

– Обрана страва додається до кошику користувача.

US4 Як користувач, я можу переглянути вміст кошику.

Сценарій користувацької історії:

– після додавання страви до кошику, з’являється кнопка «Переглянути кошик»;

– натиснувши на неї користувач може переглянути страви, які знаходяться в його кошику.

Інший можливий сценарій користувацької історії:

– натиснувши на кнопку «Повернутися в головне меню», користувач переходить до головного меню;

– в головному меню користувач натискає на кнопку «Кошик»;

– на екран виводиться вміст кошику користувача.

US5 Як користувач, я можу оформити замовлення.

Сценарій користувацької історії:

– після перегляду кошику, користувач натискає на кнопку «Оформити замовлення»;

– користувач вводить свої особисті дані, такі як ім’я, адреса доставки, номер телефону для зв’язку;

– після чого на екрані з’являється повідомлення з переліком обраних страв та введеною особистою інформацією, з метою перевірки коректності інформації;

– користувач натискає кнопку «Оформити замовлення»;

– замовлення оформлено та відправляється на обробку.

US6 Як користувач, я можу обрати спосіб оплати замовлення, щоб оплатити зручним для мене методом.

Сценарій користувацької історії:

– під час оформлення замовлення, користувачеві пропонується обрати зручний для нього метод оплати замовлення;

– натиснувши на кнопку «Картка», користувач обирає сплату замовлення банківською картою;

– натиснувши на кнопку «Готівка», користувач обирає сплату замовлення готівкою.

US7 Як користувач, я можу переглядати історію свої замовлень та стан свого замовлення, щоб мати можливість повторно робити замовлення або відстежувати на якому етапі готовності знаходиться поточне замовлення.

Сценарій користувацької історії:

– в головному меню бота, користувач натискає на кнопку «Попередні замовлення»;

– серед запропонованих номерів замовлень, клієнт обирає потрібний, натиснувши на нього;

– на екран виводиться інформація про обране замовлення користувача.

US8 Як користувач, я можу вказати додаткову інформацію про доставку, щоб уточнити деяку інформацію стосовно замовлення або його доставки.

Сценарій користувацької історії:

– при оформленні замовлення, користувачу пропонується надати додаткову інформацію до замовлення;

– користувач натискає на кнопку «Пропустити» та пропускає цей етап;

– або користувач вказує додаткову інформацію та переходить до підтвердження замовлення.

US9 Як користувач, я можу здійснити пошук страви за назвою, щоб швидше знайти необхідну страву.

Сценарій користувацької історії:

– користувач, знаходячись в меню, вводить назву страви, інформацію про яку бажає переглянути;

– після чого виводиться інформація про обрану страву та з’являється можливість додати її до кошику;

– якщо назва страви введена некоректно або страва з такою назвою не наявна в боті, виводиться повідомлення про помилку.

US10 Як користувач, я можу змінювати кількість страв в кошику, щоб отримати необхідну для мене кількість страв.

Сценарій користувацької історії:

– користувач, переглянувши вміст кошику, натискає на кнопку «Внести зміни до кошику»;

– користувач натискає на кнопку «Змінити кількість страв»;

– з наведеного переліку користувач обирає страву, кількість якої бажає змінити;

– обравши страву, виводиться інформація про неї, та її кількість в кошику;

– натискаючи на кнопку «+», користувач може додати ще одну страву до кошику;

– натискаючи на кнопку «–», користувач може зменшити кількість цієї страви в кошику.

US11 Як користувач, я можу видалити страву кошику, щоб не отримати непотрібну для мене страву.

Сценарій користувацької історії:

– користувач, переглянувши вміст кошику, натискає на кнопку «Внести зміни до кошику»;

– користувач натискає на кнопку «Видалити страву з кошику»;

– з наведеного переліку користувач обирає страву, яку бажає видалити;

– обрана страва видаляється з кошику.

Таким чином, було сформовано основні користувацькі історії для телеграм-боту замовлення їжі. Розроблені користувацькі історії допоможуть при проектуванні та розробці бота.

2.4 Визначення нефункціональних вимог до телеграм-бота

Серед нефункціональних вимог до телеграм-бота можна виділити лише наявність гарного інтернет з’єднання, щоб процес спілкування з ботом міг проходити швидко та без усіляких затримок. Ну і звісно, користувач має встановити на свій пристрій месенджер Telegram, та бути зареєстрованим в ньому.

2.5 Ідентифікація архетипу телеграм-бота

Телеграм-бота для замовлення їжі, який розробляється, можна віднести до комерційних чат-ботів, тобто до ботів, які, використовуючи платформу Telegram, надають різноманітні послуги користувачам, такі як замовлення товарів, обробка платежів тощо.

2.6 Проектування користувацького інтерфейсу телеграм-бота

На основі визначених функціональних та нефункціональних вимог до телеграм-бота замовлення їжі, а також користувацьких історій до нього, можна приступити до проектування користувацького інтерфейсу бота. Для цього необхідно створити макети ключових моментів взаємодії користувача з чат-ботом, визначити можливі сценарії цих взаємодій, а також скласти короткий опис кожного макету.

На рисунку 2.2 зображено макет головного меню телеграм-бота. Воно складається з трьох кнопок: Меню, Кошик та Попередні замовлення. За допомогою цієї клавіатури, користувач може перейти до перегляду меню, кошику або свої попередніх замовлень.

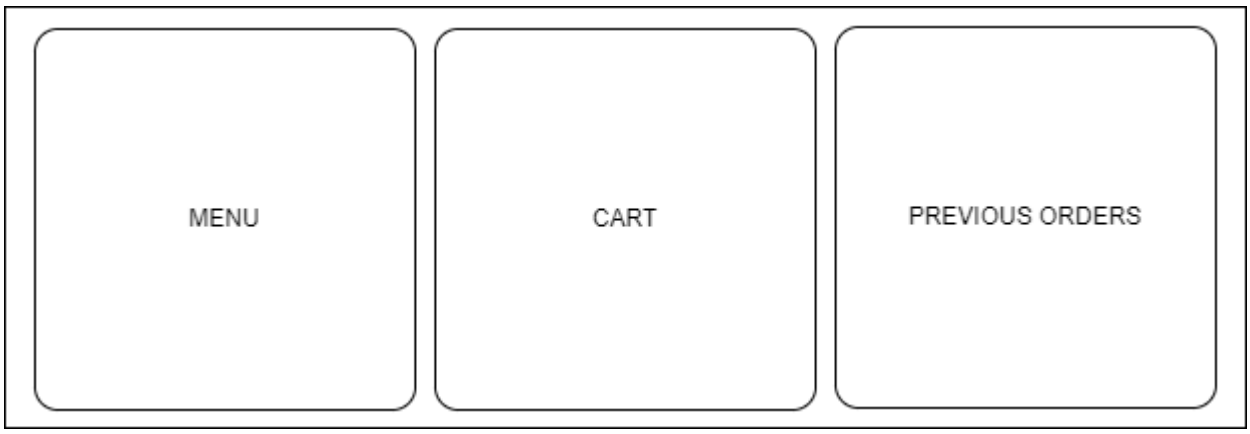

Рисунок 2.2 – Макет клавіатури головного меню бота

На рисунку 2.3 зображено макет вибору категорій страв, який доступний після того, як перейти до перегляду меню. Макет складається з кнопок для кожної категорії меню, а також кнопки повернення до головного меню. За допомогою цієї клавіатури можна обрати категорію, яка зацікавила, та перейти до перегляду страв цієї категорії.

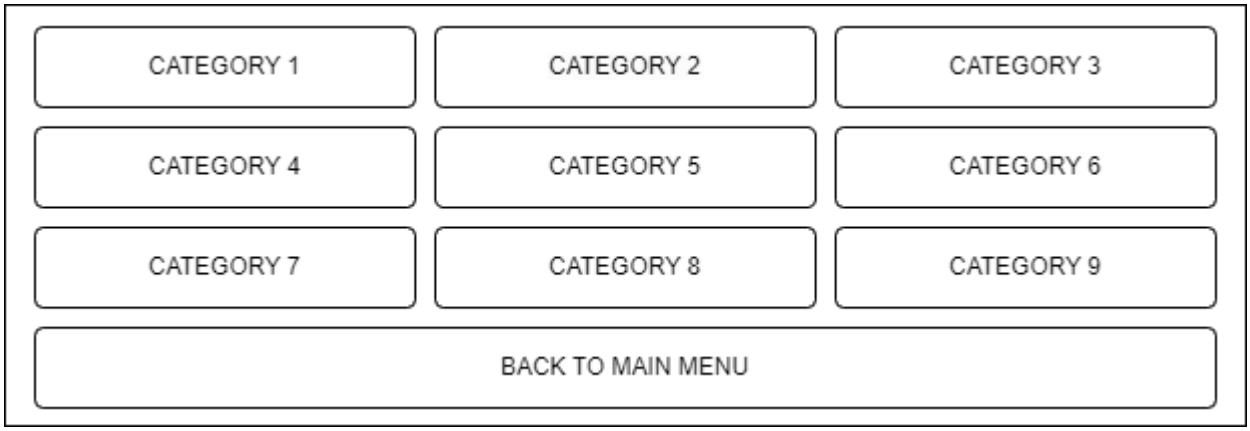

Рисунок 2.3 – Макет клавіатури вибору категорій меню

На рисунку 2.4 зображено макет вибору страви, він доступний після обрання категорії меню. Макет складається з кнопок для кожної зі страв, що належить до обраної категорії, а також кнопки повернення до перегляду меню. За допомогою цієї клавіатури можна обрати страву, яка зацікавила, з метою перегляду детальної інформації про неї та подальшого додавання до кошику.

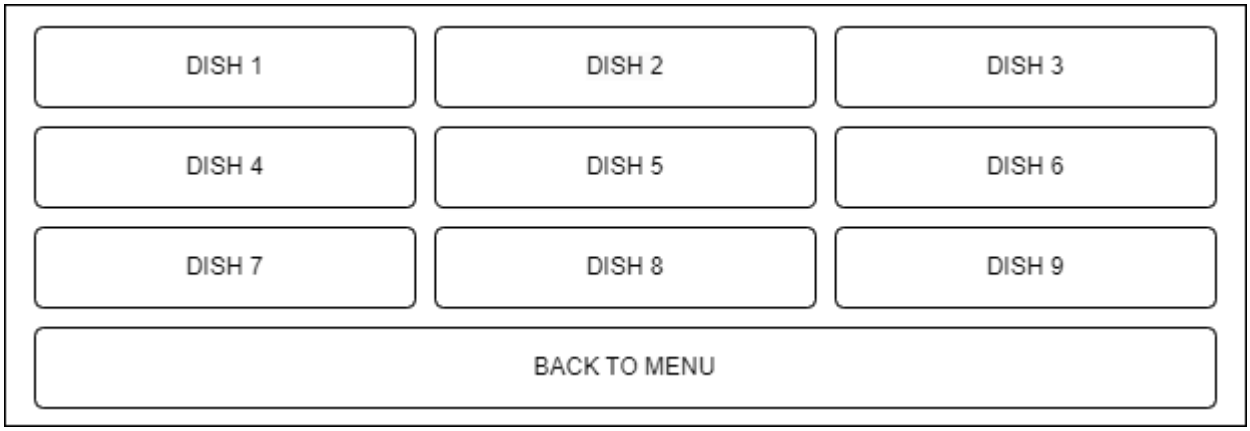

Рисунок 2.4 – Макет клавіатури вибору страви з меню

На рисунку 2.5 зображено макет повідомлення, яке отримує користувач після вибору страви, яка його зацікавила. Макет складається з зображення страви, її детального опису, який включає ціну, вагу, опис тощо, а також кнопки додавання цієї страви до кошику. З цього повідомлення користувач отримує всю необхідну інформацію про страву, що допомагає йому зрозуміти, чи бажає він замовити її, а також отримує можливість додати цю страву до кошику.

