ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інженерії програмного забезпечення

Пояснювальна записка

до бакалаврської роботи
на ступінь вищої освіти бакалавр

на тему: «**РОЗРОБКА ТЕЛЕГРАМ-БОТУ З МЕТОЮ ПОКРАЩЕННЯ ПРОЦЕСУ РЕАЛІЗАЦІЇ ПРОДУКТІВ ПОБУТОВОЇ ХІМІЇ МОВОЮ JAVASCRIPT**»

Виконав: студент 4 курсу, групи ПД-44
спеціальності
121 Інженерія програмного забезпечення
(шифр і назва спеціальності/спеціалізації)

Ярош А.О.
(прізвище та ініціали)

Керівник Жебка В.В.
(прізвище та ініціали)

Рецензент _____
(прізвище та ініціали)

Київ –2022

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра Інженерії програмного забезпечення
Ступінь вищої освіти - «Бакалавр»
Спеціальність підготовки – 121 «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____Негоденко О.В.
"____" _____2022 року

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТА

ЯРОША АРТЕМА ОЛЕГОВИЧА
(прізвище, ім'я, по батькові)

1. Тема роботи: «Розробка телеграм-боту з метою покращення процесу реалізації продуктів побутової хімії мовою JavaScript»

Керівник роботи: Жебка В.В., д.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом вищого навчального закладу від «18» лютого 2022 року № 22.

2. Строк подання студентом роботи «3» червня 2022 року

3. Вхідні дані до роботи:
Наукова література з питань, пов'язаних з програмним забезпеченням щодо роботи та особливостей розробки Telegram-ботів і подальшому їх використанні в сфері бізнесу;

4. Зміст розрахунково-пояснювальної записки(перелік питань, які потрібно розробити).

4.1. Дослідження розвитку, можливостей та актуальності ведення бізнесу в Telegram.

4.2. Розробка структури телеграм-боту з метою покращення процесу реалізації продуктів побутової хімії мовою JavaScript.

4.3. Програмна реалізація телеграм-бота та тестування.

4.4. Висновки.

5. Перелік демонстраційного матеріалу (назва основних слайдів)
   5.1 Титульний слайд
   5.2 Мета, об’єкт та предмет дослідження
   5.3 Актуальність роботи
   5.4 Найбільш подібні до аналогів
   5.5 Порівняння з аналогами
   5.6 Технічне завдання
   5.7 Програмні та технічні засоби реалізації
   5.8 Архітектура телеграм-боту
   5.9 Діаграма прецедентів
   5.10 Діаграма діяльности
   5.11 Зовнішний вигляд телеграм-боту
   5.12 Висновки
   5.13 Апробація результатів дослідження

6. Дата видачі завдання «11» квітня 2022р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Підбір науково-технічної літератури	16.02.22 - 12.04.22	Виконано
2	Аналіз та дослідження існуючих аналогів	12.04.22 - 15.04.22	Виконано
3	Проектування архітектури програмного засобу	15.04.22 - 18.04.22	Виконано
4	Моделювання об’єкту проектування	18.04.22 - 21.04.22	Виконано
5	Вступ, висновки, реферат	21.04.22 - 24.04.22	Виконано
6	1 розділ	24.04.22 - 27.04.22	Виконано
7	2 розділ	27.04.22 - 30.04.22	Виконано
8	Розробка фунцкіональності бота	30.04.22 – 05.05.22	Виконано
9	3 розділ	05.05.22 – 10.05.22	Виконано
10	Розробка обов‘язкових демонстраційних матеріалів	10.05.22 – 13.05.22	Виконано
11	Попередній захист роботи	25.05.22	
12	Здача роботи	03.06.22	

Студент _ _____
(підпис) (прізвище та ініціали)

Керівник роботи _____ _____
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Текстова частина бакалаврської роботи 83 с., 34 рис., 1 табл., 26 джерел.

ПОКРАЩЕННЯ ПРОЦЕСУ РЕАЛІЗАЦІЇ ПРОДУКТІВ ПОБУТОВОЮ ХІМІЇ, ПРЯМИЙ ПРОДАЖ ТОВАРУ ЧЕРЕЗ ТЕЛЕГРАМ-БОТА, ТЕЛЕГРАМ-БОТ ROZETKA.UA, ТЕЛЕГРАМ-БОТ ПРИВАТБАНК, ВЕБ-СТОРІНКА ДЛЯ ОФОРМЛЕННЯ ЗАМОВЛЕННЯ.

Об'єкт дослідження – процес реалізації товарів побутової хімії за допомогою застосування Телеграм-ботів.

Предмет дослідження – телеграм-бот на мові програмування JavaScript.

Мета роботи – покращення процесу реалізації товарів побутової хімії за допомогою розробленого телеграм-боту мовою JavaScript.

В даній дипломній роботі проведено аналіз використаних у сфері бізнесу таких телеграм-ботів, як телеграм-бот від Розетки, телеграм-бот від ПриватБанку та інших.

Особливістю розроблюваного телеграм-боту, у цій дипломній роботі, є можливість продавати власнику бізнесу свої товари, використовуючи перспективний та популярний месенджер Телеграм з його інструментом – ботом. За основу для розробки була взята мова програмування TypeScript (Розширений JavaScript), програмна платформа Node.js для створення серверу та СУБД PostgreSQL в якості збереження даних про оформлені замовлення клієнтів.

Отже, розроблен та описан телеграм-бот, завданням якого є покращення процесу реалізації продуктів побутової хімії.

Даний телеграм-бот може бути використан як приклад для різних типів бізнесів, які бажають збільшити власну прибутковість та не відставити від сучасних технологій і можливостей.

Галузь використання – застосування телеграм-ботів у бізнесі.

ЗМІСТ

РЕФЕРАТ ... 6
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ... 9
ВСТУП ... 10
РОЗДІЛ 1 ДОСЛІДЖЕННЯ РОЗВИТКУ, МОЖЛИВОСТЕЙ ТА АКТУАЛЬНОСТІ ВЕДЕННЯ БІЗНЕСУ В TELEGRAM ... 12
1.1 Про Telegram ... 12
1.2 Розвиток аудиторії Telegram ... 14
1.3 Телеграм в цифрах ... 16
1.4 Особливості Telegram та відмінності від інших месенджерів ... 17
1.5 Інструменти Telegram ... 18
1.6 Архітектура Telegram ... 27
1.7 Застування Telegram у бізнесі ... 29
1.8 Причини використання Telegram-ботів у бізнесі ... 30
1.9 Типи та варіанти використання Telegram-ботів ... 33
1.10 Приклади використання телеграм-ботів у бізнесі ... 36
РОЗДІЛ 2 РОЗРОБКА СТРУКТУРИ ТЕЛЕГРАМ-БОТУ З МЕТОЮ ПОКРАЩЕННЯ ПРОЦЕСУ РЕАЛІЗАЦІЇ ПРОДУКТІВ ПОБУТОВОЇ ХІМІЇ МОВОЮ JAVASCRIPT ... 39
2.1 Завдання телеграм-боту з метою покращення процесу реалізації продуктів побутової хімії JS ... 39
2.2 Про TypeScript ... 41
2.3 Моделювання об’єкту проектування ... 42
2.3.1 Діаграма прецедентів ... 42
2.3.2 Розробка діаграми прецедентів для телеграм-боту ... 46
2.3.3 Діаграма діяльності ... 46
2.3.4 Розробка діаграми діяльності для телеграм-бота ... 50
2.4 Алгоритм роботи телеграм-боту ... 51
2.5 Про PostgreSQL ... 52
2.6 Архітектурна структура телеграм-боту ... 54

РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТЕЛЕГРАМ-БОТА ТА ТЕСТУВАННЯ 59
3.1 Обгрунтування вибору підходу до реалізації 59
3.2 Набір інструментів для реалізації телеграм-боту 59
3.3 Функціонал телеграм-боту з метою покращення процесу реалізації продуктів побутової хімії 63
3.4 Зовнішній вигляд розробленого телеграм-боту 76
3.5 Тестування телеграм-боту з метою покращення процесу реалізації продуктів побутової хімії 79
ВИСНОВКИ 82
ПЕРЕЛІК ПОСИЛАНЬ 84
ДОДАТОК А 88
ДОДАТОК Б 93

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

IM - Instant Messages;

LLC - Limited Liability Company;

IT - Information Technology;

PR - Public Relations;

SMS - Short Messaging Service;

AES - Advanced Encryption Standard;

RSA - Rivest Shamir Adleman;

GNU - General Public License;

API - Application Programming Interface;

CRM - Customer Relationship Management;

TS - Type Script;

JS - Java Script;

UML - Unified Modeling Language;

ACID - Atomicity, Consistency, Isolation, Durability;

SQL - Structured Query Language;

MVP - Minimum Viable Product;

СУБД - Система Управління Базами Даних;

JSONB - JavaScript Object Notation Binary;

МП - Мова Програмування;

UI - User Interface;

БД - База Даних;

ПЗ - Програмне Забезпечення

ВСТУП

Обґрунтування вибору теми та її актуальність: під час ведення прибуткового бізнесу завжди потрібно стежити за рівнем технологій та трендів, щоб він, як мінімум, не змінював свій статус-кво.

Через стрімке поширення додатку Telegram, який залучує усе більше активних користувачів щодня, відкриваються нові можливості для бізнесменів, які бажають збільшити прибутковість власних бізнесів. Як приклад, моя родина займається виготовленням та продажом товарів побутової хімії через сторінку в Instagram, виконуючи роль невеликого інтернет-магазину. Завдяки цій сторінці ми реалізуємо до продажу певний об'єм виготовлених товарів, однак моя родина хотіла би покращити цей результат. Враховуючи це, мною було прийнято рішення розробити Telegram-бота для покращення реалізації продуктів побутової хімії, які ми виготовляємо.

Ступінь вивчення проблеми: На даний момент існує досить багато бізнесів, які ведуть свою діяльність в інтернеті. Але, небагато з таких бізнесів використовують телеграм-ботів, а ще менше використовують їх для продажу своїх товарів.

Для бізнесу Telegram застосовується досить успішно. Це обумовлюється тим, що можливості месенджера дозволяють швидко і ефективно ділитися новинами, акціями і розсилками з великою кількістю користувачів. Боти зменшують навантаження на менеджерів, беручи на себе виконання монотонних завдань, проте, звісно, не зможуть їх повністю замінити.

Об'єкт дослідження – процес реалізації товарів побутової хімії за допомогою застосування Телеграм-ботів.

Предмет дослідження – телеграм-бот на мові програмування JavaScript.

Мета роботи –покращення процесу реалізації товарів побутової хімії за допомогою розробленого телеграм-боту мовою JavaScript.

Завданням роботи є розробка Telegram-бота, що покращить реалізацію продуктів побутової хімії.

Методика дослідження: Перш за все, потрібно було визначити правильну площадку для розширення бізнесу. У нашому випадку, потрібно було визначити, чи є Telegram тією самою «правильною площадкою».

За повідомленнями, месенджер сьогодні має понад 550 мільйонів активних користувачів на місяць з різних країн. За кількістю завантажень Telegram входить до десятки найпопулярніших соціальних мереж у світі. Це дуже велика кількість потенційних клієнтів для власників малих та середніх бізнесів.

З технічного боку, бот буде написаний за допомогою мови програмування JS та програмної платформи Node.js для розширення стандартних можливостей JavaScript. У якості БД буде використана СУБД PostgreSQL.

Для клієнтської сторони, взаємодія з ботом буде зручна завдяки привітному інтерфейсу від Telegram. Клієнт зможе «спілкуватися» з ботом за допомогою повідомлень та передумовлених кнопок.

Наукова новизна роботи: Таким чином, наукова новизна полягає в створенні Telegram-боту, котрий буде призначений для покращення об’єму продажу товарів, а не тільки інформування. Наразі така практика дуже рідка та саме тому це має вигляд «свіжого» погляду щодо використання ботів у Telegram, використовуючи його можливості.

Практична значущість результатів: Даний бот може бути використаний як основний приклад для інших Telegram-ботів у сфері бізнеса, де власник бізнеса бажає розширити розпізнаваність власних товарів та збільшення прибутку своєї бізнес діяльності.

РОЗДІЛ 1 ДОСЛІДЖЕННЯ РОЗВИТКУ, МОЖЛИВОСТЕЙ ТА АКТУАЛЬНОСТІ ВЕДЕННЯ БІЗНЕСУ В TELEGRAM

1.1 Про Telegram

Telegram - це безкоштовний кроссплатформенний хмарний сервіс миттєвих повідомлень (IM). Його розробили в 2013 році браття Микола і Павло Дурови. Раніше вони заснували російську соціальну мережу "ВКонтакте", яку в 2014 році вони покинули. Павло продав свою частку, що залишилася в "ВКонтакте" і після цього він покинув Росію. Основою месенджера є протокол MTProto, який створив Микола Дуров, у свою чергу, Павло ж надав фінансову підтримку та інфраструктуру за допомогою свого фонду Digital Fortress. По заявленню месенджеру Telegram, його кінцева мета не є отримання прибутку, проте він не має таку структуру як у некомерційних організацій.

Telegram зареєстрований як компанія на Британських Віргінських островах і як LLC в Дубаї. Інформація про оренду офісів або про юридичних особів, яких Телеграм використовує для оренди, не розголошується, посилаючись на необхідність "захистити команду від непотрібного впливу" і захистити користувачів від урядових запитів даних. В 2014 Павло Дуров та його команда Telegram зробила Берлін, своєю штаб-квартирою, проте на початку 2015 року вони переїхали в Дубай та інші юрисдикції через неотримання посвідки на проживання в Німеччині для всіх членів команди. На той час команда містила 15 основних членів.