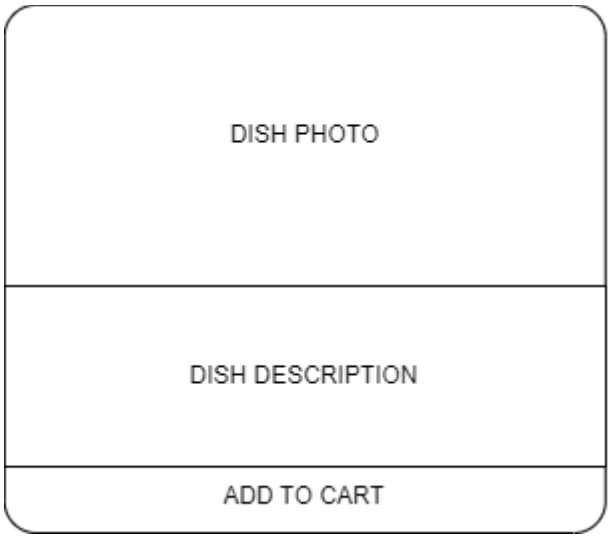

Рисунок 2.5 – Макет повідомлення з інформацією про страву

На рисунку 2.6 зображено макет повідомлення, яке отримує користувач при прийнятті рішення про зміну кількості страв в кошику. Макет складається з зображення страви, її опису, в якому також зазначається її кількість в кошику, та кнопок зменшення та збільшення кількості обраної страви. Таким чином, з цього повідомлення користувач може остаточно переконатися в тому чи потрібно йому змінювати кількість цієї страви, а також, при необхідності, додати ще одну таку страву або видалити зайву.

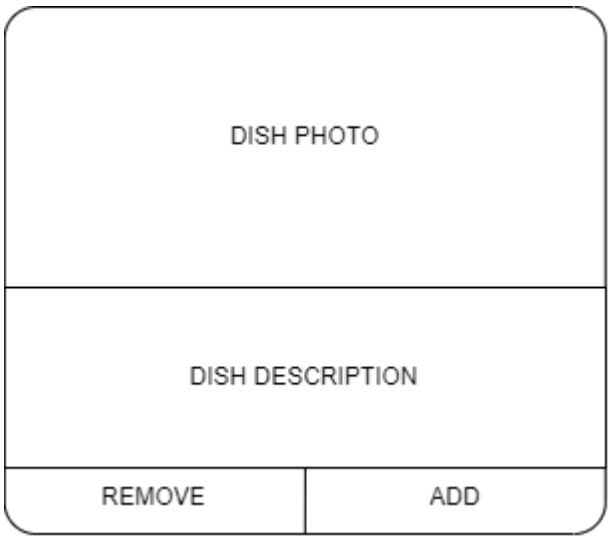

Рисунок 2.6 – Макет повідомлення для зміни кількості страв в кошику

На рисунку 2.7 зображено макет повідомлення, яке отримує користувач при оформленні замовлення, а саме перед його остаточним підтвердженням. Макет складається з опису замовлення, куди включена надана користувачем інформація про доставку замовлення та перелік обраних страв, а також під повідомленням розташована кнопка підтвердження замовлення. Таким чином, це повідомлення надає можливість користувачу перевірити коректність замовлення та підтвердити його.

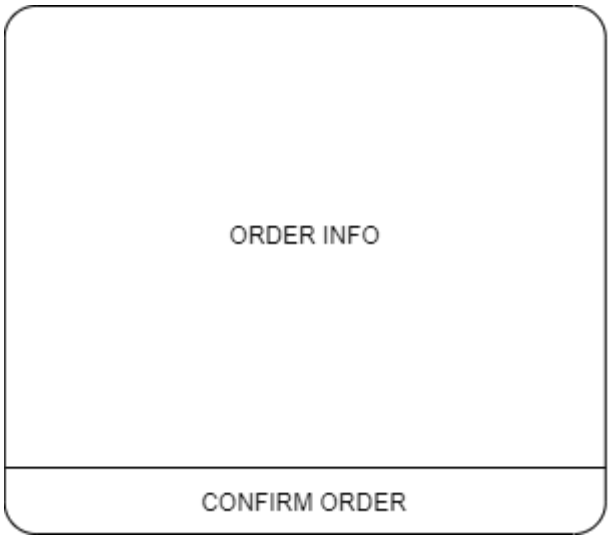

Рисунок 2.7 – Макет повідомлення про підтвердження замовлення

Таким чином, було розроблено макет основних сценаріїв взаємодії користувача з телеграм-ботом, за допомогою якого розробка бота буде більш спрямованою, оскільки вже буде розуміння того, який остаточний вигляд має бути у чат-бота. Інші сценарії взаємодії будуть побудовані на цих основах, наприклад, меню налаштування кошику буде мати схожу клавіатуру з головним меню, зміниться лише функціонал кнопок.

2.7 Висновки до другого розділу

У другому розділі цієї курсової роботи було проведено проектування телеграм-боту для замовлення їжі.

В процесі проектування було визначено головну мету та задачі, які має виконувати телеграм-бот, що знаходиться в розробці. Спираючись на ці дані, було встановлено функціональні та нефункціональні вимоги до телеграм-боту. Було розроблено користувацькі історії зі сценаріями дій до кожної з них, а також діаграму варіантів використання телеграм-бота.

З метою проектування інтерфейсу користувача були розроблені макети основних сценаріїв взаємодії користувача з телеграм-ботом, що в майбутньому допоможе при розробці бота та в реалізації зручного та інтуїтивно-зрозумілого інтерфейсу для користувача.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТЕЛЕГРАМ-БОТА ДЛЯ ЗАМОВЛЕННЯ ЇЖІ

3.1 Структура серверного програмного проекту

Пакет програмного проекту Spring для телеграм-бота замовлення їжі може складатися з різних компонентів та класів. Визначимо компоненти та класи, які буде необхідно реалізувати при розробці.

Головним класом проекту є FoodDeliveryBot, він відповідає за взаємодію між користувачем та ботом. Його основними задачами є обробка повідомлень від користувача, їх аналіз, формування та надсилання відповідей на повідомлення та запити користувача, а також створення інтерфейсу у вигляді різних клавіатур та кнопок під повідомленнями для користувача.

До моделей проекту, тобто сутностей, які використовуються в системі та, зазвичай, мають зв’язок з базою даних, можна віднести два класи: Dish та Order. Клас Dish є сутністю страви, дані про яку отримуються з бази даних та після чого оброблюються ботом. Клас Order є сутністю замовлення, телеграм-бот формує дані про нього та зберігає їх у базу даних для подальшої обробки менеджерами закладу. Також, дані про замовлення можуть бути отримані з бази даних, наприклад, для надання можливості користувачу переглядати попередні замовлення.

До створених моделей необхідно створити репозиторії, саме вони відповідають за збереження та отримання даних з бази даних. В нашому випадку необхідно створити два репозиторії: DishesRepository – репозиторій страв, OrdersRepository – репозиторій замовлень.

Клас проекту Cart є кошиком користувача. До основного функціоналу цього класу належить додавання та видалення страв з кошику, отримання інформації про те, які страви знаходяться в кошику, формування списку страв для оформлення замовлення, а також зміна кількості страв в кошику.

Також, проект містить такі класи як BotInitializer та BotConfig, за допомогою цих класів виконується налаштування телеграм-бота для його

коректної роботи з сервісом Telegram. Крім того, проект містить конфігураційні файли, за допомогою яких відбувається конфігурування та налаштування фреймворку Spring, а також зв’язку з базою даних, що сприяє коректній роботі всіх елементів проекту.

3.2 Діаграма класів телеграм-бота

Діаграма класів є одним з видів UML-діаграм, яка дозволяє простим та зрозумілим чином показати класи, інтерфейси та їх взаємодію у системі. Діаграма класів дозволяє наочно представити структуру системи та зв’язки між її складовими. Діаграма класів телеграм-бота для замовлення їжі наведена на рисунку 3.1.

3.3 Керування вихідним кодом телеграм-бота

Система контролю версій – це необхідний інструмент для розробки будь-якого програмного проекту, в тому числі і телеграм-бота замовлення їжі. Система контролю версій дозволяє зберігати історію всіх змін, які вносилися під час розробки. Це надає можливість, у разі потреби, повернутися до попереднього стану проекту, повернувшись до попередньої версії. Також, система контролю версій допомагає керувати версіями бота, що полегшує роботу з ним, а також підтримує проект в актуальному стані.

Основні метрики репозиторію телеграм-бота наведені в таблиці 3.1.

Таблиця 3.1 – Метрики керування програмним кодом телеграм-бота

Кількість комітів	Кількість pull-реквестів	Кількість гілок у репозиторії
12	0	1

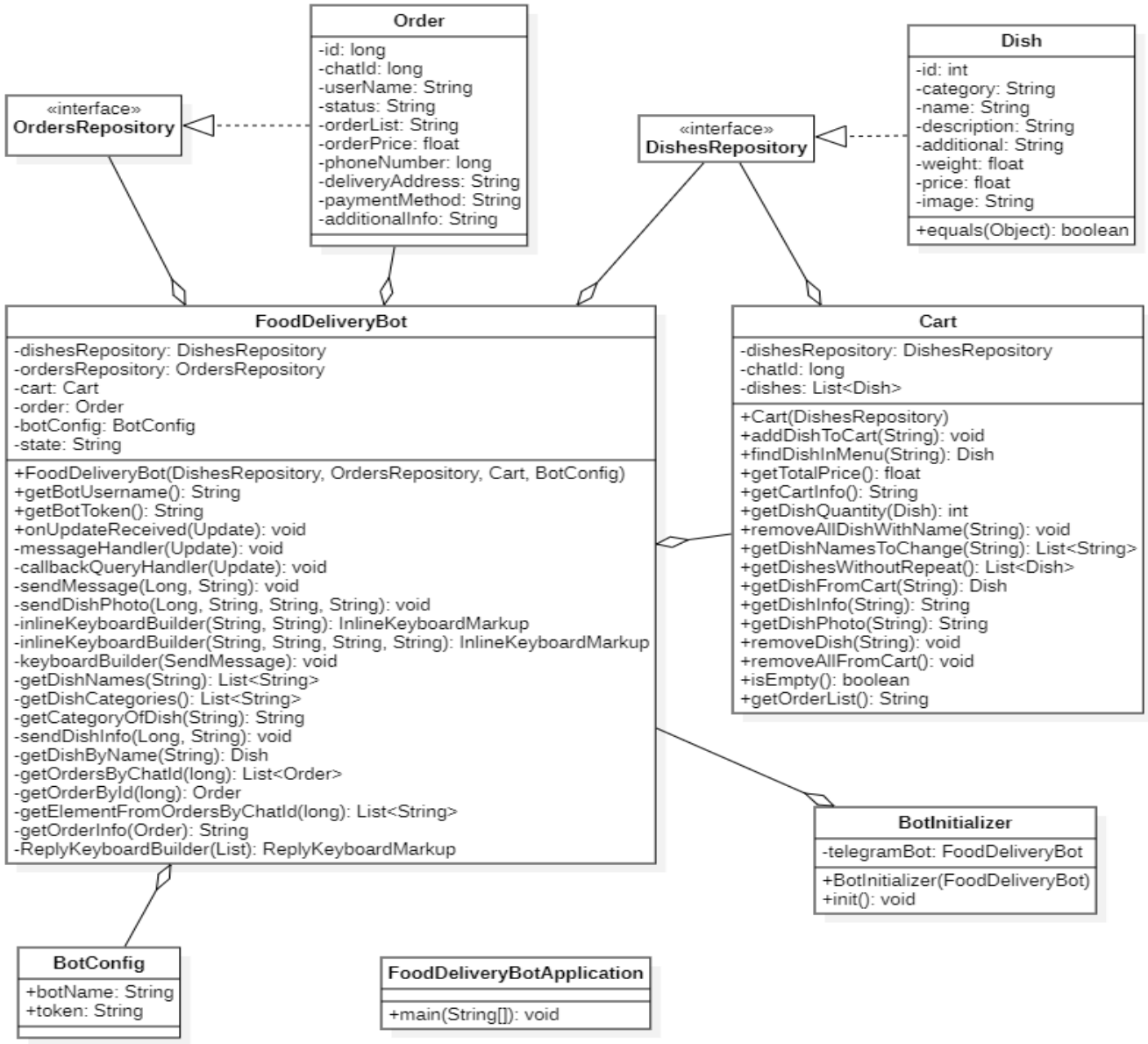

Рисунок 3.1 – Діаграма класів телеграм-бота для замовлення їжі

3.4 Функціональне тестування розробленого телеграм-бота

Функціональне тестування телеграм-бота для замовлення їжі є невід'ємним процесом його розробки, оскільки завдяки тестуванню можна перевірити коректність роботи розроблених функцій телеграм-бота.

Функціональне тестування дозволяє переконатися в правильності та коректності виконання розроблених функцій бота, а також бути впевненим в тому, що весь функціонал чат-бота працює саме так, як очікувалося, і не стається помилок, які перешкоджають роботі сервісу.

Окрім того, завдяки функціональному тестуванню виявляються помилки в роботі бота, які можуть перешкоджати коректній роботі чат-бота, негативно впливаючи на досвід користувачів. Тим самим, після виявлення цих помилок, відбувається процес їх аналізу та виправлення, щоб надати користувачам надійний та стабільний сервіс.

Для функціонального тестування необхідно скласти протокол тестування. Протокол тестування - це документ, який описує кроки, процедури та результати тестування програмного продукту, системи або її окремого компоненту. Протокол тестування призначений для системного документування процесу тестування та отриманих результатів, з метою забезпечення максимальної об’єктивності результатів.

Для функціонального тестування телеграм-бота для замовлення їжі був розроблений наступний протокол тестування.

TC1 Тест-кейс для відображення категорій меню:

- відкрити чат з ботом в месенджері;
- натиснути на кнопку «Меню»;
- переконатися в тому, що всі додані категорії меню відображаються.

TC2 Тест-кейс для відображення страв в меню:

- відкрити чат з ботом в месенджері;
- натиснути на кнопку «Меню»;
- натиснути на кнопку з категорією страв, яка цікавить;
- переконатися в тому, що всі страви, які належать до обраної категорії, відображаються коректно.

TC3 Тест-кейс для відображення інформації про страву:

- відкрити чат з ботом в месенджері;
- натиснути на кнопку «Меню»;
- натиснути на кнопку з категорією страв, яка цікавить;
- натиснути на кнопку з назвою страви, для отримання інформації про неї;

– переконатися в тому, шо відображення інформації про страву відбувається коректно.

ТС4 Тест-кейс для додавання страви до кошика:

– відкрити чат з ботом в месенджері;

– натиснути на кнопку «Меню»;

– натиснути на кнопку з категорією страв, яка цікавить;

– натиснути на кнопку з назвою страви, яку бажаємо замовити;

– натиснути на кнопку «Додати до кошику», під описом страви;

– натиснути на кнопку «Переглянути кошик»;

– переконатися в тому, що обрана страва була додана до кошику.

ТС5 Тест-кейс для перегляду вмісту кошику:

– після додавання товарів до кошику, натиснути на кнопку «Переглянути кошик»;

– переконатися в коректності інформації про вміст кошика.

ТС6 Тест-кейс для видалення страви з кошика:

– після додавання страви до кошика, натиснути кнопку «Переглянути кошик»;

– в меню кошика натиснути на кнопку «Внести зміни до кошика»;

– натиснути на кнопку «Видалити страву з кошику»;

– натиснути на кнопку з назвою страви, яку бажаємо видалити;

– натиснути на кнопку «Переглянути кошик»;

– переконатися в тому, що страва була видалена з кошику.

ТС7 Тест-кейс для зміни кількості страв в кошику:

– після додавання страви до кошика, натиснути кнопку «Переглянути кошик»;

– в меню кошика натиснути на кнопку «Внести зміни до кошика»;

– натиснути на кнопку «Змінити кількість страв»;

– натиснути на кнопку з назвою страви, кількість якої бажаємо змінити;

– натиснути на кнопку «–», тим самим зменшити кількість страв, або натиснути на кнопку «+» для збільшення кількості страв;

– натиснути на кнопку «Переглянути кошик»;

– переконатися в тому, що кількість страв була змінена.

ТС8 Тест-кейс для оформлення замовлення:

– після додавання страв до кошика, натиснути кнопку «Переглянути кошик»;

– в меню кошика натиснути на кнопку «Оформити замовлення»;

– ввести всі необхідні дані, такі як ім’я отримувача, номер телефону для зв’язку та адреса доставки замовлення;

– натиснути на кнопку «Готівка», для обрання оплати замовлення готівкою, або натиснути на кнопку «Картка», для сплати замовлення банківською картою;

– вказати додаткову інформацію до замовлення або натиснути на кнопку «Пропустити», щоб пропустити цей етап;

– перевірити коректність інформації, яка надається ботом перед підтвердженням замовлення;

– натиснути на кнопку «Оформити замовлення»;

– натиснути на кнопку «Перегляд минулих замовлень»;

– натиснути на кнопку з номером створеного замовлення;

– перевірити коректність збереження інформації про замовлення.

ТС9 Тест-кейс для перегляду попередніх замовлень:

– після оформлення замовлення натиснути на кнопку «Перегляд минулих замовлень»;

– натиснути на кнопку з номером створеного замовлення;

– перевірити коректність інформації про створене замовлення.

ТС10 Тест-кейс для застосування підказок під час оформлення замовлення:

– після додавання страв до кошика, натиснути кнопку «Переглянути кошик»;

– в меню кошика натиснути на кнопку «Оформити замовлення»;

– під час прохання ввести особисту інформацію, переконатися в тому, що відображається кнопки з варіантами, які були введені при попередніх замовленнях;

– натиснути на кнопку з одним з варіантів;

– перед підтвердженням замовлення переконатися в тому, що дані відображені коректно.

Таблиця 3.2 – Протокол функціонального тестування телеграм-бота

Номер тест-кейсу	Очікуваний результат	Фактичний результат	Результат тестування
ТС1	Коректне відображення категорій меню	Всі додані категорії меню відображаються коректно	Успішно
ТС2	Коректне відображення страв в меню	Відображаються тільки ті страви, які належать до обраної категорії	Успішно
ТС3	Коректне відображення інформації про страву	Інформація про обрану страву надається користувачу	Успішно
ТС4	Успішне додавання страви до кошику	Страва додається до кошику	Успішно
ТС5	Коректний перегляд вмісту кошика	Отримання вірної інформація про вміст кошика	Успішно
ТС6	Успішне видалення страви з кошика	Страва видаляється з кошика	Успішно
ТС7	Успішна зміна кількості страв	Кількість страв в кошику змінюється	Успішно
ТС8	Успішне оформлення замовлення	Нове замовлення додається до бази даних	Успішно
ТС9	Коректний перегляд попередніх замовлень	Отримана вірна інформація про попередні замовлення	Успішно
ТС10	Виведення підказок під час оформлення замовлення	Під час оформлення замовлення виводиться дані, які були надані при оформленні попередніх замовлень	Успішно

Результати функціонального тестування телеграм-бота замовлення їжі наведені в таблиці 3.2.

З результатів тестування можна побачити, що всі функціональні тести пройдені успішно, з чого можна зробити висновок, що фактична поведінка телеграм-бота відповідає очікуваній поведінці, яка було визначена у функціональних вимогах.

3.5 Інструкція користувача телеграм-бота

З метою забезпечення зручності використання розробленого телеграм-бота для замовлення їжі, необхідно скласти інструкцію користувача, яка включає в себе знімки екранів та пояснювальний текст до кожного з них. Це допоможе користувачам легко орієнтуватися у функціоналі та особливостях створеного бота та отримати успішний досвід використання чат-бота.

На рисунку 3.2 зображено головне меню телеграм-бота, яке зустрічає користувача одразу на початку роботи з ботом. Головне меню складається з трьох кнопок: Меню, Кошик та Перегляд минулих замовлень. За допомогою цього меню відбувається навігація серед основних напрямів: переглядом страв з меню, переглядом вмісту кошика та його зміни, а також переглядом попередніх замовлень користувача.

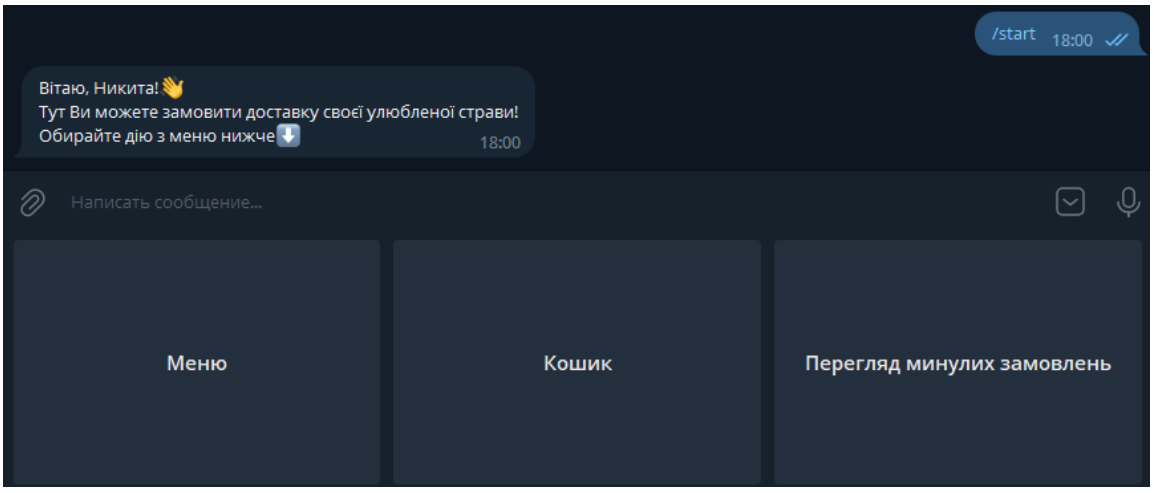

Рисунок 3.2 – Головне меню телеграм-бота

На рисунку 3.3 зображено категорії меню, які з’являються перед користувачем після обрання перегляду меню бота. Категорії представлені окремими кнопками, натискання на кожну з яких призведе до перегляду страв цієї категорії. Також присутня кнопка повернення до головного меню та кнопка перегляду всього меню, якщо користувач не бажає переглядати окрему категорію. Таким чином, це дозволяє зручно та зрозуміло орієнтуватися в меню, щоб обрати необхідну страву.

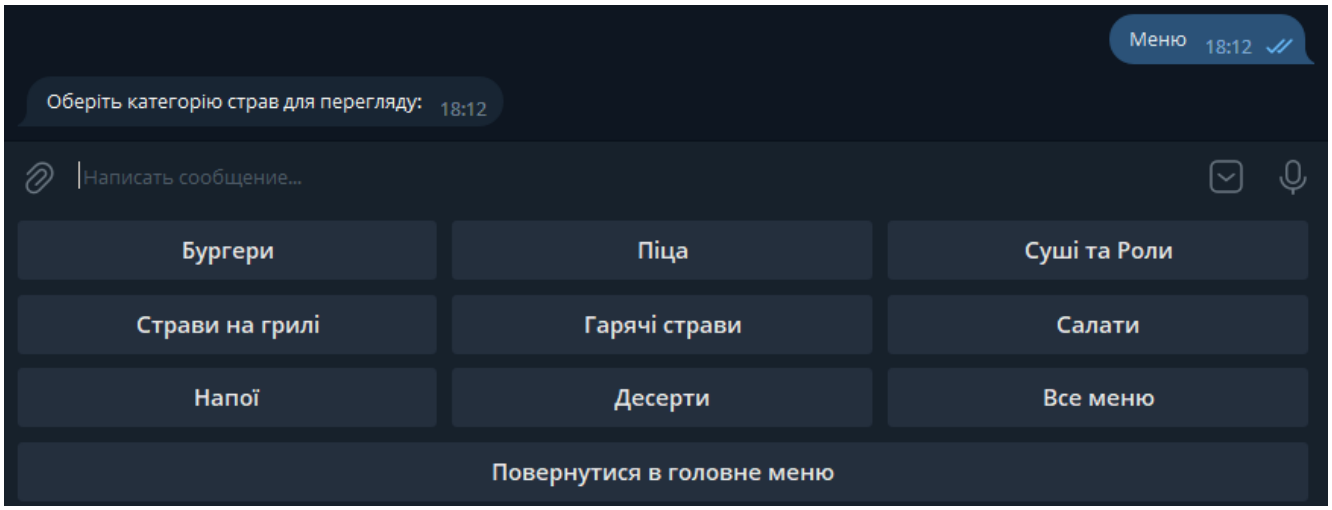

Рисунок 3.3 – Обрання категорії меню

На рисунку 3.4 зображено перелік страв, які відносять до обраної категорії, наприклад Бургери, а також демонстрація інформації про обрану страву. Натискаючи на назву страви, яку бажає переглянути користувач, з’являється повідомлення з описом обраної страви, її фотографією, а також кнопкою додавання цієї страви до кошику. Таким чином, обравши страву, можна додати її до кошику, або, якщо хочеться переглянути іншу страву, можна натиснути на її назву внизу та подивитися інформацію про неї. Також присутня кнопка для повернення в меню перегляду категорій страв. Таким чином, можна переглянути всі страви які належать до обраної категорії, переглянути детальну інформацію про кожну з них, при бажанні, додати їх до кошику та повернутися в меню, при необхідності.