Telegram надає наскрізні зашифровані голосові та відеодзвінки, а також додаткові наскрізні зашифровані "секретні" чати. Через те, що хмарні чати і групи зашифровані між додатком і сервером, сторонні користувачі в мережі та інтернет-провайдери не можуть отримати доступ до даних, проте сам сервер Telegram, звісно, може. Користувачі мають можливість надсилати текстові та голосові повідомлення, здійснювати відео- та голосові дзвінки, а також обмінюватися

необмеженою кількістю зображень, документів (2 ГБ на файл), місцеположень користувачів, анімованих стікерів, контактів та аудіофайлів.

14 серпня 2013 року Телеграм був запущений для iOS. На Android був реліз у жовтні того ж року. Щоб зменшити часте навантаження на дані, сервери Telegram розподілені по всьому світу, з п'ятьма центрами обробки даних в різних регіонах. Головний операційний центр базується в Дубаї в Об'єднаних Арабських Еміратах.

Різні клієнтські програми доступні для настільних і мобільних платформ, включаючи офіційні програми для Android, iOS, Windows, macOS і Linux (хоча для реєстрації потрібен пристрій iOS або Android і робочий номер телефону). Існують також два офіційних додатки-близнюки Telegram web, WebK і WebZ, і численні неофіційні клієнти, що використовують протокол Telegram. Окрім сервева, який є закритим і пропрієтарним, всі офіційні компоненти Telegram мають відкритий вихідний код.

У січні 2021 року кількість активних користувачів Telegram перевищила 500 мільйонів на місяць.

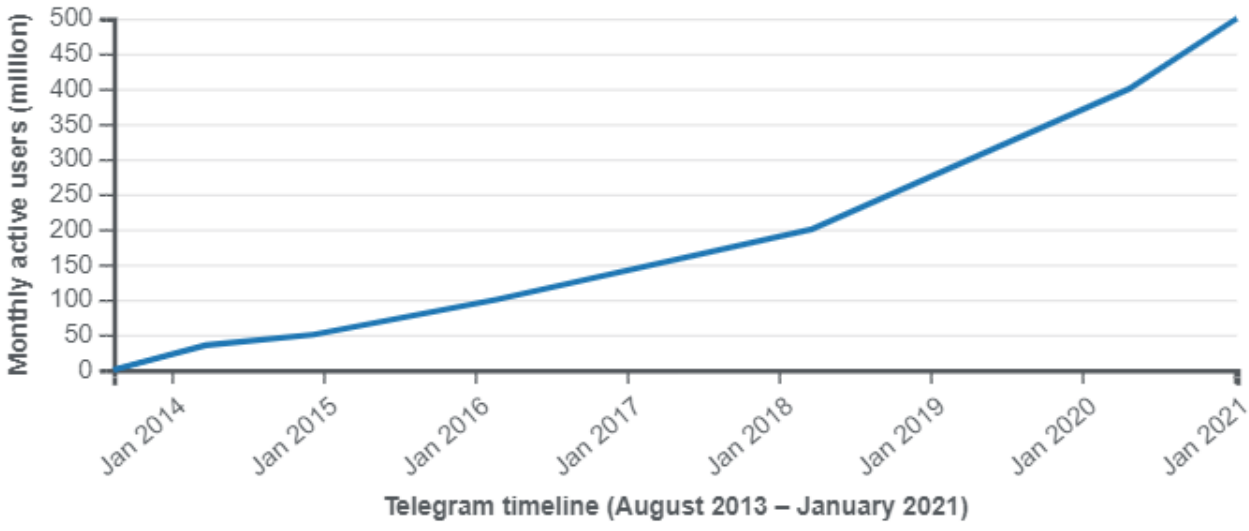


Рисунок 1.1 – Кількість щомісячних активних користувачів з 2014 року

В січні 2021 року Telegram був найзавантажуванішим додатком у світі. Станом на кінець серпня 2021 року додаток мав 1 мільярд завантажень по всьому світу.

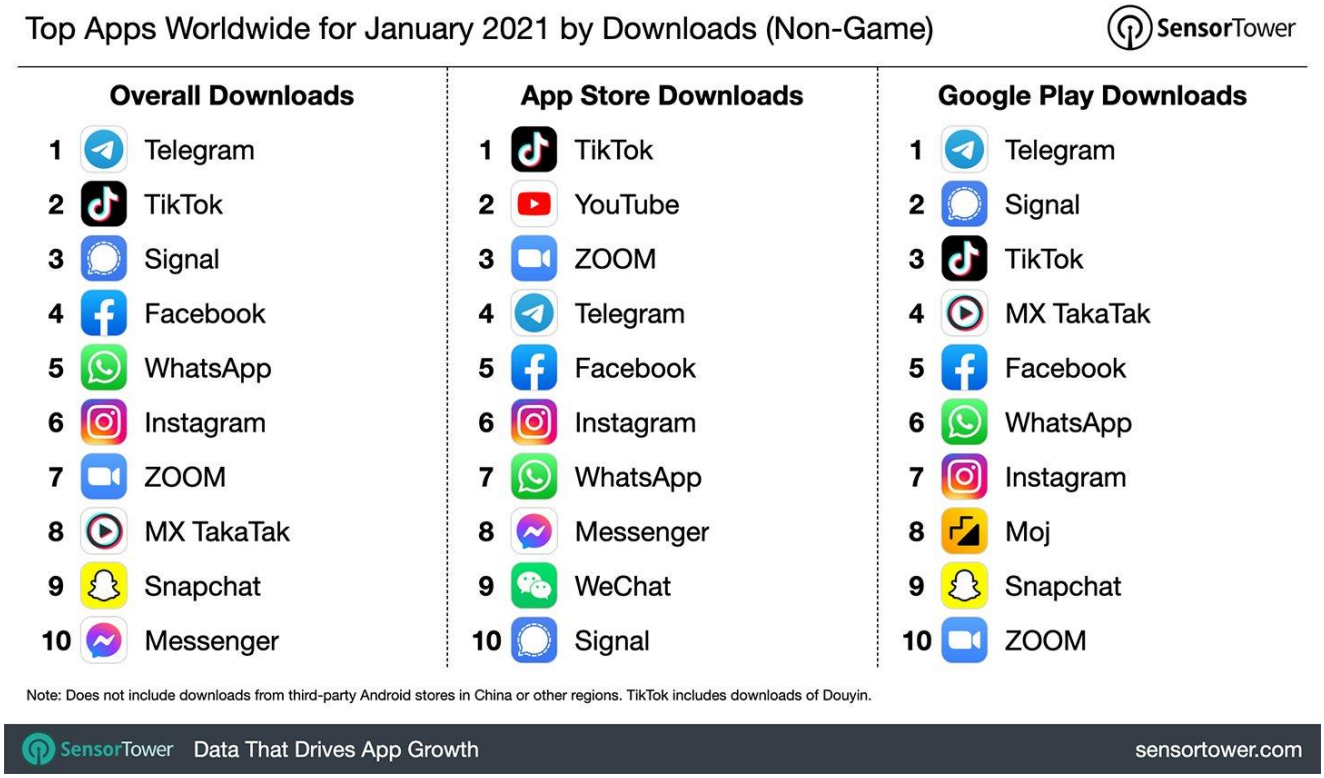


Рисунок 1.2 – Найзавантажуваніші додатки у світі за 2021 рік

1.2 Розвиток аудиторії Telegram

Перший етап

У жовтні 2013 року Telegram оголосив, що месенджер щодня маэ 100 000 активних користувачів. 24 березня 2014 року Telegram оголосив, що число реєстрацій досягло 35 мільйонів і месенджер вже мав 15 мільйонів активних користувачів щодня. Через певні плани уряду Південної Кореї щодо стеження, на жовтень 2014 прийшовся потужний потік до Telegram від громадян цієї країни. Станом на грудень 2014 року Telegram оголосив, що у нього 50 мільйонів активних

користувачів, які генерують 1 мільярд повідомлень щодня, і що щотижня кількість реїстрацій нових користувачів становить 1 мільйон. За п'ять місяців трафік подвоївся до 2 мільярдів щоденних повідомлень. У вересні 2015 року, Telegram оголосив, що додаток налічує вже 60 мільйонів активних користувачів і трафік повідомлень становить вже 12 мільярдів.

Другий етап

Станом на лютий 2016 року Telegram оголосив, що має 100 мільйонів активних користувачів на місяць, при цьому кожен день реєструється 350 000 нових користувачів, які щодня відправляють 15 мільярдів повідомлень. На грудень 2017 року кількість активних користувачів Telegram прийшлося вже 180 мільйонів на місяць. У березні 2018 року кількість активних користувачів Telegram досягла 200 мільйонів на місяць. 14 березня 2019 року Павло Дуров заявив, що "за останні 24 години в Telegram зареєструвалося 3 мільйони нових користувачів". Він не уточнив, чим був спричинений цей потік нових реєстрацій, проте цей період відповідав тривалому технічному збою в Facebook і його сімействі додатків, включаючи Instagram.

Третій етап

С таном на жовтень 2019 року за даними Комісії з цінних паперів і бірж США, число щомісячних користувачів Telegram становить 300 мільйонів чоловік по всьому світу. 24 квітня 2020 року Telegram оголосив, що його число активних користувачів досягло 400 мільйонів на місяць. 8 січня 2021 року Дуров написав у своєму блозі, що Telegram досяг "близько 500 мільйонів" активних користувачів на місяць. 5 жовтня 2021 року Telegram набрав понад 70 мільйонів нових користувачів в результаті збою, який торкнувся Facebook і його філії.

За повідомленнями, месенджер сьогодні має понад 550 мільйонів активних користувачів на місяць з різних країн. За кількістю завантажень Telegram входить до десятки найпопулярніших соціальних мереж у світі.

1.3 Телеграм в цифрах

Користувачами Telegram найчастіше є освічені платоспроможні чоловіки у віці від 18 до 40 років. Жінки використовують даний месенджер рідше: їх частка становить трохи більше 30% від загального числа користувачів. Більша частина користувачів працює в IT-сфері, потім йде виробництво, а після нього маркетинг і PR.

Зручність і таємниця листування найчастіше є причиною використання Telegram для його користувачів. Крім того, користувачі вважають за краще дізнаватися новини з Telegram-каналів, а не з інших джерел. Статистика підказує, що більше 80% користувачів ведуть в месенджері особисте листування, а близько чверті ще й ділову. 42% користувачів використовують боти Телеграм, відзначаючи також їх зручність і практичність.

Як і в 2019 році, лідирує вікова група «25-34 роки», проте загальний розподіл став набагато більш «згладженим» — користувачів «18-24», «35-44» і «45-64» тепер приблизно однакова кількість. Ще два роки тому аудиторії старше 35 років було 27%, тепер же їх вже майже 40%. Це є помітним зростанням і говорить про те, що тепер месенджером користуються не тільки early adopters в особі міленіалів, але і більш старше покоління.

Щодо користування месенджером в Україні, то в українському Telegram лідирують Київська, Харківська, Одеська та Дніпропетровська області — разом їх жителі складають дві третини опитаних, ще третина припадає на решту регіонів країни. Аналогічна ситуація спостерігається і серед міст -жителів Києва серед опитаних близько третини, потім йдуть жителі Харкова, Одеси, Дніпра і Львова.

1.4 Особливості Telegram та відмінності від інших месенджерів

Telegram можна використовувати не тільки як засіб спілкування з друзями або знайомими. Його також використовують для спілкування з потенційним замовником, для ведення робочого чат, як засіб оперативного документообігу і додатковий рекламний канал. Для цього Telegram має ряд незаперечних переваг:

- Висока швидкість роботи;
- Популярність серед клієнтів з високим рівнем доходів;
- Простота використання;
- Можливість спілкуватися, використовуючи не тільки смартфон;
- Відсутність зовнішніх втручань в комунікацію з користувачем.

Павло Дуров хотів зробити безпечний для спілкування месенджер. Ця ідея прийшла до нього ще в 2011 під час спілкування з фахівцями підрозділу спеціального призначення.

Відмінності месенджера від інших додатків полягають в наступному:

- Можливість передачі файлів будь-якого формату і розміру до 2 ГБ (архіви, таблиці, документи, аудіо- та відео файли, тощо.);
- Хмарне збереження інформації;
- Можливість синхронізації з різними пристроями. Telegram не обмежує користувача за кількістю сесій і тому допускає вхід з декількох пристроїв одночасно;
- Швидка доставка повідомлень та файлів;
- Створення чатів з великою кількістю учасників;
- Високий рівень безпеки. Микола і Павло Дурови пообіцяли винагороду в 200000$ тому, хто зможе зламати месенджер і прочитати їх листування. На даний момент у цьому ніхто не досяг успіху;

Telegram має функції у системі безпеки, які дозволяють налаштувати самовидалення листування. Також є можливість вести конфіденційні бесіди, оскільки повідомлення будуть видалятися після прочитання адресатом.

1.5 Інструменти Telegram

Акаунт

Для реєстрації акаунту необхідний пристрій із iOS або Android. Акаунти у Telegram прив'язані до телефонних номерів і проходять перевірку за допомогою SMS-коду. Користувачі мають можливість використовувати месенджер одночасно з декількох пристроїв і отримувати повідомлення на всі з них. Усі пристрої, які були підключені користувачем можуть бути видалені всі відразу або окремо їм особисто. У налаштуваннях прив'язаний номер можна змінити в будь-який час, і при цьому усі контакти користувача автоматично будуть сповіщені про те, що користувач змінив старий номер на новий.