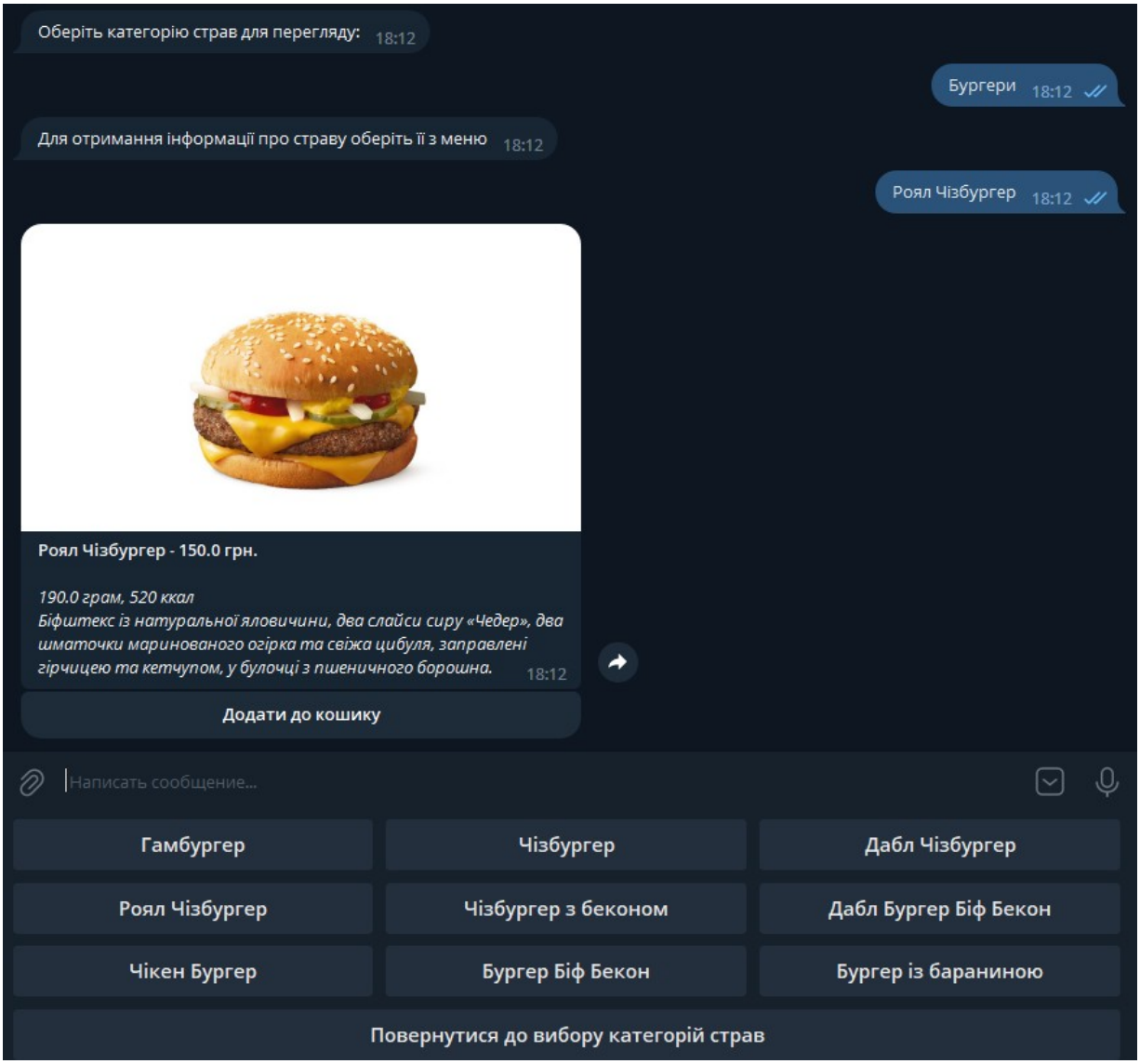

Рисунок 3.4 – Перегляд страв з меню

На рисунку 3.5 зображено додавання страви до кошику. Після натискання на кнопку «Додати до кошику» страва додається до кошику. Після цього користувач може натиснути на кнопку «Переглянути кошик». На екрані з’явиться меню кошику, таке саме, як при переході з головного меню до кошику. Користувачу демонструється перелік страв, які на поточний момент знаходяться в його кошику. Також, користувач має можливість, шляхом натискання на кнопки розташовані знизу, перейти до оформлення замовлення, внести зміни до кошику, повторно переглянути його вміст, а також повернутися в головне меню.

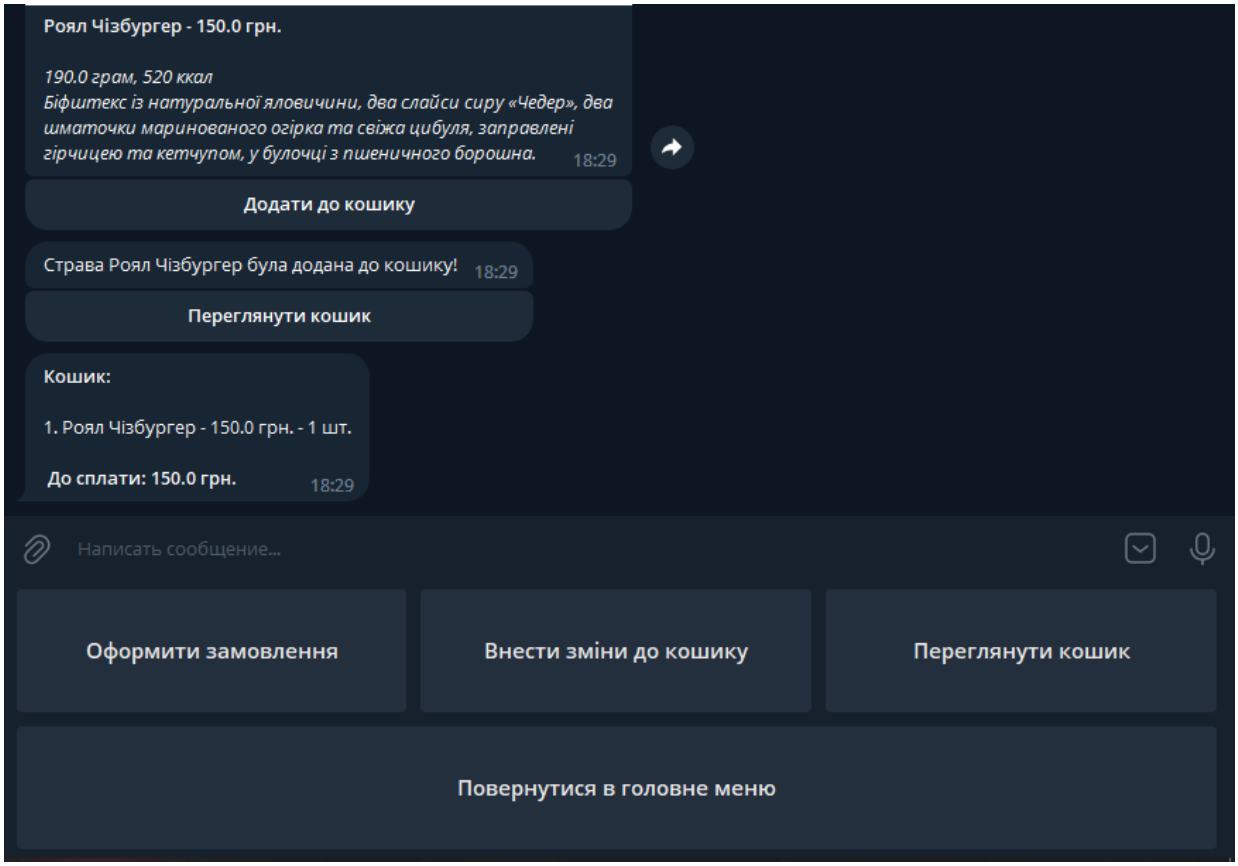

Рисунок 3.5 – Додавання страви до кошику та перегляд його вмісту

На рисунку 3.6 зображено меню внесення змін до кошику. Воно складається з кнопок зміни кількості страв, видалення страв з кошику, очищення кошику та повернення до меню кошика. За допомогою цього меню користувач може виконувати основні операції зі вмістом кошика, змінюючи його.

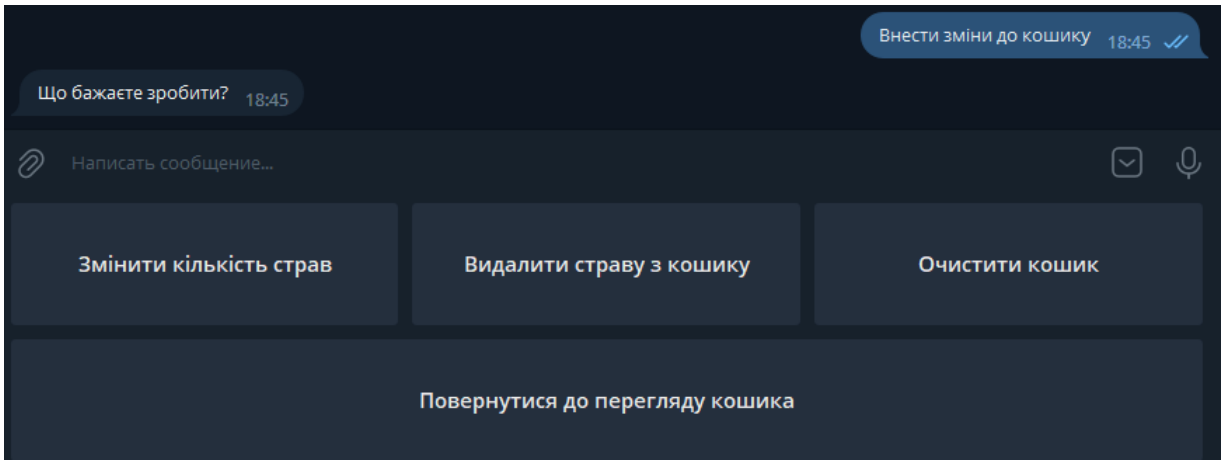

Рисунок 3.6 – Внесення змін до кошику

На рисунку 3.7 зображено видалення страви з кошику. Меню видалення страв складається з кнопок з назвою кожної страви з кошика. Для видалення страви необхідно натиснути на кнопку з її назвою внизу, після чого страва буде видалена, а користувача буде перенаправлено до меню кошика.

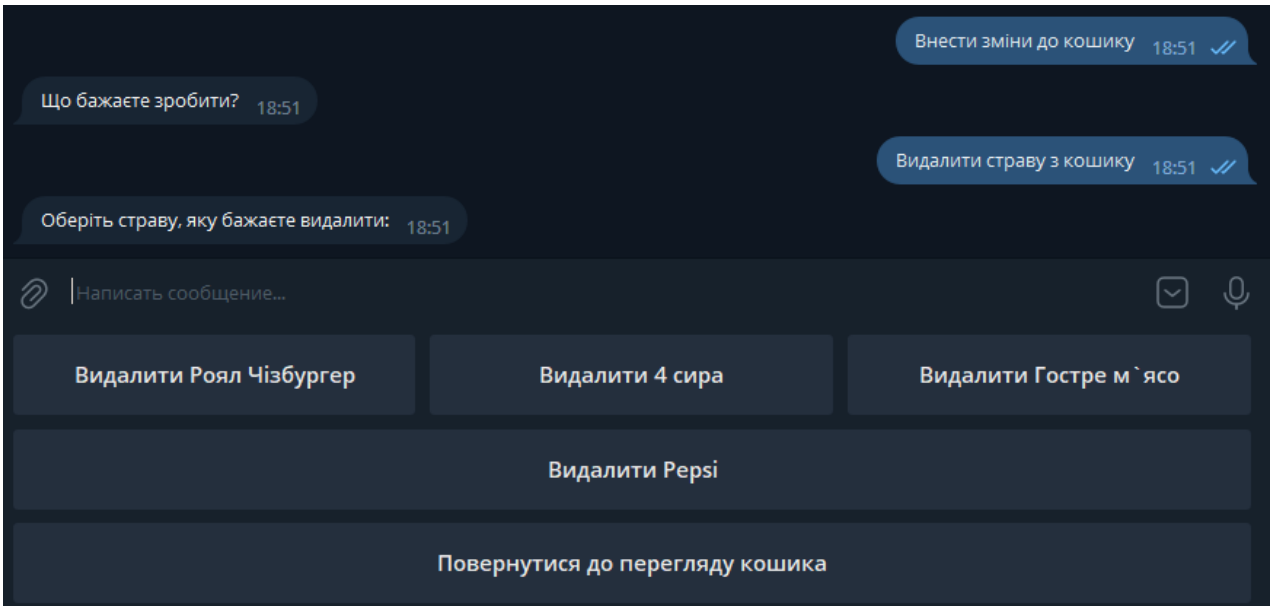

Рисунок 3.7 – Видалення страви з кошику

На рисунку 3.8 зображено зміна кількості страв в кошику. Так само, як і меню видалення страв, це меню складається з назв страв, які знаходяться в кошику та кількість яких можна змінити. Обравши страву, з’являється зображення цієї страви, опис, який включає кількість цієї страви в кошику, а також дві кнопки під цим повідомленням, за допомогою яких можна збільшити або зменшити кількість обраної страви. Після натискання однієї з кнопок, користувачу буде повідомлено про успішне виконання операції та надана можливість перейти до кошику, щоб переглянути його вміст після внесення змін. Таким чином, користувачу надається можливість зручно змінювати кількість страв в кошику та видаляти непотрібні страви.

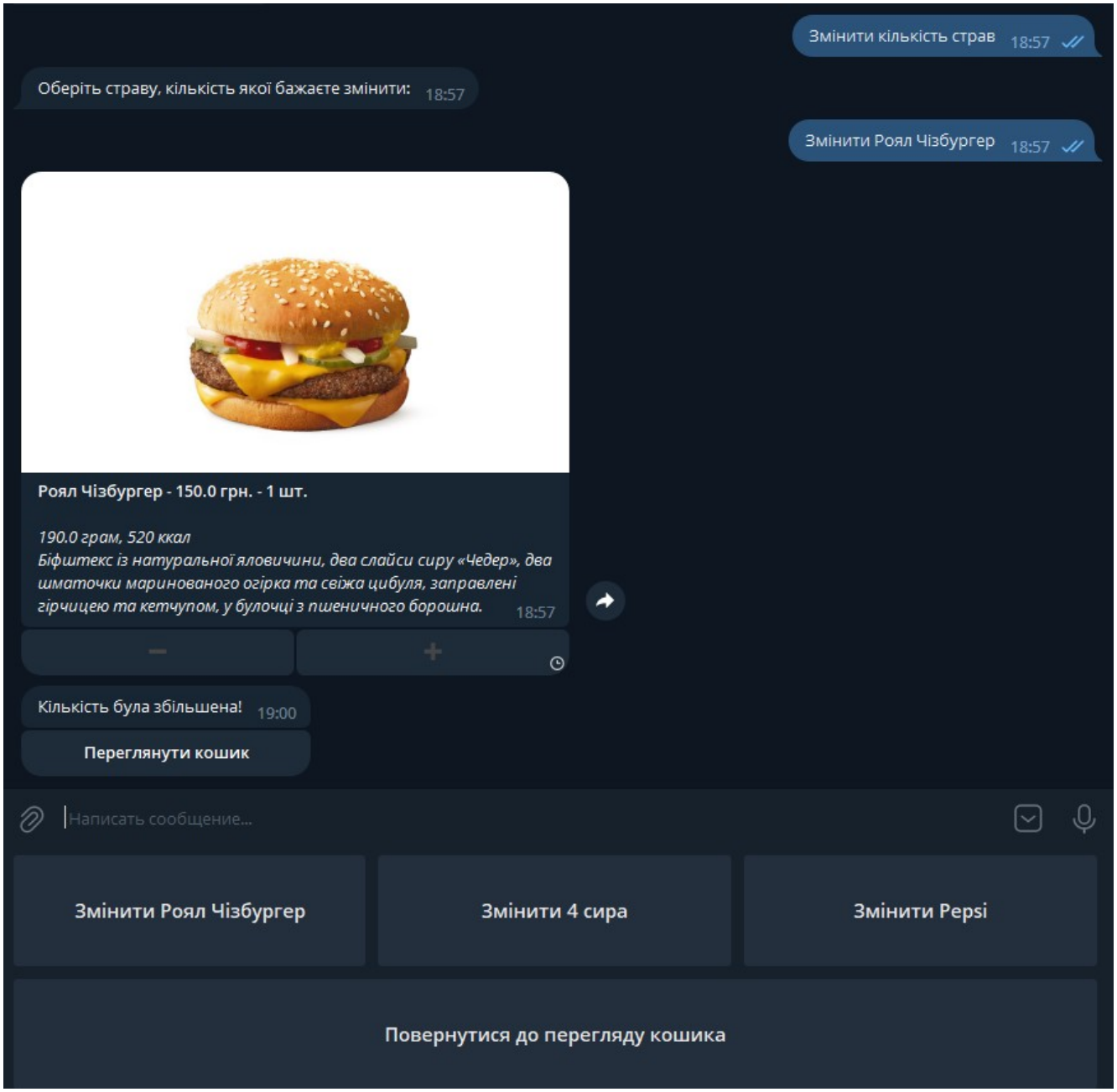

Рисунок 3.8 – Зміна кількості страв в кошику

На рисунку 3.9 зображено процес оформлення замовлення. Після того, як користувач визначився зі стравами, які бажає замовити, він може перейти до оформлення замовлення. Обравши в меню кошика оформлення замовлення, користувачу необхідно ввести свої контактні дані та адресу доставки, після чого, натиснувши на відповідну кнопку, обрати спосіб оплати замовлення, та, при необхідності, вказати додаткову інформацію до замовлення. Після цього буде сформовано повідомлення для підтвердження замовлення, в якому користувачу

потрібно перевірити коректність наданої інформації, після чого оформити замовлення, натиснувши на відповідну кнопку. Таким чином замовлення буде успішно оформлено та відправлено на обробку до менеджеру закладу.

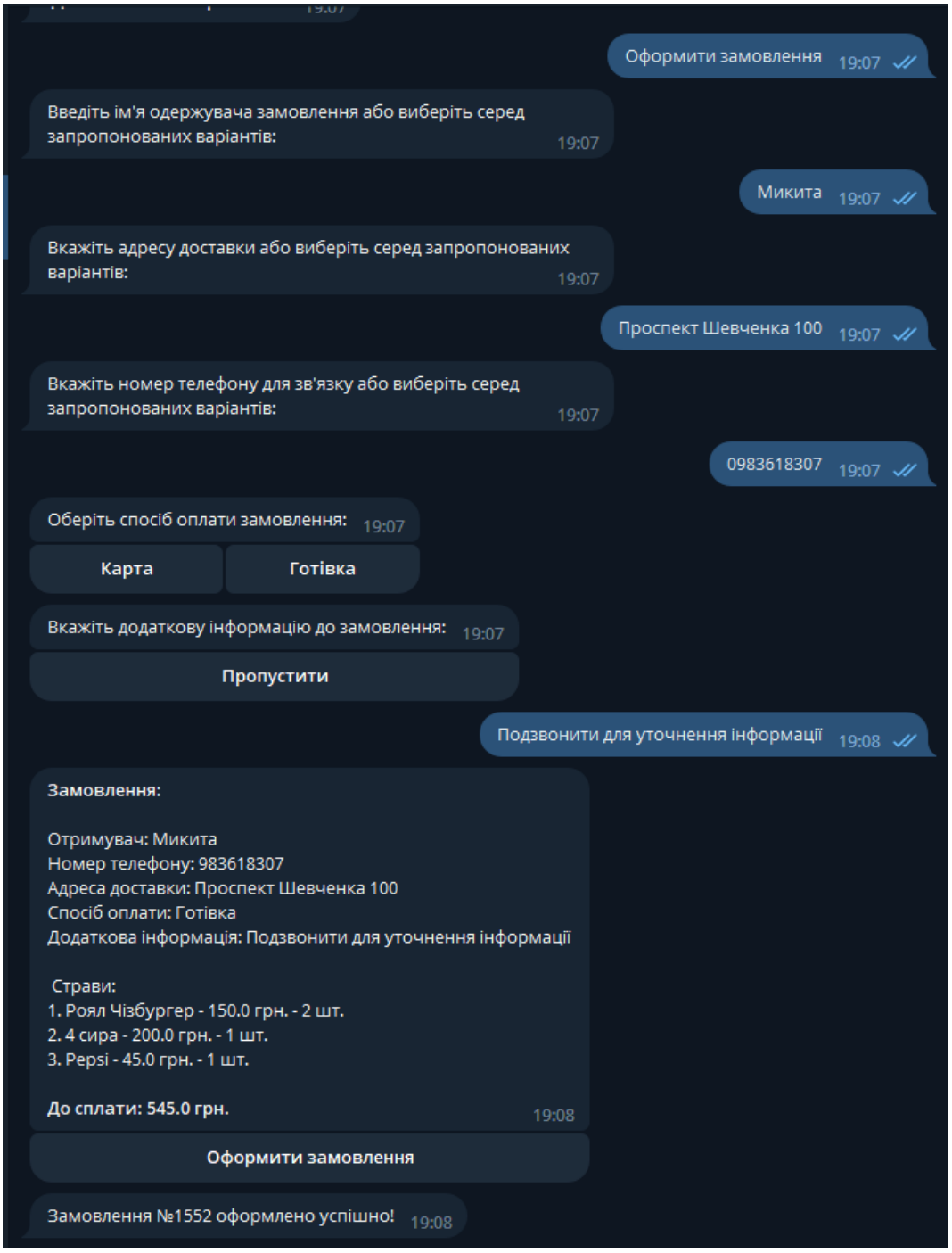

Рисунок 3.9 – Оформлення замовлення

На рисунку 3.10 зображено перегляд попередніх замовлень користувача. Для перегляду попередніх замовлень, користувачу необхідно зайти у відповідне меню. Воно складається з номерів замовлень, які робив цей користувач. Натиснувши на відповідний номер замовлення, користувач може переглянути всю інформацію про нього, включно з його станом готовності.