Крім того, при бажанні користувача, він може не розкривати свій номер телефону іншим користувачам, завдяки псевдоніму, який цей користувач має придумати сам. Це дозволяє йому відправляти і отримувати повідомлення на свій акаунт без розкриття номера телефону. Створений акаунт в Telegram може бути видалений користувачем в будь-який момент часу. За замовчуванням, якщо акаунт знаходиться у стані бездіяльності 6 місяців, то він видаляється автоматично. При бажанні період бездіяльності можна змінити з 1 місяця до 12 місяців максимум. Користувачі також можуть змінити відображення їх статусу присутності в месенджеру у той чи інший час через налаштування. Так наприклад, вони мають можливість точні тимчасові мітки "останній раз бачили" змінити на більш приховане відображення їх статусу, такий як "останній раз бачили нещодавно".

Telegram використовує однофакторну аутентифікацію на основі SMS для входу в систему, яка є методом аутентифікації за замовчуванням. Для входу до власного акаунту потрібен одноразовий пароль, який відправляється по SMS на номер телефону користувача. Користувачі також мають можливість додати пароль як форму двоетапної перевірки для покращення безпеки власного облікового запису.

Хмарні повідомлення

Завдяки тому, що повідомлення в Telegram за замовчуванням знаходяться в хмарі, вони є доступні на будь-якому з підключених пристроїв користувача. Користувачі можуть обмінюватися із іншими користувачами будь-якими файлами до 2 ГБ: фотографіями, відео- та аудіоповідомленнями, документами, таблицями, архівами, тощо. Користувачі мають нагоду надсилати повідомлення групам користувачів до 200 000 учасників або індивідуально. Відправлені повідомлення можуть бути видалені в будь-який час з обох сторін та можуть бути відредаговані протягом 48 годин після їх відправки. Для повідомлень у всіх чатах, включаючи групи і канали, можна налаштувати автоматичне видалення через певні періоди часу, наприклад: 24 години, 7 днів або місяць. Проте для того, щоб це застосовувати тільки до повідомлень, потрібно включити таймер автоматичного видалення.

Telegram має чернетку для повідомлень, яка синхронізуються між усіма пристроями користувача, наприклад, коли він починає вводити повідомлення на одному пристрої, то потім може закінчити своє повідолмення у тому ж діалозі на іншому. В області редагування чернетка буде зберігатися на будь-якому пристрої до тих пір, поки вона не буде видалена або відправлена. Користувач може відсортувати всі чати, включаючи групи і канали по папкам, які він сам може встановити. Також користувачі мають можливість запланувати відправку повідомлень в діалозі з іншим користувачем, коли він підключається до мережі. Telegram також підтримує можливість імпортувати історію чатів користувачів з

WhatsApp, Line і Kakaotalk, включаючи усі повідомлення і медіа, завдяки переносимості даних.

Як згадувалося раніше, за допомогою протоколу MTProto шифрується передача повідомлень на сервери Telegram, в той час як в "секретних чатах" використовується наскрізне шифрування, яке побудоване на основі того ж протоколу. Враховуючи політику конфіденційності Telegram, яка наголошує, що всі дані зберігаються в суворому зашифрованому вигляді, а ключі шифрування в кожному випадку зберігаються в декількох інших центрах обробки даних в різних юрисдикціях. Виходячи з цього, фізичні зловмисники або місцеві інженери не зможуть отримати доступ до даних користувачів. Що стосується локальної бази даних повідомлень Telegram, то вона не є зашифрована за замовчуванням. Звісно не всі, про деякі клієнти Telegram дозволяють користувачам шифрувати локальну базу даних повідомлень. Робиться це за допомогою задавання кодової фрази.

Користувачі Telegram також мають нагоду ділитися своїм місцеположенням в чаті протягом певного часу. Наприклад, це може бути 15 хвилин, година або 8 годин. Якщо декілька користувачів діляться своїм місцеположенням в групі, вони відображаються на зручній для користувача інтерактивній карті, який її бачить. Показ свого місцеположення може бути припинено в будь-який час.

Секретні чати

В так званих «секретних чатах» повідомлення можуть бути відправлені з шифруванням від користувача до користувача. Ці повідомлення шифруються за допомогою протоколу MTProto служби, який згадувався раніше. На відміну від звичайних повідомлень в чаті з іншим корристувачем, доступ до повідомлень, відправлених в секретному чаті є тільки на 2 пристроях. Перший пристрій – це той на якому був ініційований секретний чат, а другий – той на якому був прийнятий секретний чат. Повідомлення, відправлені в секретних чатах можуть самознищуватися і бути видалені в будь-який час при бажанні користувача.

Алгоритм створення «секретного чату» полягає в наступному: чат повинен бути ініційован на одному пристрої й прийнятий за допомогою запрошення на іншому. Після цього відбувається обмін ключами шифрування між користувачами для сеансу. Користувачі в секретному чаті можуть переконатися, що їх діалог є зашифрованим, порівнявши зображення, на яких можна побачити відбитки їх відкритих ключів.

Згідно з офіційною заявою Telegram, «секретні чати» підтримують Perfect Forward Secret (властивість деяких протоколів для погодження ключей) з грудня 2014 року. Якщо ключ шифрування використовувався більше 100 разів або використовувався більше тижня, то він змінюється на інший. Старі ж ключі шифрування знищуються назавжди.

«Секретні чати» доступні тільки в програмних клієнтах для Android, iOS і macOS.

Канали

З вересня 2015 року Telegram додав такий інструмент як «канали». Канали - це форма одностороннього обміну повідомленнями, при якій повідомлення можуть відправляти тільки адміністратори, а інші користувачі ні.

Будь-який користувач може створити свій канал та підписуватися на інші. Канали можуть бути створені для трансляції повідомлень різного характеру контенту для необмеженої кількості абонентів. Задля того, щоб приєднатися до каналу міг будь-хто з користувачів, хто цього забажає, канали можуть бути загальнодоступними, мати псевдонім і постійну URL-адресу.

Користувачі, які приєднуються до каналу, можуть переглядати всю історію повідомлень за весь час існування цього каналу. Користувачі можуть приєднуватися до каналів в будь-який час, так і покинути їх в будь-який час. Залежно від налаштувань каналу, які робить його адміністратор, повідомлення можуть бути підписані ім'ям каналу або ж ім'ям адміністратора.

Користувачі, які не є адміністраторами, не можуть бачити інших користувачів, які підписалися на канал. Оскільки кожне повідомлення має свій власний лічильник переглядів, який показує, скільки користувачів переглянуло це повідомлення, адміністратор каналу може переглядати статистику активності каналу і робити висновки щодо зробленого їм контенту: як він сприяє розширенню аудиторії. А з травня 2019 року творець каналу може додати дискусійну групу, в якій автоматично публікуються повідомлення в каналі для спілкування передплатників. Це дозволяє коментувати повідомлення в каналі.

Bloomberg у грудні 2019 року перевів свій сервіс розсилки новин з бази месенджера WhatsApp на Telegram. Це сталося через те, що WhatsApp заборонив масові і автоматичні повідомлення. Bloomberg намагається розширити свою аудиторію за межами США.

З грудня 2021 року були введені нові функції захисту контенту, які дозволяють адміністраторам приватних груп і каналів відключати пересилання повідомлень, скріншоти і збереження даних у своїх спільнотах.

Відео- та голосові виклики

В кінці березня 2017 року Telegram представив свої власно розроблені наскрізні зашифровані голосові виклики. З'єднання є одноранговим, коли це можливо, а в іншому випадку починає використовуватися сервер, який є найближчим до клієнта. Telegram заявляє, що існує певна нейронна мережа, яка працює над вивченням різних технічних параметрів виклику для забезпечення найкращої якості обслуговування для майбутнього використання.

В грудні 2020 року Telegram додав групові голосові чати. Будь-який адміністратор каналу або групи може запустити спеціальний голосовий чат, який буде відкритий для всіх учасників і буде таким, навіть якщо їм ніхто не буде користуватися на той момент часу. Адміністратори мають можливість відключати певних учасників, а також створювати посилання для запрошень до чату для людей,

які були відключені за замовчуванням від можливості приєднатися до голосового чату.

Щоб висловити своє бажання виступити у чаті, учасники, як приклад, можуть використовувати кнопку "підняти руку". Мобільна версія містить функцію push-to-talk, а також за допомогою поєднання клавіш можна відключити і включити звуку на робочому просторі Telegram. Адміністратори каналів або груп можуть приєднатися до свого каналу або групи, приховавши свій акаунт. Для голосових чатів є можливість запису за допомогою червоної крапкою, яка відображається як попередження під час періоду запису для користувачів.

На другому місяці весни 2020 року Telegram оголосив, що до кінця року додасть можливість здійснення групових відеодзвінків. 15 серпня 2020 року Telegram додав відеодзвінки з наскрізним шифруванням.

На початку літа 2021 року Telegram додав як стандартну функцію здійснення групових відеодзвінків для всіх своїх клієнтів. Користувачі мають можливість транслювати відео зі своєї камери, ділитися своїм екраном або робити і те, і інше одночасно.

Telegram заявив, що наразі групові дзвінки обмежені 30 людьми, проте найближчим часом ліміт буде збільшено. Групові виклики підтримують вибірковий загальний доступ до екрану, розділений перегляд екрану і покращене усунення шуму.

У середині літа 2021 року в оновленні Telegram додалася можливість перегляду потокового відео до тисячі осіб, а прямі трансляції були оновлені і тепер могли містити необмежену кількість учасників.

Боти

Червень 2015 року війшов в історію Telegram, як місяць, коли месенджер запустив спеціальну платформу для створення ботів, яка була орієнтована на сторонніх розробників.

Боти - це облікові записи Telegram, керовані програмами. У них є можливість відповідати на повідомлення або згадки, вони можуть бути запрошені в групи і можуть бути інтегровані з іншими програмами. Вони також можуть приймати онлайн-платежі за допомогою кредитних карт або Apple Pay.

Tweakers (голландський веб-сайт) зробив заяву, що запрошений бот теоритично може читати всі повідомлення в групі, якщо той, хто контролює бота, пізніше непомітно для всіх змінить опції доступу. Telegram на це відповів, що компанія розглядає можливість створення спеціальної функції, яка б повідомляла про таку зміну статусу доступу для відповідної групи.

Також існують, так звані, «вбудовані боти», якими користувач може користуватися з будь-якого екрану чату. Для активації такого бота, користувач має ввести ім'я бота і запит в поле повідомлення. Після цього бот пропонує свій контент. Користувач може вибрати те, що йому до вподоби з цього контенту і після, поділитися цим у чаті з іншими користувачами.

До функціональності ботів також входе можливість обробляти транзакції від таких сервісів, як: Paymentwall, Яндекс.Money, Stripe, Ravepay, Razorpay, QiWi і Google Pay для різних країн.

У Telegram також є своя ігрова платформа, яку боти можуть використовувати. Ця платформа використовує HTML5 і тому ігри завантажуються як звичайні веб-сторінки. Ігри працюють від iPhone 4 і новіше, і від Android 4.4 і новіше.

Після квітневого оновлення Payments 2.0 у 2021 року, стало можливим здійснювати грошові транзакції за допомогою ботів в будь-якому чаті, використовуючи такі сервіси як: ECOMMPAY, LiqPay, CLICK, Payme, Tranzoo і Sberbank.

У червневому оновленні 2021 року в у бота з'явилося нове меню, в якому користувачі можуть переглядати і відправляти спеціальні команди, перебуваючи в чаті з ботом.

Миттєвий перегляд

Інструмент миттєвого перегляду - це можливість перегляду веб-статей з нульовим часом завантаження сторінки. Користувачі Telegram можуть читати статті з блогів або ЗМІ в єдиному і зручному для читання вигляді за допомогою Instant View. Сторінки Instant View підтримують текст і мультимедіа будь-якого типу і працюють, навіть якщо відкритий через миттєвий перегляд веб-сайт не був оптимізований для мобільних пристроїв.

До того ж, сторінки Instant View кешуються на серверах Telegram і є надзвичайно легкими. Через це вони миттєво завантажуються майже при будь-якому підключенні.

Телеграф

Telegraph - це інструмент публікації, який використовується для створення форматованих повідомлень з фотографіями та вбудованими носіями. Цей інструмент розроблен в максимально простому стилі, сторінки статей не мають жодних елементів керування. На сайті кожна стаття є окремою. Можливості об'єднувати статті в ієрархії або групи відсутні. Для кожної статті автор задає назву і підзаголовок, якщо це потребує сам автор, який зазвичай використовується для імені того, хто написав статтю. У назві статті вказується дата найпершої публікації. На цю дату автор ніяк вплинути не може.

Параметри форматування тексту є також дуже простими: підтримка двох рівнів заголовків, однорівневих списків, жирного шрифту, курсиву, лапок і гіперпосилань. Автори мають можливість завантажувати відео та зображення на сторінку з обмеженням до 5 МБ. Сервіс не забороняє вставляти контент з YouTube, Vimeo або Twitter в статтю через посилання, яке автор додає до статті.

URL - адреса автоматично генерується з назви статті, коли вона вперше публікується. Нелатинські символи транслітеруються, до адреси додається дата публікації, а прогалини замінюються дефісами.

Стікери та емодзі

Telegram має у собі більше 20 000 різноманітних стікерів. Стікери - це певні хмарні зображення, які можна використовувати замість повідомлень з високою роздільною здатністю та які призначені для створення більш виразних смайликів.

Користувачеві, при введенні смайлика, пропонується замість нього відправити асоціативний стікер. Стікери створюються в наборах, які ще називаються по-іншому "пакетами". До одного смайлика можна запропонувати декілька різних стікерів.