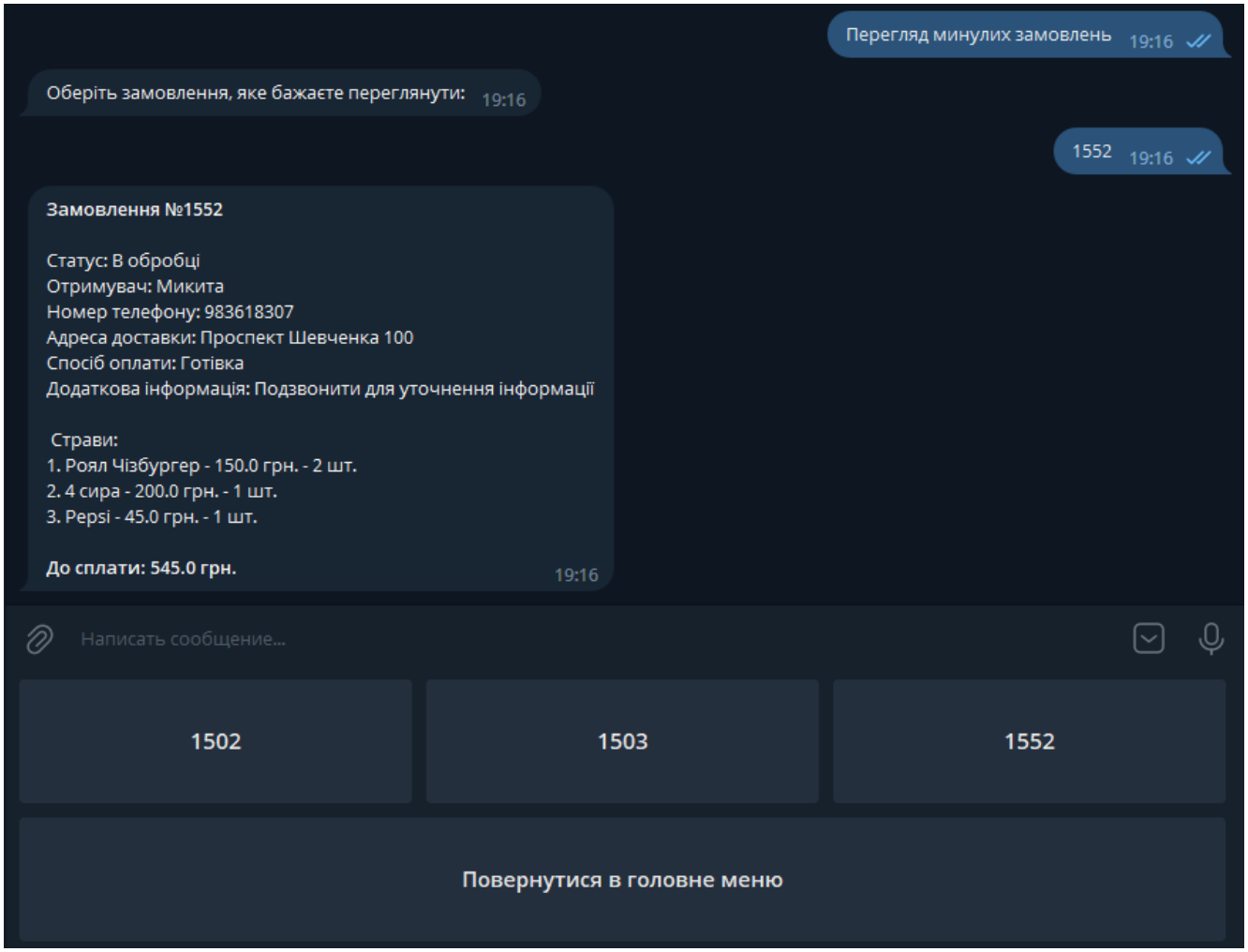

Рисунок 3.10 – Перегляд попередніх замовлень

Таким чином, було розглянуто основні меню телеграм-бота та можливі сценарії взаємодії користувача з ним. Виходячи з цього можна зробити висновок, що користувач може зручно та всього за декілька хвилин переглянути різні страви з меню, додати їх до кошику та оформити замовлення, а також, при необхідності, видаляти страви з кошику та змінювати їх кількість.

3.6 Вихідний код телеграм-бота

Вихідний код програмних класів розробленого телеграм-бота наведено в Додатку А даної курсової роботи.

3.7 Висновки до третього розділу

В третьому розділі курсової роботи було виконано програмну реалізацію телеграм-бота для замовлення їжі. Для успішного виконання програмної реалізації бота було використано розроблений у другому розділі цієї роботи проект телеграм-бота.

Була розглянута структура серверного програмного проекту. Як результат, було визначено основні компоненти та класи, з яких складається проект телеграм-бота для замовлення їжі.

Наступним кроком була створена діаграма класів телеграм-бота, за допомогою якої можна з легкістю визначити основні класи чат-бота та взаємодію між ним. Також, було розглянуто використання системи контролю версій, під час чого було визначено основні метрики керування програмним кодом проекту.

Крім того, було проведено функціональне тестування розробленого телеграм-бота. Був розроблений протокол тестування, який складався з множини тест-кейсів, за яким було проведено тестування розробленого чат-бота. В результаті чого, всі тест-кейси були успішно виконані. Це дозволяє переконатися в тому, що фактична поведінка телеграм-бота відповідає очікуваній поведінці, яка було визначена у функціональних вимогах.

Завершальним етапом, стала розробка інструкції користувача, яка складається з множини знімків екрану та пояснювального тексту до них. В поясненні розглядаються можливі дії користувача та функціонал телеграм-бота при конкретних сценаріях взаємодії користувача з чат-ботом. Також був наданий вихідний код програмних класів розробленого телеграм-бота.

ВИСНОВКИ

У даній курсовій роботі було розроблено телеграм-бот для замовлення доставки їжі.

Розроблений телеграм-бот надає можливість користувачам зручно та легко замовляти доставку своїх улюблених страв всього за декілька хвилин, не відволікаючись від важливих спілкувань та чатів. Користувач може переглядати різні страви в меню, додавати їх до кошику, оформлювати замовлення, вносити зміни до складу кошика, а також переглядати свої попередні замовлення. Таким чином, можна сказати, що основна мета, поставлена в цій курсовій роботі, успішно досягнута. Для досягнення мети було вирішено наступні задачі.

В першому розділі курсової роботи було детально розглянуто предметну область розробки телеграм-бота для замовлення їжі. Було проведено дослідження та аналіз аналогічних чат-ботів для доставки їжі, завдяки чому, були сформовані та детально описані основні вимоги до власного телеграм-бота. Окрім того, був проведений огляд інформаційних технологій для розробки телеграм-бота замовлення їжі. В якості яких виступають фреймворк Spring, а також Telegram API.

У другому розділі курсової роботи було проведено проектування телеграм-бота для замовлення їжі. Було визначено головну мету та задачі, які має виконувати телеграм-бот. Після чого було встановлено основні функціональні та нефункціональні вимоги до боту. Також було сформовано користувацькі історії зі сценаріями дій до кожної з них, а також розроблена діаграма варіантів використання телеграм-бота. Окрім того, були розроблені макети основних сценаріїв взаємодії користувача з телеграм-ботом.

В третьому розділі курсової роботи було виконано програмну реалізацію телеграм-бота. Була розглянута структура серверного програмного проекту та визначено основні компоненти та класи, з яких складається бот. Після чого була створена діаграма класів телеграм-бота та розглянуто використання системи контролю версій. Окрім того, було проведено функціональне тестування

розробленого телеграм-бота, яке пройшло успішно, а також розроблена інструкція користувача, створена з множини знімків екрану та пояснювального тексту до них. За результатами реалізації було надано вихідний код розробки телеграм-бота.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Spring. URL: https://spring.io (дата звернення 20.03.2023)
2. Фреймворк Spring та його особливості. URL: https://highload.today/uk/frejmvork-spring-ta-jogo-osoblivosti/ (дата звернення 20.03.2023)
3. Spring Framework пояснення за 5 хвилин. URL: https://techukraine.net/spring-framework-пояснюється-за-5-хвилин-або-менше/ (дата звернення 20.03.2023)
4. Telegram Bot API. URL: https://core.telegram.org/bots/api (дата звернення 10.04.2023)
5. Bots: An introduction for developers. URL: https://core.telegram.org/bots (дата звернення 10.04.2023)
6. Introducing Bot API 2.0. URL: https://core.telegram.org/bots/2-0-intro (дата звернення 10.04.2023)
7. Порівнюємо Spring Data JDBC та JPA. URL: https://dou.ua/forums/topic/36109/ (дата звернення 15.04.2023)
8. Будуємо телеграм чат-бот на Java: від ідеї до деплою. URL: https://dou.ua/forums/topic/38358/ (дата звернення 20.04.2023)
9. Чат-бот «М`ясторія». URL: https://myastoriya.com.ua/ua/blog/article/chat-bot-myastorii-kak-zakazat-nashu-produktsiyu-v-telegrame/ (дата звернення 20.04.2023)
10. Телеграм-бот для доставки їжі компанії Roll Club. URL: https://artjoker.ua/raboty/chat-bot/roll-club/ (дата звернення 20.04.2023)