За замовчуванням, Telegram поставляється лише з одним пакетом стікерів, проте користувачі можуть встановлювати додаткові пакети стікерів, які створюються іншими авторами, які зазвичай також є користувачами Telegram.

Пакети стікерів, які встановлені на одному клієнті, автоматично стають доступними на всіх інших клієнтах. Формат файлу WebP використовують для зображення стікерів, адже він краще оптимізований для передачі через Інтернет. На всіх клієнтах Telegram також є підтримка анімованих смайликів - емодзі.

Ідентифікація в реальному житті

У середині літа 2018 року була представлена онлайн-система управління ідентифікацією та авторизації від Telegram під назвою Telegram Passport. Ця платформа була розроблена за потребою ідентифікації в реальному житті.

Платформа працює наступним чином. Вона просить користувачів завантажити свої офіційні документи, це може бути паспортом, посвідченням особи, водійськими правами та іншим. При їх завантаженні зі сторони користувача

та подальшому його підтверджені, ці документи загружаються безпосередньо на платформу.

Telegram наголосив, що у нього немає доступу до цих даних. І у свою чергу, платформа Telegram Passport буде передавати інформацію щодо завантажених документів тільки авторизованому одержувачу. Проте, не дивлячись на це, онлайн-система піддалася критиці, адже виявилася вразливою до онлайн-атак методом bruteforce.

Опитування

Інструмент «Опитування» є доступним на всіх операційних платформах. Є два види «Опитування»: анонімне або видиме. Користувач має змогу ввести в опитування декілька варіантів, один за будь-який інший користувач може віддати свій голос.

У інструмента «Опитування» є ще режим вікторини, який можна включити за бажанням. У ньому користувач може обрати правильну, на його думку, відповідь для свого опитування і далі, дати його групі або каналу вгадати її.

Люди поблизу і групи поблизу

Завдяки даному інструменту, люди, які знаходяться поблизу один одного, можуть познайомитися один з одним й стати новими друзями. Для активації цього інструменту потрібно включити своє місцеположення на телефоні й обрати цю функцію в месенджері. Також можна створити локальну групу. Для цього потрібно додати дані про місцеположення в групі.

1.6 Архітектура Telegram

Схема шифрування

Telegram використоє схему шифрування MTProto, яка є симетричною. Як згадувалося раніше, розробник протоколу є Микола Дуров – брат Павла Дурова. Протокол заснований на обміні ключами Діффі-Хеллмана, 256-бітному симетричному шифруванні AES, і 2048–бітному шифруванні RSA.

Сервер

Telegram використовує сервери, які є централізованими. Це поширена практика серед протоколів обміну миттєвими повідомленнями. Щоб зменшити час відгуку, сервера Telegram розташовані в рядах країн по всьому світу. Як згадувалося раніше, серверне програмне забезпечення Telegram, на відміну від інших офіційних компонентів, є закритим і пропрієтарним.

Для користувачів з ЄЕЗ (Європейської економічної зони) або СК (Сполученого Королівства), загальні правила захисту їх даних зберігаються виключно на серверах в Нідерландах і призначаються лондонській компанії, яка виступає у ролі контролера цих даних.

Клієнтські програми

Більшість клієнтських програм Telegram є безкоштовними, мають відкритий вихідний код й є випущеними під ліцензією GNU (General Public License) версії 2 або 3. Деякі з клієнтських програм розроблені Telegram Messenger LLP, проте є і ті, які були розроблені спільнотою. В офіційних клієнтах є підтримка надсилання форматів будь-яких файлів. Також в офіційних клієнтах є вбудований переглядач, який має можливість підтримувати поширені формати мультимедіа, такі як:

- JPEG, PNG, WebP для зображень;
- H. 264 і HEVC для відео в контейнері MP4 і MP3;

- FLAC, Vorbis, Opus і AAC для аудіо.

У 2021 році Telegram заявив, що буде розробляти пряму збірку додатка для платформи Android. Він є доступним безпосередньо з офіційного веб-сайту Telegram. Прямий клієнт має можливість автоматичного оновлення. Скоріш за все, він буде отримувати нові версії швидше, ніж версіі додатків в Play Store і App Store. Важлива особливість цієї версії заключається у можливості перегляду груп або каналів без цензури, які неможливо переглядати з версій додатків, які розповсюджуються з Google Play і Apple Store через їх політики.

API-інтерфейси

API Telegram є загальнодоступними. За допомогою них розробники мають можливість до доступу тих же параметрів, які є і в офіційних версіях додатку Telegram, щоб, наприклад, створити власні додатки миттєвого обміну повідомлень. Так наприкінці зими 2015 року, розробники неофіційного клієнта Whatsapp+ розробили додаток Telegram Plus, який пізніше був переіменован в Plus Messenger. А вже на початку осені 2015 року, команія Samsung розробила свій додаток для миттєвого обміну повідомленнями, який був заснований на цих API.

Telegram також має окреме API, яке дозволяє створювати ботів, яких можна використовувати у різних цілях. Ботів, наприклад, можна використовувати для емуляції і відтворення старих ігор в месенджері, а також, за потреби, для різного характеру інформування користувачів.

Також, Telegram пропонує різним типам користувачів можливості у вигляді функцій для здійснення транзакцій усередині платформи, із використанням зовнішнім сервісом, наприклад, таким як Stripe.

1.7 Застування Telegram у бізнесі

Telegram має успішне застосування для бізнесу. Це передумовлено можливостями месенджера, які дозволяють ефективно, швидко й безпечно ділитися різною інформацією у вигляді новин, акцій і розсилок для великої кількості користувачів.

Як приклад, Телеграм може використовується для бізнесу в таких випадках:

- Для створення тематичного каналу, який дозволяє ділитися з великою кількістю користувачів різною тематичною інформацією;
- Для технічної і клієнтської підтримки за допомогою чату або чат-боту;
- Для створення зручних та корисних клієнтських серверів (наприклад, перевірка статусу замовлення);
- Для створення робочого чату для обміна необхідної інформації з колегами;
- Для створення ботів, які вирішать проблему однотипних нудних задач;
- Для створення каналу для продажу реклами з великою популярністю;

1.8 Причини використання Telegram-ботів у бізнесі

Telegram-бот – це особливий тип користувачів, який є не людиною, а комп’ютерною програмою, яка може обслуговувати компанії чи бренди з багатьма функціями, такими як надсилання інформації, нагадування, відтворення мелодій, замовлення тощо.

Користувачі можуть взаємодіяти з телеграм-ботом, надсилаючи повідомлення. З усіх чотирьох однорангових типів бот Telegram має багато корисних функцій для бізнесу. Telegram надає API для створення ботів для соціальних взаємодій, продуктивності, ігор та послуг електронної комерції. Крім цього, боти Telegram також можуть надавати підтримку клієнтам або збирати потенційних клієнтів, підключаючи їх до CRM, системи продажу квитків або платформи обміну повідомленнями.

Розглянемо 7 причин використовувати Telegram-ботів у бізнесі:

1. Це безкоштовно

Telegram – це безкоштовна платформа з відкритим вихідним кодом, що означає, що Telegram дозволяє будь-кому використовувати свій API та код. Завдяки цьому створення чат-бота також безкоштовне. Є безкоштовні розробники телеграм-ботів, які можуть допомогти вам створити та розгорнути вашого бота в Telegram. Боти Telegram можуть допомогти вам і вашій компанії залучити потенційних клієнтів, використовуючи його функції безкоштовно.

2. Це безпечно

Всі час від часу чують новини про порушення даних майже кожного додатка соціальних мереж. Однак Telegram є дійсно безпечним додатком завдяки додатковим перевагам шифрування даних, безпеки та конфіденційності. Те ж саме стосується і ботів Telegram, тобто кожен обмін повідомленнями між користувачем і ботом Telegram шифрується. Дотримуючись протоколу безпеки " pair-to-pair", лише відправник і одержувач можуть отримати доступ до цих повідомлень. У даному випадку телеграм-боту, цими двома є користувач і сам телеграм-бот.

3. Краще залучення

Для бізнесу або бренду, наявність Telegram-боту може стати відмінним активом для взаємодії, якщо клієнтська база компанії активно використовує Telegram. Telegram-боти можуть забезпечити дуже цікаві розмови між користувачем і ботами в Telegram. Telegram-боти можуть взаємодіяти з анімованими смайликами, стікерами та GIF-файлами.

Окрім маркетингу, деяким компаніям потрібно надавати підтримку клієнтів цілодобово, тому підприємства можуть створити чат-бот для Telegram або створити

команду підтримки клієнтів, щоб майже миттєво відповідати на запити клієнтів. Це може допомогти підприємствам підтримувати кращу взаємодію з клієнтами.

4. Доступність

Месенджер Telegram охоплює всі основні операційні системи, такі якAndroid, iOS, Mac, Linux та Windows. Крім того, він також має веб-версію, яка дозволить вам націлити ваших потенційних клієнтів у широкому масштабі. Бот Telegram не є справжнім користувачем, тому він не має обмежень, таких як непрацездатність 24/7, помилки, затримка повідомлень тощо. Тож ваш бізнес постійно активний. Завдяки цим перевагам телеграм-ботів, це може допомогти вам досягти кращого коефіцієнта конверсії.

5. Індивідуальні клавіатури

Бот Telegram може дозволити налаштувати клавіатуру користувача. Боти Telegram дозволяють його творцям налаштовувати взаємодію користувача з ботом в телеграмі, замінюючи традиційну клавіатуру опціями, пов’язаними з функціональністю цього Telegram-бота.

6. Грошові операції

Telegram є одним із перших, хто прив’язав ваш банківський рахунок до чат-месенджерів. Ця функція телеграм-ботів дозволить користувачам здійснювати фінансові операції з іншими користувачами або підприємствами на платформі. Ця функція також дозволяє зробити збір коштів гнучким процесом, оскільки за допомогою бота Telegram людина може допомогти в транзакції.

На даний момент ця функція доступна в Росії та Бразилії. Однак я вірю, що незабаром він буде доступний і в інших країнах. Таким чином, телеграм-бот може дозволити клієнту переказувати гроші або оплачувати послуги, не залишаючи платформи.

7. Електронна комерція

Понад 550 мільйонів людей є активними користувачами Telegram. Рекламна активність через Telegram, може допомогти вам створити миттєвий електронний магазин, де клієнти зможуть швидко отримувати пропозиції, продукти чи послуги та купувати їх у першу чергу.

1.9 Типи та варіанти використання Telegram-ботів

Боти можуть поєднувати в собі різноманітний функціонал. Проте найчастіше їх використовують у цілях, які можно охарактеризувати наступними типами:

1. Роботи, які спілкуються з клієнтами

Людина відправляє повідомлення компанії, а йому відповідає бот. При цьому відповіді можуть бути різні. Є два варіанти розвитку подій в даному випадку. Перший - в компанії є оператори, які займаються обробкою заявок, що надходять. Другий - це самонавчальний штучний інтелект, який може відповідати на поширені питання.

2. Спеціальні боти для нагадувань

Тут все передбачувано - програма нагадує про різні зустрічі, доставки, візити, необхідності пройти техобслуговування автомобіля, знижки, акції і т.д.

Важливий момент: для того, щоб використовувати Telegram-боти для нагадувань в бізнесі, необхідна наявність великої бази інформації про своїх клієнтів. Тільки в цьому випадку можливий успіх.

3. Боти, що функціонують на базі web-сервісів

Це сервіси для замовлення таксі, прогноз погоди, написання відгуків, каталоги різних закладів. Все це можна використовувати в рамках Телеграм-ботів. У бізнесі активно використовують такі програми-роботи.

Доступна навіть така цікава функція: можна «повідомити» програмі своє місцезнаходження і вона покаже, які заклади знаходяться поруч.

4. Боти для самих же ботів

Оригінальний алгоритм, який можна використовувати в якості предмодерации публікацій. Наприклад, створюється два бота в Телеграмі. Перший отримує повідомлення від користувачів і направляє їх адміністратору, який переглядає листи. Потім він натисненням однієї з клавіш вирішує, яку команду відправити другому боту, публікувати чи ні.

5. Для онлайн-магазинів

В даному випадку через бота можна приймати замовлення і навіть оплату за товари. Активно цю функцію використовують сервіси доставки.

6. Для засобів масової інформації

Це такі програми, що працюють на базах web-сервісів у сфері мас-медіа. Спрямовані вони на пошук свіжих новин за певними запитами, підписку на цікаві новинні розділи, на які в автоматичному режимі відправляються нові записи по темі.

7. Для супроводу різних заходів, тренінгів або конференцій

Дуже зручні боти, які допоможуть клієнтам бути в курсі, що відбувається в конкретний момент часу. Наприклад, проходить якась конференція, а завдяки боту користувачі сервіса Telegram зможуть дізнатися, хто і на яку тему зараз веде дискусію з публікою. Також можна голосувати в реальному часі, відправляти питання ведучим і спікерам і т.д.

8. Боти-помічники в проведенні конкурсів, квестів та акцій

Алгоритм вікторин та інших заходів можна реалізувати за допомогою таких роботів. Це можна зробити навіть, якщо конкурс тривалий і користувачам необхідно щодня протягом якогось часу відповідати на питання і заробляти очки на свою користь. Потім підраховуються підсумки, хто набрав найбільше балів, той і виграв. Вручну цим всім займати досить проблематично і довго, а боти в Телеграм справляться з поставленим завданням без проблем і помилок.