ДОДАТОК А
КОД ПРОГРАМИ

Клас FoodDeliveryBot.java:

```
package CourseWork.FoodDeliveryBot;

import CourseWork.FoodDeliveryBot.config.BotConfig;
import CourseWork.FoodDeliveryBot.model.*;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;
import org.telegram.telegrambots.bots.TelegramLongPollingBot;
import org.telegram.telegrambots.meta.api.methods.ParseMode;
import org.telegram.telegrambots.meta.api.methods.send.SendMessage;
import org.telegram.telegrambots.meta.api.methods.send.SendPhoto;
import org.telegram.telegrambots.meta.api.objects.InputFile;
import org.telegram.telegrambots.meta.api.objects.Update;
import org.telegram.telegrambots.meta.api.objects.replykeyboard.InlineKeyboardMarkup;
import org.telegram.telegrambots.meta.api.objects.replykeyboard.ReplyKeyboardMarkup;
import org.telegram.telegrambots.meta.api.objects.replykeyboard.buttons.InlineKeyboardButton;
import org.telegram.telegrambots.meta.api.objects.replykeyboard.buttons.KeyboardRow;
import org.telegram.telegrambots.meta.exceptions.TelegramApiException;

import java.util.ArrayList;
import java.util.List;
import java.util.Objects;

@Component
public class FoodDeliveryBot extends TelegramLongPollingBot {

    public static final String START = "Старт";
    public static final String MAIN_MENU = "Головне меню";
    public static final String CATEGORY_OF_DISHES = "Категорії страв";
    public static final String CART = "Кошик";
    public static final String ALL_DISHES = "Все меню";
    public static final String MAKE_ORDER = "Оформити замовлення";
    public static final String CART_CHANGES = "Зміни у кошику";
    public static final String REMOVE_FROM_CART = "Видалення з кошику";
    public static final String CHANGE_NUMBER_OF_DISHES = "Змінити кількість страв";
    public static final String ADDED_TO_CART = "Додано в кошик";
    public static final String CART_REVIEW = "Перегляд кошику";
    public static final String ADD_TO_CART = "Додавання до кошику";
    public static final String DELIVERY_ADDRESS = "Вказати адресу доставки";
    public static final String PHONE_NUMBER = "Вказати номер телефону";
    public static final String PAYMENT_METHOD = "Вказати спосіб оплати";
    public static final String ADDITIONAL_INFO = "Вказати додаткову інформацію";
    public static final String CONFIRM_ORDER = "Підтвердити замовлення";
    public static final String PREVIOUS_ORDERS_REVIEW = "Перегляд минулих замовлень";
```

```
    @Autowired
    private final DishesRepository dishesRepository;
    @Autowired
    private final OrdersRepository ordersRepository;
    Cart cart;
    Order order;
    private final BotConfig botConfig;
    private String state;

     public FoodDeliveryBot(DishesRepository dishesRepository, OrdersRepository ordersRepository,
Cart cart, BotConfig botConfig) {
        this.dishesRepository = dishesRepository;
        this.ordersRepository = ordersRepository;
        this.cart = cart;
        this.order = new Order();
        this.botConfig = botConfig;
        this.state = START;
    }

    @Override
    public String getBotUsername() {
        return botConfig.getBotName();
    }

    @Override
    public String getBotToken() {
        return botConfig.getToken();
    }

    @Override
    public void onUpdateReceived(Update update) {
        if (update.hasMessage() && update.getMessage().hasText()) {
            messageHandler(update);
        } else if (update.hasCallbackQuery()) {
            callbackQueryHandler(update);
        }
    }

    private void messageHandler(Update update) {
        String textFromUser = update.getMessage().getText();
        Long chatId = update.getMessage().getChatId();
        String userFirstName = update.getMessage().getFrom().getFirstName();

        if (textFromUser.equals("/start") || textFromUser.equals("Старт")) {
            cart.setChatId(chatId);
            state = MAIN_MENU;
            String message = "Вітаю, " + userFirstName + "!\uD83D\uDC4B" + "\n" +
                    "Тут Ви можете замовити доставку своєї улюбленої страви!" + "\n" +
                    "Обирайте дію з меню нижче ⬇";
            sendMessage(chatId, message);
          } else if (textFromUser.equals("Меню") || textFromUser.equals("Повернутися до вибору
категорій страв")) {
```

```
            state = CATEGORY_OF_DISHES;
            sendMessage(chatId, "Оберіть категорію страв для перегляду: ");
        } else if (textFromUser.equals("Кошик") || textFromUser.equals("Переглянути кошик")) {
            state = CART;
            sendMessage(chatId, cart.getCartInfo());
        } else if (getDishCategories().contains(textFromUser)) {
            state = textFromUser;
            sendMessage(chatId, "Для отримання інформації про страву оберіть її з меню");
        } else if (getDishNames(getCategoryOfDish(textFromUser)).contains(textFromUser)) {
            sendDishInfo(chatId, textFromUser);
        } else if (textFromUser.equals("Повернутися в головне меню")) {
            state = MAIN_MENU;
            sendMessage(chatId, "Повернулися в головне меню. \nЩо бажаєте зробити?");
        } else if (textFromUser.equals("Все меню")) {
            state = ALL_DISHES;
            sendMessage(chatId, "Для отримання інформації про страву оберіть її з меню");
        } else if (textFromUser.equals("Оформити замовлення")) {
            if (cart.isEmpty()) {
                sendMessage(chatId, "Кошик порожній, тому оформити замовлення неможливо!");
            } else {
                state = MAKE_ORDER;
                        sendMessage(chatId, "Введіть ім'я одержувача замовлення або виберіть серед
запропонованих варіантів:");
            }
        } else if (state.equals(MAKE_ORDER)) {
            order.setChatId(chatId);
            order.setUserName(textFromUser);
            order.setOrderList(cart.getOrderList());
            order.setOrderPrice(cart.getTotalPrice());
            state = DELIVERY_ADDRESS;
                sendMessage(chatId, "Вкажіть адресу доставки або виберіть серед запропонованих
варіантів:");
        } else if (state.equals(DELIVERY_ADDRESS)) {
            order.setDeliveryAddress(textFromUser);
            state = PHONE_NUMBER;
                    sendMessage(chatId, "Вкажіть номер телефону для зв'язку або виберіть серед
запропонованих варіантів:");
        } else if (state.equals(PHONE_NUMBER)) {
            order.setPhoneNumber(Long.parseLong(textFromUser));
            state = PAYMENT_METHOD;
            sendMessage(chatId, "Оберіть спосіб оплати замовлення:");
        } else if (state.equals(ADDITIONAL_INFO)) {
            order.setAdditionalInfo(textFromUser);
            state = CONFIRM_ORDER;
            sendMessage(chatId, getOrderInfo(order));
        } else if (textFromUser.equals("Внести зміни до кошику")) {
            if (cart.isEmpty()) {
                sendMessage(chatId, "Кошик порожній, тому внесення змін неможливе!");
            } else {
                state = CART_CHANGES;
                sendMessage(chatId, "Що бажаєте зробити?");
            }
```

```
        } else if (textFromUser.equals("Видалити страву з кошику")) {
            state = REMOVE_FROM_CART;
            sendMessage(chatId, "Оберіть страву, яку бажаєте видалити:");
        } else if (cart.getDishNamesToChange("Видалення з кошику").contains(textFromUser)) {
            state = CART;
            textFromUser = textFromUser.replaceFirst("Видалити ", "");
            cart.removeAllDishWithName(textFromUser);
            sendMessage(chatId, "Страву " + textFromUser + " видалено з кошику.");
        } else if (textFromUser.equals("Змінити кількість страв")) {
            state = CHANGE_NUMBER_OF_DISHES;
            sendMessage(chatId, "Оберіть страву, кількість якої бажаєте змінити:");
        } else if (cart.getDishNamesToChange("Змінити кількість страв").contains(textFromUser)) {
            state = CART_CHANGES;
            textFromUser = textFromUser.replaceFirst("Змінити ", "");
                sendDishPhoto(chatId, cart.getDishPhoto(textFromUser), cart.getDishInfo(textFromUser),
textFromUser);
        } else if (textFromUser.equals("Очистити кошик")) {
            state = MAIN_MENU;
            cart.removeAllFromCart();
            sendMessage(chatId, "Всі страви з кошику було видалено.");
        } else if (textFromUser.equals("Повернутися до перегляду кошика")) {
            state = CART;
            sendMessage(chatId, "Повернулися до перегляду кошика.");
        } else if (textFromUser.equals("Перегляд минулих замовлень")) {
            state = PREVIOUS_ORDERS_REVIEW;
            sendMessage(chatId, "Оберіть замовлення, яке бажаєте переглянути:");
                                    } else if (state.equals(PREVIOUS_ORDERS_REVIEW) &&
getElementFromOrdersByChatId(chatId).contains(textFromUser)) {
                                                                                sendMessage(chatId,
getOrderInfo(Objects.requireNonNull(getOrderById(Long.parseLong(textFromUser)))));
        } else {
            state = MAIN_MENU;
            sendMessage(chatId, "Введена команда не підтримується.");
        }
    }

    private void callbackQueryHandler(Update update) {
        String callbackData = update.getCallbackQuery().getData();
        long chatId = update.getCallbackQuery().getMessage().getChatId();

                    if (getDishNames(getCategoryOfDish(callbackData)).contains(callbackData) &&
state.equals(ADD_TO_CART)) {
            cart.addDishToCart(callbackData);
            state = ADDED_TO_CART;
            sendMessage(chatId, "Страва " + callbackData + " була додана до кошику!");
                } else if (callbackData.equals("Кошик") && (state.equals(ADDED_TO_CART) ||
state.equals(CART_REVIEW))) {
            state = CART;
            sendMessage(chatId, cart.getCartInfo());
                    } else if (cart.getDishNamesToChange("Decrease").contains(callbackData) &&
state.equals(CART_CHANGES)) {
            callbackData = callbackData.replaceFirst("Decrease ", "");
```

```
            cart.removeDish(callbackData);
            state = CART_REVIEW;
            sendMessage(chatId, "Кількість була зменшена!");
                    } else if (cart.getDishNamesToChange("Increase").contains(callbackData) &&
state.equals(CART_CHANGES)) {
            callbackData = callbackData.replaceFirst("Increase ", "");
            cart.addDishToCart(callbackData);
            state = ADDED_TO_CART;
            sendMessage(chatId, "Кількість була збільшена!");
                } else if ((callbackData.equals("Карта") || callbackData.equals("Готівка")) &&
state.equals(PAYMENT_METHOD)) {
            order.setPaymentMethod(callbackData);
            state = ADDITIONAL_INFO;
            sendMessage(chatId, "Вкажіть додаткову інформацію до замовлення:");
        } else if (callbackData.equals("Пропустити") && state.equals(ADDITIONAL_INFO)) {
            state = CONFIRM_ORDER;
            sendMessage(chatId, getOrderInfo(order));
        } else if (callbackData.equals("Оформити замовлення") && state.equals(CONFIRM_ORDER))
{
            order.setStatus("В обробці");
            ordersRepository.save(order);
            String message = "Замовлення №" + order.getId() + " оформлено успішно!";
            cart.removeAllFromCart();
            order = new Order();
            state = MAIN_MENU;
            sendMessage(chatId, message);
        }
    }

    private void sendMessage(Long chatId, String textToSend) {
        SendMessage message = new SendMessage();
        message.setChatId(String.valueOf(chatId));
        message.setText(textToSend);
        message.setParseMode(ParseMode.HTML);

        keyboardBuilder(message);

        try {
            execute(message);
        } catch (TelegramApiException e) {
            System.out.println("Error while sending message!");
        }
    }

        private void sendDishPhoto(Long chatId, String imageToSend, String textToSend, String
dishName) {
        SendPhoto photo = new SendPhoto();
        photo.setChatId(chatId);
        photo.setPhoto(new InputFile(imageToSend));
        photo.setParseMode(ParseMode.HTML);
        photo.setCaption(textToSend);
```

```
        if (state.equals(ADD_TO_CART)) {
            photo.setReplyMarkup(inlineKeyboardBuilder("Додати до кошику", dishName));
        } else if (state.equals(CART_CHANGES)) {
                    photo.setReplyMarkup(inlineKeyboardBuilder("➖", "Decrease " + dishName, "➕",
"Increase " + dishName));
        }

        try {
            execute(photo);
        } catch (TelegramApiException e) {
            System.out.println("Error while sending photo!");
        }
    }

    private InlineKeyboardMarkup inlineKeyboardBuilder(String ButtonText, String callbackData) {
        InlineKeyboardMarkup inlineMarkup = new InlineKeyboardMarkup();
        List<List<InlineKeyboardButton>> inlineRows = new ArrayList<>();
        List<InlineKeyboardButton> inlineRow = new ArrayList<>();

        InlineKeyboardButton inlineButton = new InlineKeyboardButton();

        inlineButton.setText(ButtonText);
        inlineButton.setCallbackData(callbackData);
        inlineRow.add(inlineButton);

        inlineRows.add(inlineRow);
        inlineMarkup.setKeyboard(inlineRows);

        return inlineMarkup;
    }

            private InlineKeyboardMarkup inlineKeyboardBuilder(String firstButtonText, String
firstButtonCallbackData, String secondButtonText, String secondButtonCallbackData) {
        InlineKeyboardMarkup inlineMarkup = new InlineKeyboardMarkup();
        List<List<InlineKeyboardButton>> inlineRows = new ArrayList<>();
        List<InlineKeyboardButton> inlineRow = new ArrayList<>();

        InlineKeyboardButton firstButton = new InlineKeyboardButton();
        firstButton.setText(firstButtonText);
        firstButton.setCallbackData(firstButtonCallbackData);
        inlineRow.add(firstButton);

        InlineKeyboardButton secondButton = new InlineKeyboardButton();
        secondButton.setText(secondButtonText);
        secondButton.setCallbackData(secondButtonCallbackData);
        inlineRow.add(secondButton);

        inlineRows.add(inlineRow);
        inlineMarkup.setKeyboard(inlineRows);

        return inlineMarkup;
    }
```

```
    private void keyboardBuilder(SendMessage message) {
        List<String> buttons = new ArrayList<>();

        if (state.equals(START)) {
            buttons.add("Старт");
            message.setReplyMarkup(ReplyKeyboardBuilder(buttons));
        } else if (state.equals(MAIN_MENU)) {
            buttons.add("Меню");
            buttons.add("Кошик");
            buttons.add("Перегляд минулих замовлень");
            message.setReplyMarkup(ReplyKeyboardBuilder(buttons));
        } else if (state.equals(CATEGORY_OF_DISHES)) {
            buttons.add("Бургери");
            buttons.add("Піца");
            buttons.add("Суші та Роли");
            buttons.add("Страви на грилі");
            buttons.add("Гарячі страви");
            buttons.add("Салати");
            buttons.add("Напої");
            buttons.add("Десерти");
            buttons.add("Все меню");
            buttons.add("Повернутися в головне меню");
            message.setReplyMarkup(ReplyKeyboardBuilder(buttons));
        } else if (getDishCategories().contains(state) || state.equals(ALL_DISHES)) {
            buttons = getDishNames(state);
            buttons.add("Повернутися до вибору категорій страв");
            message.setReplyMarkup(ReplyKeyboardBuilder(buttons));
        } else if (state.equals(CART)) {
            buttons.add("Оформити замовлення");
            buttons.add("Внести зміни до кошику");
            buttons.add("Переглянути кошик");
            buttons.add("Повернутися в головне меню");
            message.setReplyMarkup(ReplyKeyboardBuilder(buttons));
        } else if (state.equals(CART_CHANGES)) {
            buttons.add("Змінити кількість страв");
            buttons.add("Видалити страву з кошику");
            buttons.add("Очистити кошик");
            buttons.add("Повернутися до перегляду кошика");
            message.setReplyMarkup(ReplyKeyboardBuilder(buttons));
                                    } else if (state.equals(REMOVE_FROM_CART) ||
state.equals(CHANGE_NUMBER_OF_DISHES)) {
            buttons = cart.getDishNamesToChange(state);
            buttons.add("Повернутися до перегляду кошика");
            message.setReplyMarkup(ReplyKeyboardBuilder(buttons));
              } else if (state.equals(PREVIOUS_ORDERS_REVIEW) || state.equals(MAKE_ORDER) ||
state.equals(PHONE_NUMBER) || state.equals(DELIVERY_ADDRESS)) {
            buttons = getElementFromOrdersByChatId(Long.parseLong(message.getChatId()));
            buttons.add("Повернутися в головне меню");
            message.setReplyMarkup(ReplyKeyboardBuilder(buttons));
        } else if (state.equals(ADDED_TO_CART) || state.equals(CART_REVIEW)) {
            message.setReplyMarkup(inlineKeyboardBuilder("Переглянути кошик", "Кошик"));
```

```
        } else if (state.equals(PAYMENT_METHOD)) {
            message.setReplyMarkup(inlineKeyboardBuilder("Карта", "Карта", "Готівка", "Готівка"));
        } else if (state.equals(ADDITIONAL_INFO)) {
            message.setReplyMarkup(inlineKeyboardBuilder("Пропустити", "Пропустити"));
        } else if (state.equals(CONFIRM_ORDER)) {
                message.setReplyMarkup(inlineKeyboardBuilder("Оформити замовлення", "Оформити
замовлення"));
        }
    }

    private List<String> getDishNames(String dishCategory) {
        var dishes = dishesRepository.findAll();

        List<String> dishNames = new ArrayList<>();

        for (Dish dish : dishes) {
            if (dishCategory.equals(dish.getCategory()) || dishCategory.equals(ALL_DISHES))
                dishNames.add((dish.getName()).trim());
        }
        return dishNames;
    }

    private List<String> getDishCategories() {
        var dishes = dishesRepository.findAll();

        List<String> dishCategories = new ArrayList<>();

        for (Dish dish : dishes) {
            if (!dishCategories.contains(dish.getCategory()))
                dishCategories.add(dish.getCategory());
        }
        return dishCategories;
    }

    private String getCategoryOfDish(String dishName) {
        Dish dish = getDishByName(dishName);

        if (dish != null)
            return dish.getCategory();

        return "none";
    }

    private void sendDishInfo(Long chatId, String dishName) {
        Dish dish = getDishByName(dishName);
        String message = "Помилка";

        if (dish == null) {
            sendMessage(chatId, message);
            return;
        }
```

```
        if (dish.getCategory().equals("Напої")) {
                message = "<b>" + dish.getName() + " - " + dish.getPrice() + " грн.</b>\n\n<i>" +
dish.getWeight() + " л, " +
                dish.getAdditional() + "\n" + dish.getDescription() + "</i>";
        } else {
                message = "<b>" + dish.getName() + " - " + dish.getPrice() + " грн.</b>\n\n<i>" +
dish.getWeight() + " грам, " +
                dish.getAdditional() + "\n" + dish.getDescription() + "</i>";
        }

        if (!state.equals(CART_CHANGES))
            state = ADD_TO_CART;

        sendDishPhoto(chatId, dish.getImage(), message, dishName);
    }

    private Dish getDishByName(String dishName) {
        var dishes = dishesRepository.findAll();

        for (Dish dish : dishes) {
            if (dishName.equals((dish.getName().trim())))
                return dish;
        }

        return null;
    }

    private List<Order> getOrdersByChatId(long chatId) {
        var orders = ordersRepository.findAll();

        List<Order> userOrders = new ArrayList<>();

        for (Order order : orders) {
            if (order.getChatId() == chatId)
                userOrders.add(order);
        }

        return userOrders;
    }

    private Order getOrderById(long orderId) {
        var orders = ordersRepository.findAll();

        for (Order order : orders) {
            if (order.getId() == orderId)
                return order;
        }

        return null;
    }

    private List<String> getElementFromOrdersByChatId(long chatId) {
```

```
        List<Order> orders = getOrdersByChatId(chatId);

        List<String> result = new ArrayList<>();

        switch (state) {
            case PREVIOUS_ORDERS_REVIEW:
                for (Order order : orders) {
                    result.add("" + order.getId());
                }
                break;
            case MAKE_ORDER:
                for (Order order : orders) {
                    if (!result.contains(order.getUserName()))
                        result.add(order.getUserName());
                }
                break;
            case PHONE_NUMBER:
                for (Order order : orders) {
                    if (!result.contains("" + order.getPhoneNumber()))
                        result.add("" + order.getPhoneNumber());
                }
                break;
            case DELIVERY_ADDRESS:
                for (Order order : orders) {
                    if (!result.contains(order.getDeliveryAddress()))
                        result.add(order.getDeliveryAddress());
                }
                break;
        }

        return result;
    }

    private String getOrderInfo(Order order) {
        String orderInfo;

        if (order.getId() != 0) {
            orderInfo = "<b>Замовлення №" + order.getId() + "</b>\n\nСтатус: " + order.getStatus();
        } else {
            orderInfo = "<b>Замовлення:</b>\n";
        }

        orderInfo = orderInfo + "\nОтримувач: " + order.getUserName() +
                        "\nНомер телефону: " + order.getPhoneNumber() + "\nАдреса доставки: " +
order.getDeliveryAddress() +
                "\nСпосіб оплати: " + order.getPaymentMethod();

        if (order.getAdditionalInfo() != null) {
            orderInfo = orderInfo + "\nДодаткова інформація: " + order.getAdditionalInfo();
        }
             orderInfo = orderInfo + "\n\n Страви: \n" + order.getOrderList() + "\n<b>До сплати: " +
order.getOrderPrice() + " грн. </b>";
```

```
        return orderInfo;
    }

    private ReplyKeyboardMarkup ReplyKeyboardBuilder(List<String> list) {
        ReplyKeyboardMarkup keyboardMarkup = new ReplyKeyboardMarkup();
        List<KeyboardRow> keyboardRows = new ArrayList<>();
        KeyboardRow row = new KeyboardRow();

        for (int i = 0; i < list.size(); i++) {
            row.add(list.get(i));
            if ((1 + i) % 3 == 0 || i == list.size() - 1 || (i == list.size() - 2 && !state.equals(MAIN_MENU)))
{
                keyboardRows.add(row);
                row = new KeyboardRow();
            }
        }

        keyboardMarkup.setKeyboard(keyboardRows);
        return keyboardMarkup;
    }

}
```