9. Для сервісних IT-служб

В інформаційному просторі поки що мало інформації щодо застосування Telegram-ботів у цій сфері, але є один досить яскравий приклад.

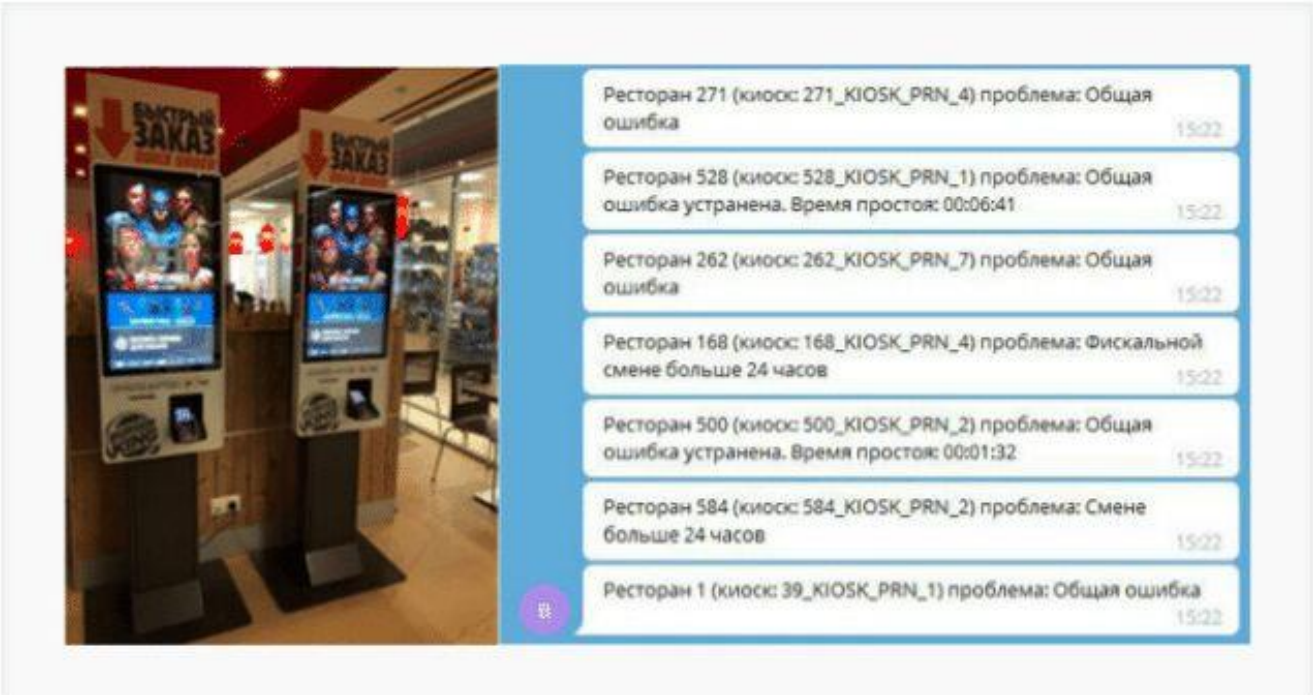


Рисунок 1.3 – Телеграм-бот для оповіщення роботи кіосків самообслуговування Burger King

Такі програми використовуються в кіосках самообслуговування Burger King і спрямовані на те, щоб зменшити час простою. Бот відповідає за оповіщення про поломки і створення звітностей про поточну роботу.

1.10 Приклади використання телеграм-ботів у бізнесі

1. Телеграм-бот від українського інтернет-магазину та маркетплейсу «Rozetka.ua». За версією SimilarWeb станом на лютий 2022 року сайт посідає 17 місце серед найвідвідуваніших в Україні.

Цей бот створений для автоматизації підтримки компанії «Rozetka.ua». Також завдяки ньому користувач має можливість дізнатись, де зараз знаходиться замовлення, відмінити його, повернути товар, дізнатися інформацію щодо оформлення нового замовлення й безпосередньо зв'язатися з одним із технічних спеціалістів.

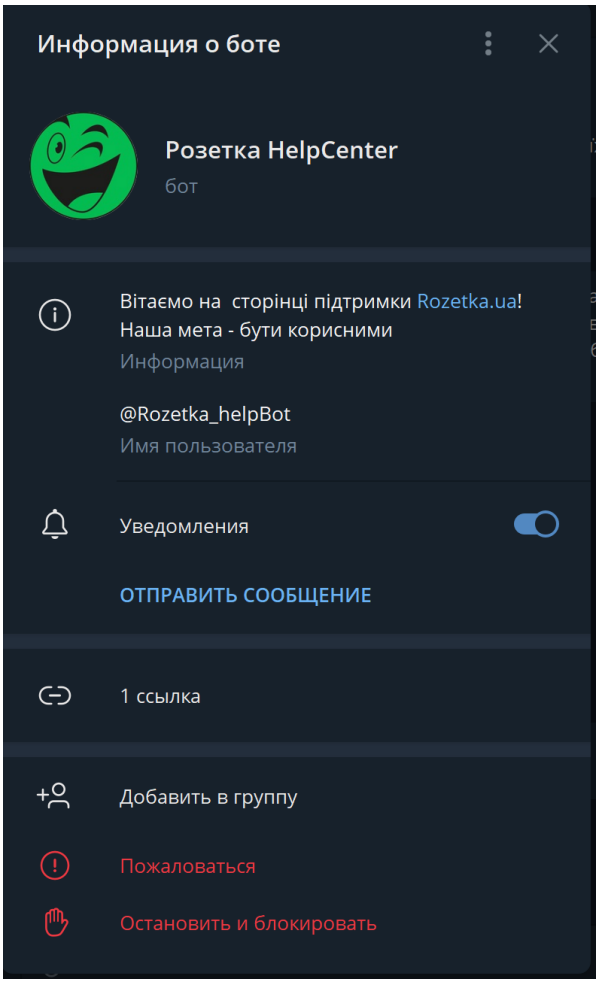


Рисунок 1.4 – Телеграм-бот від Rozetka.ua

2. Телеграм-бот від найбільшого за розмірами активів українського банку та лідеру роздрібного банківського ринку України «ПриватБанк».

Завдяки цьому боту, користувачі можуть відправляти та запитувати гроші в чаті з друзями, передивитися інформацію щодо депозиту, дізнатися актуальний курс валют, переводити грощі між власними картками, подивитися архів транзакцій, дізнатися місцезнаходження найближчого терміналу, відділення або банкомату, а також поповнити баланс мобільного рахунку.

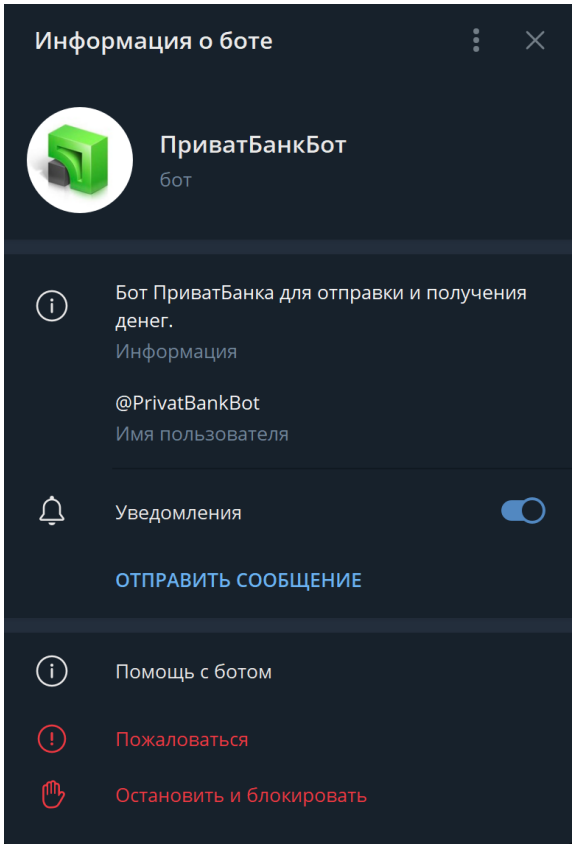


Рисунок 1.5 – Телеграм-бот від ПриватБанк

Отже, як можна побачити, телеграм-боти широко використовуються у сфері бізнеса в Україні. Великим компаніям вдалося «познайомитися» із таким інструментом бізнесу, як телеграм-бот та використати його під свої потреби. Проте, продаж товарів через телеграм-бота все ще залишається невідомим простором для ринку для безлічі бізнесів. Нажаль, під час пошуку телеграм-ботів, за допомогою

яких можна було б придбати товари від їх компаній в повній мірі, не було знайдено. Саме по цій причині, телеграм-бот, який буде розроблятися упродовж цієї дипломної роботи буде дійсно унікальним.

РОЗДІЛ 2 РОЗРОБКА СТРУКТУРИ ТЕЛЕГРАМ-БОТУ З МЕТОЮ ПОКРАЩЕННЯ ПРОЦЕСУ РЕАЛІЗАЦІЇ ПРОДУКТІВ ПОБУТОВОЇ ХІМІЇ МОВОЮ JAVASCRIPT

2.1 Завдання телеграм-боту з метою покращення процесу реалізації продуктів побутової хімії JS

Telegram має масштабну географію. Найбільш популярним сервіс став в Україні, Росії, Білорусії, Казахстані, Узбекистані та Ірані. Серед інших країн: Німеччина, Італія, Китай, Нідерланди, Киргизія.

Користувачі месенджера - мрія рекламодавця. Платоспроможні, активно користуються всіма можливостями месенджера і цікаво черпають інформацію з просторів Telegram. Використання Telegram для бізнесу відрізняється від електронної пошти або SMS. Telegram - це приватна компанія, яка контролює те, що можливо в додатку. Telegram орієнтований на забезпечення безкоштовного, швидкого і безпечного обміну повідомленнями, і функції створюються з урахуванням цих цілей.

Завдяки такому інструменту Телеграму, як бот, власник бізнесу має змогу розвивати свій бізнес у ньому не тільки через приватні або публічні канали або групи. Основна перевага ботів як одного з інструментів для ведення бізнесу в Телеграмі полягає в автоматизації рутинних задач з боку робітника. Наприклад, Телеграм дозволяє створити бота, який зможе замінити консультанта або менеджера у меті інформування можливого клієнта щодо товару. Однак, як було вищеописано, безліч бізнесів наразі не використовують телеграм-ботів із метою кінцевого продажу товару безпосередньо в самому боті. Для цього є декілька причин, які можна вважати за проблеми:

- майже повна відсутність подібної практики;
- майже повня відсутність інтегрування бота із банківськими системами (зокрема в нашому випадку – з українськими);

- небажання деяких бізнесменів слідкувати за розвитком технологій

При досліджуванні обраної теми дипломної роботи ці проблеми були виявлені й проаналізовані. Саме тому завданням дипломної роботи є розробка унікального на даний момент часу телеграм-боту з цілю покращення реалізації продуктів побутової хімії. Розроблений по власній структурі телеграм-бот буде мати такі можливості:

- аутентифікація для комфортної роботи з ботом на перспективу;
- перегляд можливого асортименту товару;
- онлайн придбання обраного товару серед асортименту;
- перегляд власних створених замовлень усередині бота, які були підтверджені;
- перегляд власних створених замовлень усередині бота, які не були підтверджені;
- зміна своїх контактних даних у будь-який час усередині бота;
- відправляти листи для підтвердження замовлення з детальною інформацією про нього користувачу на його пошту

Обраною мовою програмування для розробки телеграм-боту стала JavaScript, проте не стандартний, а розширений. Розширений JavaScript ще називають також TypeScript. У подальшому ході дипломної роботи, для більшої зручності, мова програмування для написання боту буде іменуватись TypeScript. Також, буде використана програмна платформа Node.js для створення локального серверу, на якому буде запускатись бот.

Для зберігання даних про оформлення замовлень користувачів буде використовуватись об'єктно-реляційна система управління базами даних PostgreSQL, на якій також буде зупинено більш детально далі по ходу виконання цієї дипломної роботи.

2.2 Про TypeScript

TypeScript - це популярний статичний типізатор (static type checker) або типізоване надмножество (typed superset) для JavaScript, інструмент, розроблений Microsoft, який додає систему типів до гнучкості і динамічним можливостям JavaScript.

TypeScript розвивається як проект з відкритим вихідним кодом, поширюється під ліцензією Apache 2.0, має дуже активне і високопрофесійне співтовариство, а також величезний вплив на екосистему JavaScript. Головними перевагами TypeScript перед звичайним JS були визначені:

- Сувора типізація. Багато проблем у JavaScript з'являються через динамічну типізацію та загалом дивну поведінку типів даних;
- Покращене ООП. І в JS, і в TS є підтримка об'єктно-орієнтованого програмування: класи, об'єкти, спадкування. Однак TypeScript зробив крок трохи далі і використовує більше можливостей ОПП. В тому числі, наприклад, інтерфейси.

Проте у TypeScript є, звісно, також і свої мінуси.

Хоча розробники люблять цю мову, а деякі великі проекти вже переходять на нього. Наприклад, популярний фреймворк Angular.JS. Але цього все одно недостатньо, щоб він став таким же затребуваним, як JavaScript. Це пов'язано з тим, що розробка веб-додатків на TypeScript коштує дорожче і забирає більше часу. Особливо це проявляється, якщо необхідно використовувати якусь бібліотеку або фреймворк, які не портовані на TS. У цьому випадку розробникам доведеться самостійно описувати сигнатури (вказувати типи даних) всіх функцій і методів — досить тривалий процес, враховуючи розміри сучасних бібліотек.

Також поріг входу в TypeScript вищий. Щоб використовувати його переваги, важливо знати типи даних і об'єктно-орієнтоване програмування.

Враховуючи усе вищеописане, буде дуже логічним наслідком, зв’явлення у багатьох розробників в голові питання, на кшталт: “Що краще обрати для вивчення JavaScript або TypeScript?“.

Відповідь полягає в тому, що не можна вивчити TypeScript, не вивчивши JavaScript. TypeScript поділяє синтаксис і поведінку під час виконання з JavaScript, тому все, що розробник може дізнатися про JavaScript, допоможе одночасно вивчити TypeScript.

2.3 Моделювання об’єкту проектування

2.3.1 Діаграма прецедентів

Діаграма прецедентів системи - це поведінковий тип діаграми UML, який часто використовується для аналізу різних систем. Вони дозволяють візуалізувати різні типи ролей у системі та те, як ці ролі взаємодіють із системою.

Діаграми прецедентів системи використовуються для збору вимог до використання системи. Залежно від вимог, їх можливо використовувати по-різному. Нижче наведено кілька способів їх використання:

- Визначити функції і те, як ролі взаємодіють з ними;
- Для представлення системи на високому рівні. Є можливість виділити ролі, які взаємодіють з системою, і функціональні можливості, що надаються системою, не заглиблюючись у внутрішню роботу системи;
- Для визначення внутрішніх і зовнішніх факторів. Це може здатися простим – але у великих складних проектах система може бути ідентифікована як зовнішня роль в іншому варіанті використання.

Діаграми прецедентів складаються з 4 об'єктів:

- Актор;
- Приклад використання (Use case);
- Система;
- Пакет;

Ці об'єкти більш детально пояснюються нижче.

Актор - суб'єкт на діаграмі варіантів використання-це будь-який об'єкт, який виконує певну роль в одній даній системі. Це може бути людина, організація або зовнішня система і зазвичай малюється як скелет, показаний нижче.

Рисунок 2.1 – Діаграма прецедентів. Актор.

Use case - варіант використання являє собою функцію або дію в системі. Він намальований у вигляді овалу і названий за допомогою функції.

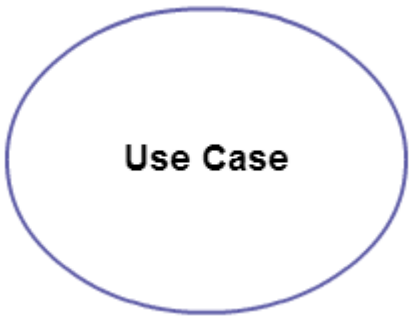


Рисунок 2.2 – Діаграма прецедентів. Use Case.

Система – об’єкт для визначення області застосування варіанту використання і малюється у вигляді прямокутника. Це необов'язковий елемент, але корисний, коли потрібно візуалізувати великі системи. Наприклад, можливо створити всі варіанти використання, а потім використовувати системний об'єкт для визначення області, охопленої проектом. Або навіть можливо використовувати його, щоб показати різні області, порушені в різних випусках.

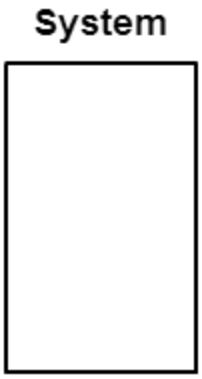


Рисунок 2.3 – Діаграма прецедентів. Система.

Пакет - це ще один необов'язковий елемент, який надзвичайно корисний у складних діаграмах. Подібно до діаграм класів, пакети використовуються для групування варіантів використання. Вони намальовані так, як показано на малюнку нижче.

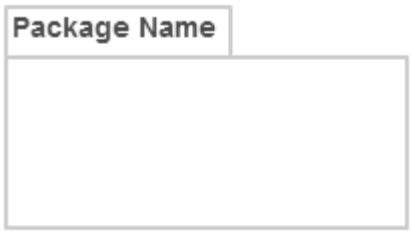


Рисунок 2.4 – Діаграма прецедентів. Пакет.

Також у діаграмі прецедентів існує п'ять типів взаємозв'язків:

- Зв'язок між суб'єктом і варіантом використання;
- Узагальнення актора;
- Розширення взаємозв'язку між двома варіантами використання;
- Включення зв'язку між двома варіантами використання;
- Узагальнення варіанту використання

Приклад діаграми прецедентів:

Рисунок 2.5 – Приклад діаграми прецедентів.

2.3.2 Розробка діаграми прецедентів для телеграм-боту

Оскільки, метою даної дипломної роботи є створення телеграм-боту з ціллю покращення реалізації продуктів побутової хімії, то для кращого розуміння логікі праці бота необхідно розробити власну діаграму прецедентів. Користувачами даного боту будуть потеційні клієнти, які зацікавленні у придбанні товарів побутової хімії ручного виробництва. На рисунку 2.6 показані основні дії, які будуть доступні користувачеві розроблюваного телеграм-бота.

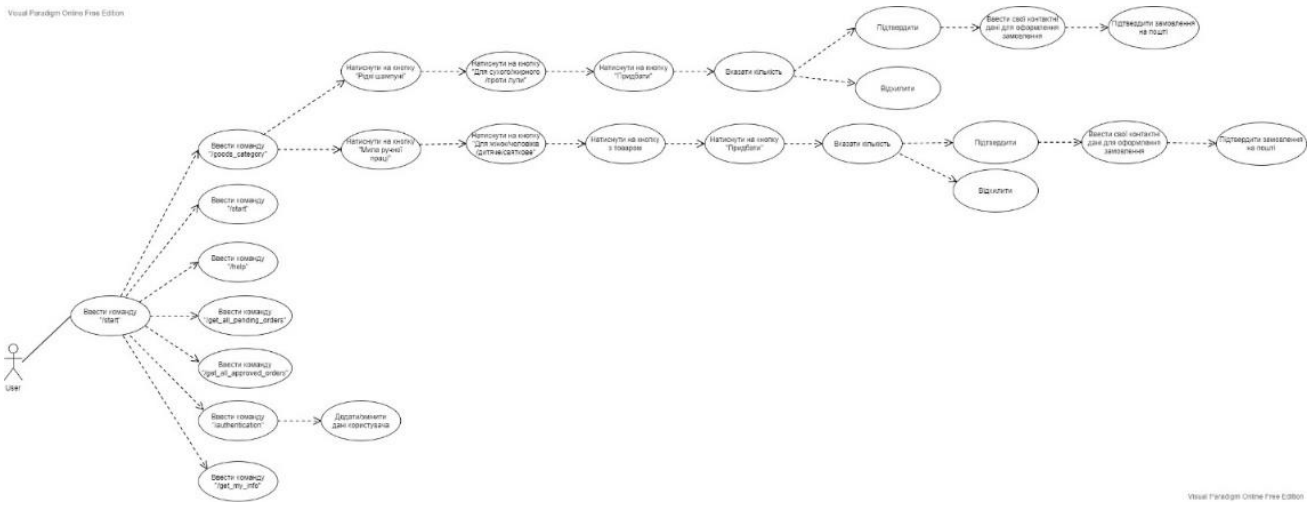


Рисунок 2.6 – UML Діаграма прецедентів телеграм-бота

2.3.3 Діаграма діяльності

Діаграми діяльності можуть використовуватися на всіх етапах розробки програмного забезпечення і для різних цілей. І оскільки вони багато в чому схожі на блок-схеми, вони, як правило, більш популярні, ніж інші типи діаграм UML. Діаграма діяльності UML допомагає візуалізувати певний варіант використання на більш детальному рівні. Це поведінкова діаграма, яка ілюструє потік дій через систему.

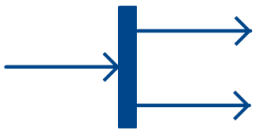
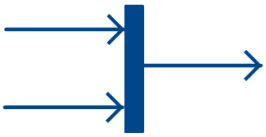
Діаграми діяльності UML також можна використовувати для відображення потоку подій у бізнес-процесі. Вони можуть бути використані для вивчення бізнес-

процесів з метою визначення їх потоку і вимог. UML визначив набір символів і правил для малювання діаграм діяльності. Нижче наведені у вигляді таблиці часто використовувані символи діаграми діяльності з поясненнями.

Таблиця 2.3.1 Символи діаграми діяльності

Symbol	Name	Use
	Start/ Initial Node	Used to represent the starting point or the initial state of an activity
	Activity / Action State	Used to represent the activities of the process
	Action	Used to represent the executable sub-areas of an activity
	Control Flow / Edge	Used to represent the flow of control from one action to the other
	Object Flow / Control Edge	Used to represent the path of objects moving through the activity
	Activity Final Node	Used to mark the end of all control flows within the activity
	Flow Final Node	Used to mark the end of a single control flow

Продовження таблиці 2.3.1 - Символи діаграми діяльності

Symbol	Name	Use
	Decision Node	Used to represent a conditional branch point with one input and multiple outputs
	Merge Node	Used to represent the merging of flows. It has several inputs, but one output.
	Fork	Used to represent a flow that may branch into two or more parallel flows
	Merge	Used to represent two inputs that merge into one output
	Signal Sending	Used to represent the action of sending a signal to an accepting activity
	Signal Receipt	Used to represent that the signal is received
	Note/ Comment	Used to add relevant comments to elements

Приклади діаграм діяльності:

А)

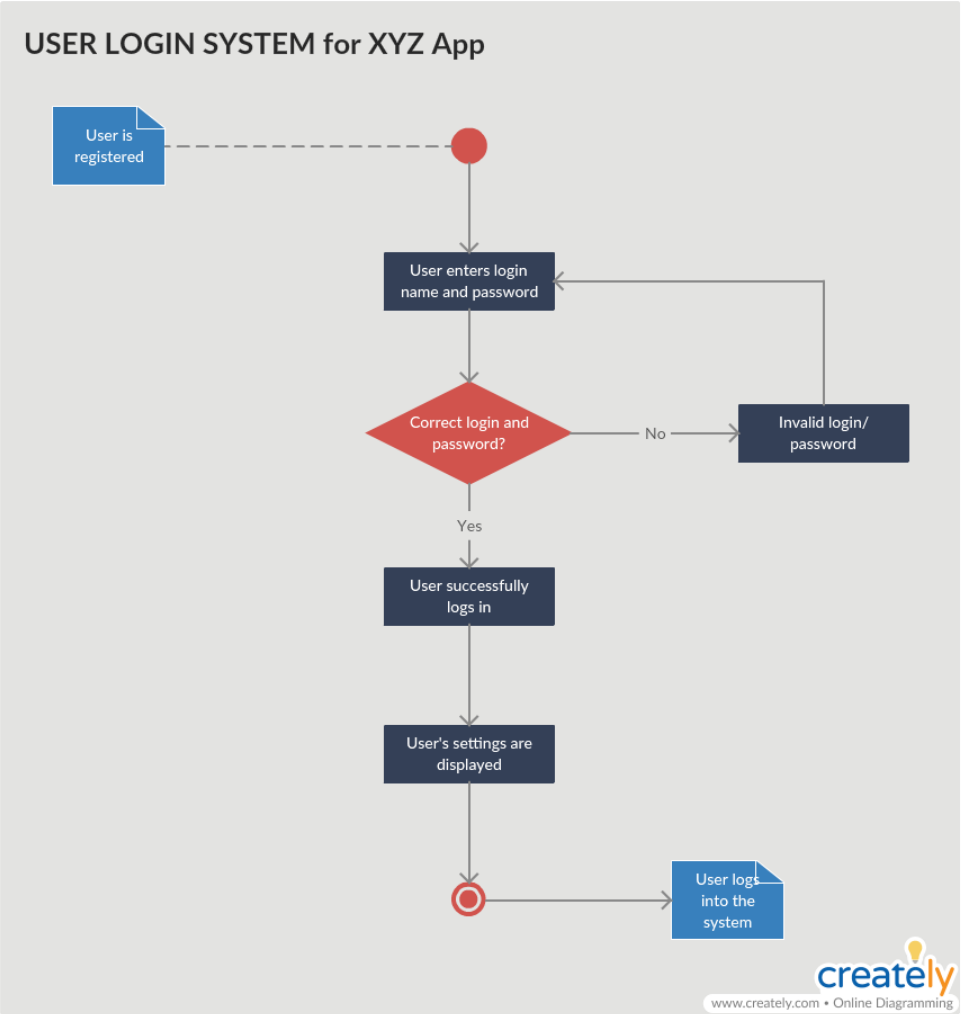


Рисунок 2.7 – Приклад UML Діаграми діяльності №1.