Клас Cart.java:

```
package CourseWork.FoodDeliveryBot.model;

import lombok.Getter;
import lombok.Setter;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

import java.util.ArrayList;
import java.util.List;

@Getter
@Setter
@Component
public class Cart {

    @Autowired
    private DishesRepository dishesRepository;

    private long chatId;
    private List<Dish> dishes;

    public Cart(DishesRepository dishesRepository) {
        this.dishesRepository = dishesRepository;
        this.dishes = new ArrayList<>();
```

```
    }

    public void addDishToCart(String dishName) {
        if (findDishInMenu(dishName) != null)
            dishes.add(findDishInMenu(dishName));
    }

    public Dish findDishInMenu(String dishName) {
        var dishes = dishesRepository.findAll();

        for (Dish dish : dishes) {
            if (dishName.equals(dish.getName()))
                return dish;
        }
        return null;
    }

    public float getTotalPrice() {
        float totalPrice = 0;

        if (!dishes.isEmpty()) {
            for (Dish dish : dishes) {
                totalPrice += dish.getPrice();
            }
        }

        return totalPrice;
    }

    public String getCartInfo() {
        StringBuilder message = new StringBuilder("Кошик порожній");

        if (!dishes.isEmpty()) {
            message = new StringBuilder("<b>Кошик:</b>\n\n" + getOrderList() + "\n<b> До сплати: " +
getTotalPrice() + " грн.</b>");
        }

        return message.toString();
    }

    public int getDishQuantity(Dish dish) {
        int dishQuantity = 0;

        for (Dish value : dishes) {
            if (value.getId() == dish.getId())
                dishQuantity++;
        }

        return dishQuantity;
    }

    public void removeAllDishWithName(String dishName) {
```

```
    if (findDishInMenu(dishName) != null && !dishes.isEmpty()) {
        dishes.removeIf(dish -> dish.getName().equals(dishName));
    }
}

public List<String> getDishNamesToChange(String change) {
    List<String> dishNames = new ArrayList<>();

    switch (change) {
        case "Видалення з кошику" -> change = "Видалити ";
        case "Змінити кількість страв" -> change = "Змінити ";
        case "Decrease" -> change = "Decrease ";
        case "Increase" -> change = "Increase ";
    }

    for (Dish dish : getDishesWithoutRepeat()) {
        dishNames.add(change + dish.getName());
    }
    return dishNames;
}

public List<Dish> getDishesWithoutRepeat() {
    List<Dish> uniqueDishes = new ArrayList<>();
    for (Dish dish : dishes) {
        if (!uniqueDishes.contains(dish)) {
            uniqueDishes.add(dish);
        }
    }
    return uniqueDishes;
}

public Dish getDishFromCart(String dishName) {
    List<Dish> dishes = getDishesWithoutRepeat();

    for (Dish dish : dishes) {
        if (dish.getName().equals(dishName))
            return dish;
    }

    return null;
}

public String getDishInfo(String dishName) {
    Dish dish = getDishFromCart(dishName);
    String message = "Помилка";

    if (dish == null) {
        return message;
    }

    if (dish.getCategory().equals("Напої")) {
```

```
                message = "<b>" + dish.getName() + " - " + dish.getPrice() + " грн. - " +
getDishQuantity(dish) + " шт.</b>\n\n<i>" +
                dish.getWeight() + " л, " + dish.getAdditional() + "\n" + dish.getDescription() + "</i>";
        } else {
                message = "<b>" + dish.getName() + " - " + dish.getPrice() + " грн. - " +
getDishQuantity(dish) + " шт.</b>\n\n<i>" +
                dish.getWeight() + " грам, " + dish.getAdditional() + "\n" + dish.getDescription() +
"</i>";
        }

        return message;
    }

    public String getDishPhoto(String dishName) {
        Dish dish = getDishFromCart(dishName);

        if (dish == null)
            return null;

        return dish.getImage();
    }

    public void removeDish(String dishName) {
        if (findDishInMenu(dishName) != null && !dishes.isEmpty()) {
            for (int i = dishes.size() - 1; i >= 0; i--) {
                if (dishes.get(i).getName().equals(dishName)) {
                    dishes.remove(i);
                    return;
                }
            }
        }
    }

    public void removeAllFromCart() {
        dishes.clear();
    }

    public boolean isEmpty() {
        return dishes.isEmpty();
    }

    public String getOrderList() {
        StringBuilder message = new StringBuilder();

        if (!dishes.isEmpty()) {

            List<Dish> dishesWithoutRepeat = new ArrayList<>(getDishesWithoutRepeat());

            for (Dish dish : dishesWithoutRepeat) {
                                message.append(dishesWithoutRepeat.indexOf(dish) + 1).append(".
").append(dish.getName()).append(" - ").append(dish.getPrice()).append(" грн. -
").append(getDishQuantity(dish)).append(" шт.\n");
```

```
            }
        }

        return message.toString();
    }
}
```

Клас Dish.java:

```
package CourseWork.FoodDeliveryBot.model;

import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import lombok.Getter;
import lombok.Setter;

@Getter
@Setter
@Entity(name = "dish")
public class Dish {

    @Id
    private int id;

    private String category;

    private String name;

    private String description;

    private String additional;

    private float weight;

    private float price;

    private String image;


    @Override
    public boolean equals(Object o) {
        if (this == o)
            return true;
        if (o == null || getClass() != o.getClass())
            return false;

        Dish dish = (Dish) o;

        return id == dish.getId() && name.equals(dish.getName());
    }
```

```
}
```

Клас Order.java:

```
package CourseWork.FoodDeliveryBot.model;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import lombok.Getter;
import lombok.Setter;

@Getter
@Setter
@Entity(name = "orders")
public class Order {

    @Id
    @GeneratedValue(strategy = GenerationType.SEQUENCE)
    private long id;

    private long chatId;

    private String userName;

    private String status;

    private String orderList;

    private float orderPrice;

    private long phoneNumber;

    private String deliveryAddress;

    private String paymentMethod;

    private String additionalInfo;

}
```

Інтерфейс DishesRepository.java:

```
package CourseWork.FoodDeliveryBot.model;

import org.springframework.data.repository.CrudRepository;
```

```
public interface DishesRepository extends CrudRepository<Dish, Integer> {
}
```

Інтерфейс OrdersRepository.java:

```
package CourseWork.FoodDeliveryBot.model;

import org.springframework.data.repository.CrudRepository;

public interface OrdersRepository extends CrudRepository<Order, Integer> {

}
```

Клас BotConfig.java:

```
package CourseWork.FoodDeliveryBot.config;

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

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

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

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

}
```

Клас BotInitializer.java:

```
package CourseWork.FoodDeliveryBot.config;

import CourseWork.FoodDeliveryBot.FoodDeliveryBot;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.event.ContextRefreshedEvent;
import org.springframework.context.event.EventListener;
import org.springframework.stereotype.Component;
import org.telegram.telegrambots.meta.TelegramBotsApi;
import org.telegram.telegrambots.meta.exceptions.TelegramApiException;
```

```
import org.telegram.telegrambots.updatesreceivers.DefaultBotSession;

@Component
public class BotInitializer {
   private final FoodDeliveryBot telegramBot;

   @Autowired
   public BotInitializer(FoodDeliveryBot telegramBot) {
      this.telegramBot = telegramBot;
   }

   @EventListener({ContextRefreshedEvent.class})
   public void init()throws TelegramApiException {
      TelegramBotsApi telegramBotsApi = new TelegramBotsApi(DefaultBotSession.class);
      try{
         telegramBotsApi.registerBot(telegramBot);
      } catch (TelegramApiException e){
         System.out.println("Error while registering Bot!");
      }
   }
}
```

Клас FoodDeliveryBotApplication.java:

```
package CourseWork.FoodDeliveryBot;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootApplication
public class FoodDeliveryBotApplication {

        public static void main(String[] args){
                SpringApplication.run(FoodDeliveryBotApplication.class, args);
        }

}
```