Б)

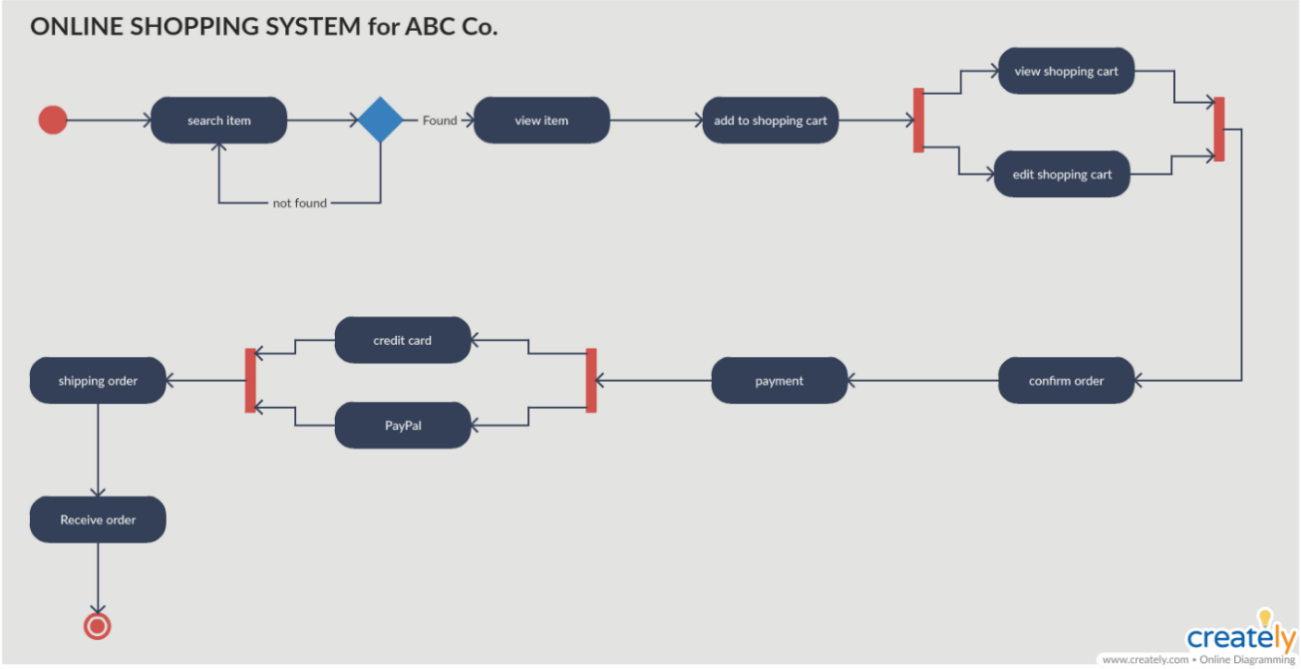


Рисунок 2.8 – Приклад UML Діаграми діяльності №2.

2.3.4 Розробка діаграми діяльності для телеграм-бота

Оскільки, метою даної дипломної роботи є створення телеграм-боту з ціллю покращення реалізації продуктів побутової хімії, то для кращої ілюстрації потоку взаємодій користувача із ботом необхідно розробити власну діаграму діяльності. Користувачами даного боту будуть потенційні клієнти, які зацікавленні у придбанні товарів побутової хімії ручного виробництва. На рисунку 2.9 зображено певний варіант використання, який буде доступний користувачеві розроблюваного телеграм-бота.

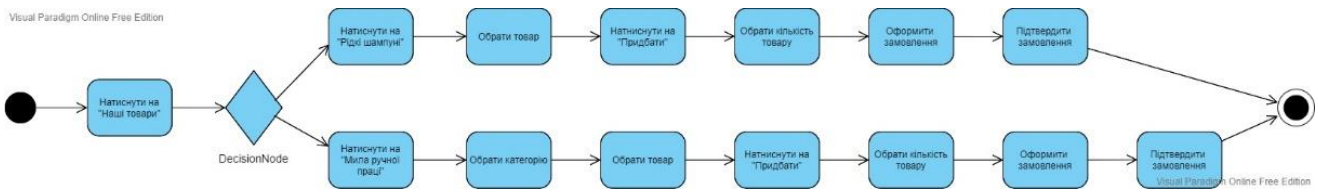


Рисунок 2.9 – UML Діаграма діяльності телеграм-бота

2.4 Алгоритм роботи телеграм-боту

Алгоритм — набір правил (інструкцій) або керувальних впливів, виконання яких завершується бажаним результатом.

Кожне ПЗ має свій певний алгоритм роботи, який має власні інструкції та бажаний кінцевий результат. Телеграм-бот, який розробляється упродовж цієї дипломної роботи не є виключенням. Алгоритм дає змогу зрозуміти, який саме кінцевий результат має бути здобутий, дотримуючись та виконуючі правила, які включені до нього. Основна кінцева мета розроблюваного Телеграм-бота – це продаж обраного товару користувачем. То ж, щоб досягти такого результату необхідно:

1. У користувача має бути встановлений додаток Telegram на його смартфоні чи комп'ютері;
2. У полі пошуку користувач має вписати ім'я боту («Handsoapshop_bot») та перейти до діалогу із ним;
3. В діалозі користувач має написати команду «/start»;
4. У користувача з'явиться кнопка «Меню» усередині якої знаходяться 7 різних команд із певною запрограмованою функціональністю для подальшої взаємодії з ботом: **"/start"**, **"/help"**, **"/authentication"**, **"/goods_category"**, **"/get_all_pending_orders"**, **"/get_all_approved_orders"**, **"/get_my_info".** Користувачу необхідно ввести одну з них для продовження.
5. Оскільки кінцева мета – це придбання товару, то в цьому алгоритмі буде розглянутий випадок із придбанням товару. То ж, у нашому випадку, користувач має обрати команду «/**goods_category**».
6. Далі користувач повинен обрати тип товарів: рідкі шампуні або мила ручної праці.

7. Якщо користувач обрав тип товарів «Рідкі шампуні», то він одразу переходить до вибору продуктів серед цього типу товарів, які представлені в асортименті боту. Якщо ж користувач обрав тип товарів «Мила ручної праці», то перед тим як він потрапить до вибору товарів серед запропонованого асортименту йому необхідно обрати категорію для цього типу товарів: «Для жінок», «Для чоловіків», «Дитяче» або «Святкове».
8. Обравши необхідний продукт, він має натиснути на кнопку «Придбати!».
9. Далі користувач вказує кількість бажаного товару.
10. Завершує оформлення замовлення шляхом вписування своїх контактних даних, якщо попередньо не зробив цього, які будуть зберігатися в БД після вписання.
11. Заходе на свою електронну пошту, яку вказував у якості контакту та підтверджує замовлення у надісланому йому листі. Статус його замовлення після цього також оновиться в БД.
12. Надалі менеджер зв’язується з користувачем по оформленому їм замовлення та обсуджує деталі щодо доставки та способу сплати за товар.

2.5 Про PostgreSQL

Оскільки для запису оформлених замовлень необхідно якесь середовище для збереження даних, було прийнято рішення використати для цих цілей PostgreSQL.

PostgreSQL – це об'єктно-реляційна система управління базами даних, вона є похідною від пакету POSTGRES, написаного в Каліфорнійському університеті в Берклі. Маючи за плечима більше двох десятиліть розробки, PostgreSQL в даний час є найбільш передовою базою даних з відкритим вихідним кодом, доступною де б то не було.

PostgreSQL заслужив міцну репутацію завдяки своїй перевіреній архітектурі, надійності, цілісності даних, надійному набору функцій, розширюваності і прихильності спільноти розробників програмного забезпечення з відкритим вихідним кодом постійного надання високопродуктивних та інноваційних рішень. PostgreSQL працює у всіх основних операційних системах, сумісний з ACID з 2001 року і має потужні доповнення, такі як популярний PostGIS geospatial database extender. Не дивно, що PostgreSQL став кращою реляційною базою даних з відкритим кодом для багатьох людей та організацій.

Почати використовувати PostgreSQL ще ніколи не було так просто-виберіть проект, який ви хочете створити, і дозвольте PostgreSQL безпечно і надійно зберігати ваші дані.

Чому був обраний саме PostgreSQL як необхідна БД? В ній є можливість визначати власні типи даних, створювати власні функції й писати код з різних мов програмування без перекомпіляції бази даних. PostgreSQL намагається відповідати стандарту SQL там, де така відповідність не суперечить традиційним функціям або може призвести до поганих архітектурних рішень. Підтримуються багато функцій, необхідних стандартом SQL, хоча іноді з трохи відмінним синтаксисом або функціями. З часом можна очікувати подальших кроків у напрямку відповідності. Станом на випуск версії 14 у вересні 2021 року PostgreSQL відповідає принаймні 170 з 179 обов'язкових функцій для SQL: відповідність ядру 2016 року.

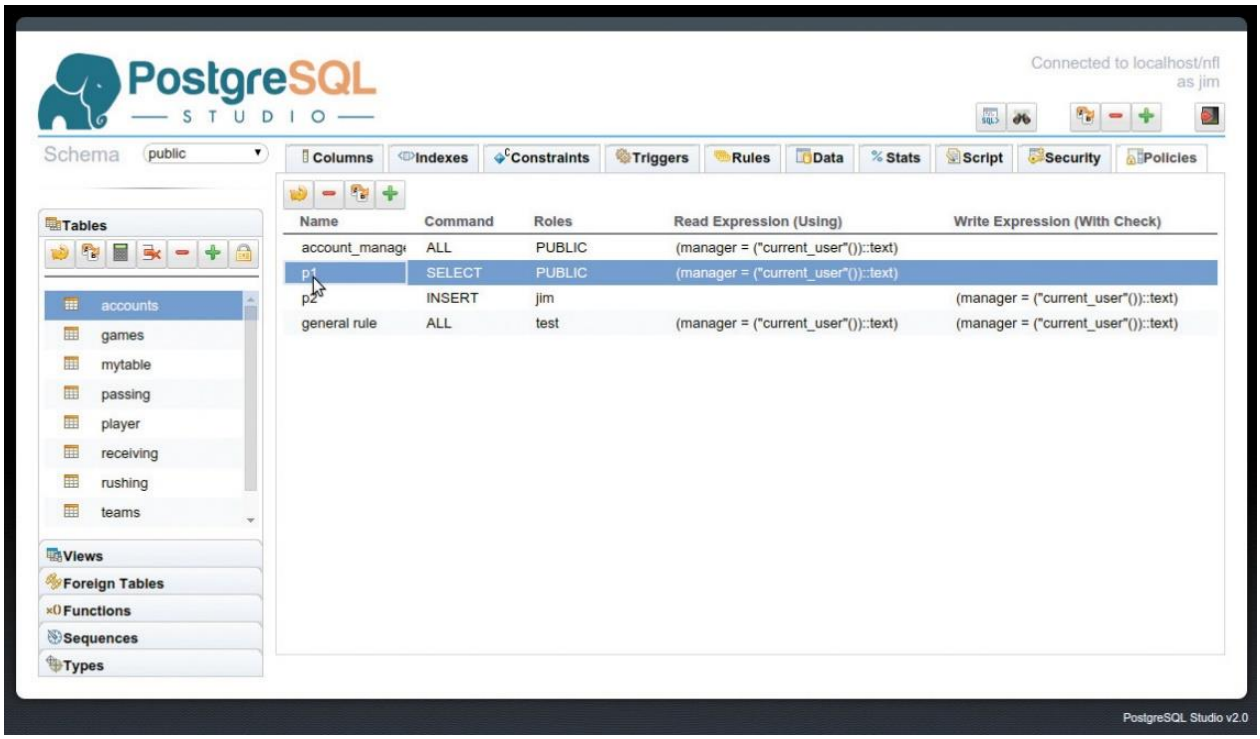


Рисунок 2.10 – Вигляд інтерфейсу PostgreSQL

2.6 Архітектурна структура телеграм-боту

Як згадувалося раніше у ході написання цієї дипломної роботи, архітектурна структура розроблюваного телеграм-боту з ціллю покращення процесу реалізації продуктів побутової хімії була попередньо розроблена у вигляді mind-map. Архітектура є авторською, а тому є унікальною.

Первісний вигляд розробленої архітектури зображено на рисунку 2.11:

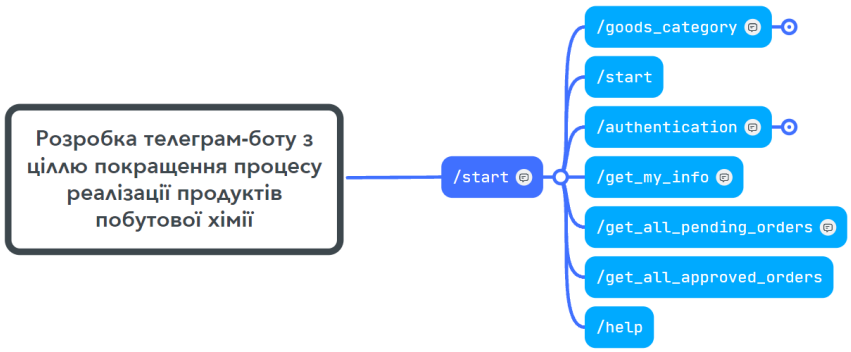


Рисунок 2.11 – Початковий вигляд архітектури розроблюваного телеграм-боту

Як можна побачити все починається зі звичної для праці із ботами в Telegram команди «/start». Вона дає зрозуміти боту, що користувач почав з ним працювати. Після її написання користувачу повідомляється базову інформацію про бота з яким він почав «діалог». Ця інформація має бути прописана завчасно з боку розробника для більшої впевненості в компетенції бота.

Після привітального повідомлення на вибір користувача буде представлено 7 команд, які знаходяться в «Меню» усередині діалогу з ботом. Кожна з команд має подальшу взаємодію. Ось самі прописані з технічного боку команди:

- “/start”;
- “/help”;
- “/authentication”;
- “/goods_category”;
- “/get_all_pending_orders”;
- “/get_all_approved_orders”;
- “/get_my_info”

Натиснувши на будь-яку команду користувач переходить до подальшого сценарію взаємодії із ботом, адже кожна з команд містить у собі цікаву або необхідну для користувача інформацію.

Найбільший та основний сценарій подальшої взаємодії із користувачем міститься у команді «/goods_category», яка наразі і буде розглянута:

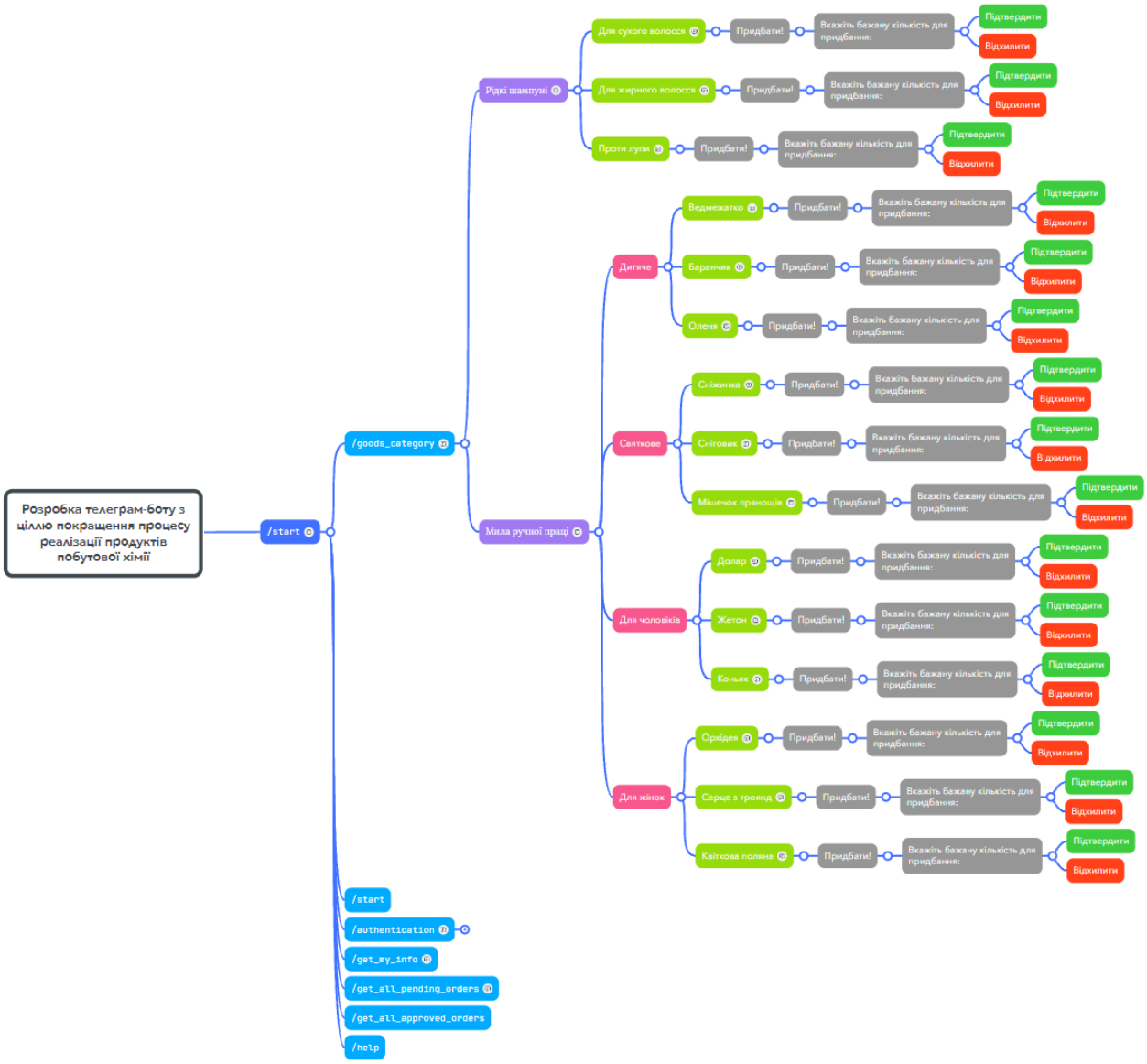


Рисунок 2.12 – Вигляд архітектури телеграм-боту через взаємодію з командою «/goods_category»

Як можна побачити з рисунку 2.12, архітектура взаємодії з користувачем за допомогою команди «/goods_category» містить у собі декілька розгалужень, які в собі мають ще декілька розгалужень. Може здатися, що все це має дуже складний і незрозумілий вигляд, проте насправді все просто й логічно.

Після натиснення команди «/goods_category» іде перше розгалуження, яке має наступний вигляд:

Рисунок 2.13 – Первісний вигляд архітектури телеграм-боту після натиснення команди «/goods_category»

На вибір користувачу буде запропоновано у вигляді ще двох кнопок два типи товарів: «Рідкі шампуні» та «Мила ручної праці». Після обрання більш бажаного типу товару, шляхом натискання по одній з кнопок, для користувача відкриється нове розгалуження. Для прикладу, уявімо, що користувач обрав як бажаний тип товару «Рідкі шампуні»:

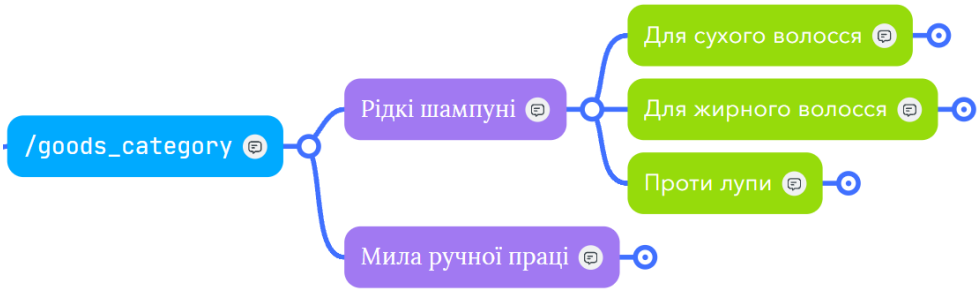


Рисунок 2.13 – Вигляд архітектури телеграм-боту після натиснення команди «/goods_category»

Тут на вибір користувача пропонується 3 шампуні, кожен з яких має свою ціль для використання: «Для сухого волосся», «Для жирного волосся» та «Проти лупи». Припустимо, що користувач обрав як потрібний йому товар шампунь «Проти лупи»:

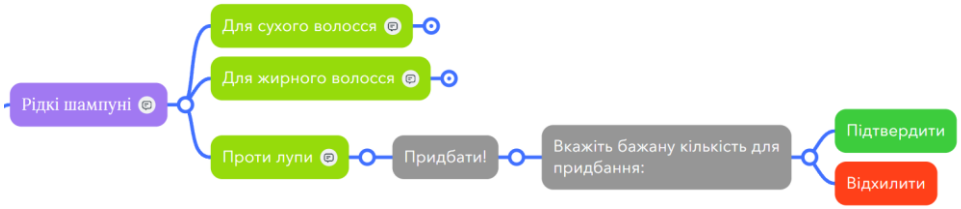


Рисунок 2.14 – Вигляд архітектури телеграм-боту після натиснення кнопки «Проти лупи»

Як можна побачити з рисунку 2.14, користувач майже завершив свій сценарій покупки необхідного для нього товару. При натисканні кнопки «Проти лупи», йому відкрилась не тільки кнопка «Придбати!», але й уся інформація по обраному їм товару: фото, склад та ціна.

Якщо його все задовольняє, то він має натиснути кнопку «Придбати!», далі вказує необхідну йому кількість товару після чого для закінчення оформлення замовлення він має вказати свої контактні дані, якщо попередньо цього не зробив, а після того користувач має підтвердити своє замовлення на вказаній їм пошті у листі, який буде надісланий на його пошту. Після успішного підтвердження замовлення з ним вийде на зв’язок менеджер, щоб оговорити усі деталі оплати та доставки обраного їм товару.

РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТЕЛЕГРАМ-БОТА ТА ТЕСТУВАННЯ

3.1 Обгрунтування вибору підходу до реалізації

Оскільки, раніше у ході виконання дипломної роботи було розглянуто усе необхідне для підтвердження доцільності обраної теми, наразі є необхідність перейти до безпосередньої реалізації та поглибитися у технічну частину.

Реалізація телеграм-боту з метою покращення процесу реалізації продуктів побутової хімії буде виконана у вигляді MVP (Minimum Viable Product). Причини для вирішення питання щодо саме такого підходу є дуже простими:

1. Політичний стан в країні, який, наразі, не передумовлює перспектив для бізнесу та більш глибокої розробки, аніж у вигляді MVP.
2. Обмеження у часі, яке не дозволяє зробити увесь функціонал, який був би можливий в ідеалі.
3. Власні вразливості щодо технічних навичків.

3.2 Набір інструментів для реалізації телеграм-боту

В якості основного (та єдиного в нашому випадку) середовища розробки, була обрана середа Visual Studio Code від компанії Microsoft, яка вже давно зарекомендувала себе як надійне та зручне ПЗ. До того ж, саме ПЗ було якраз спеціально розроблено для написань коду програм, які базуються на веб-технологіях і використовують HTML, CSS, JavaScript, TypeScript та інші.

Основними перевагами Visual Studio Code вважаються:

- Надійність роботи;
- Зручність користувацького інтерфейсу;

- Підтримка великої кількості різних мов програмування;
- Велика кількість плагінів та розширень, що можуть значно спростити роботу;
- Інтеграція з Git (система контроля версій);
- Можливість кастомізації інтерфейсу

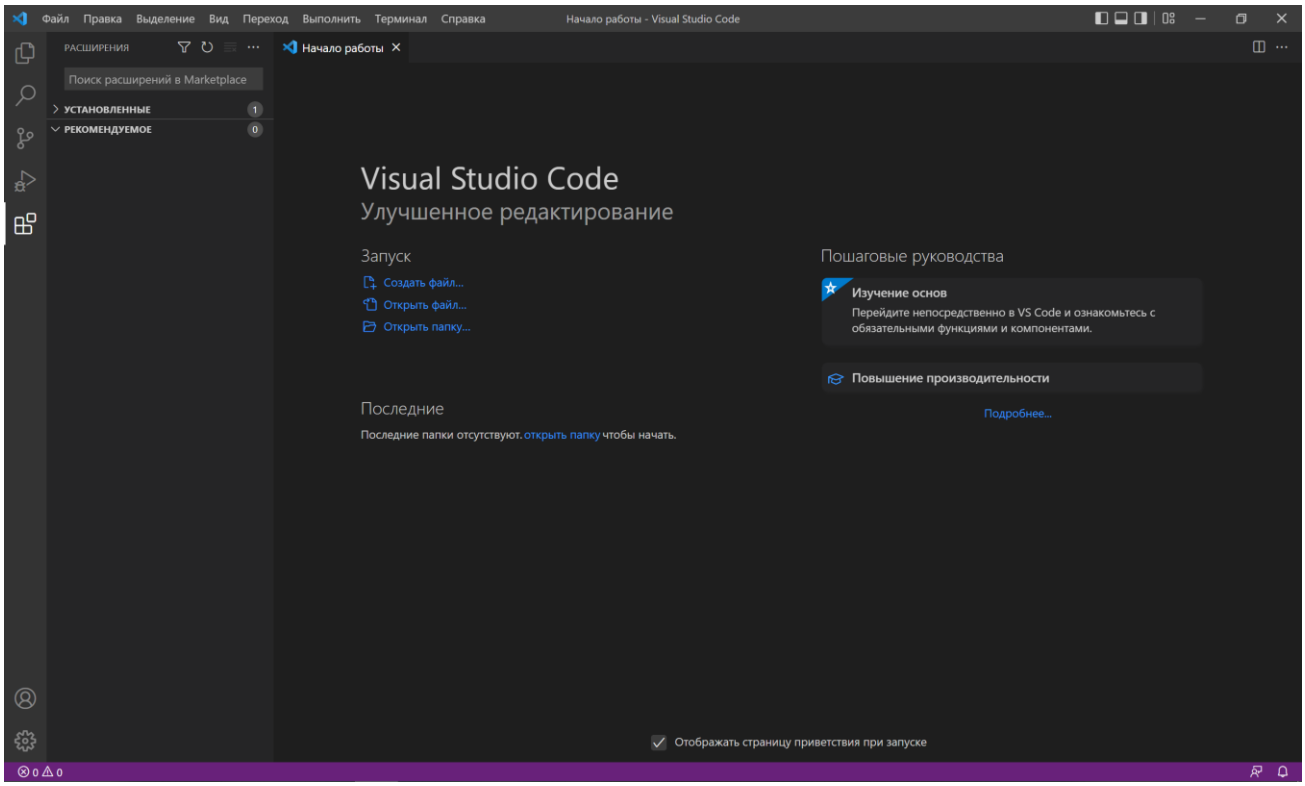


Рисунок 3.1 – Початковий вигляд інтерфейсу Visual Studio Code при першому запуску

Оскільки для збереження даних товарів асортименту бота та архіву замовлень користувачів необхідне якесь середовище для збереження даних, як згадувалося раніше, була обрана СУБД PostgreSQL, більш детально про яку було написано у попередньому розділі.

Основними перевагами СУБД PostgreSQL вважаються:

- Підтримка написання функцій бази даних з використанням SQL, Tcl, Perl, Python, Java, JavaScript, Lua, R, Shell;

- Підтримка великої кількості різних типів даних;
- Можливість визначати свої власні складні типи;
- Повнотекстовий пошук;
- Надійні системи аутентифікації, контролю доступу та управління привілеями;
- Підтримка зовнішніх оболонок даних, що дозволяють представляти таблиці і дані на віддалених серверах і отримувати до них доступ;
- Підтримка уявлень і матеріалізованих уявлень, що забезпечує зручний і спрощений доступ до даних за рахунок абстрагування вихідних структур таблиць для отримання інформації, яка часто запитується разом;
- Наявність коментарів до об'єктів бази даних;
- Ведення журналу з випередженням запису для забезпечення відновлення в певний момент часу, відпрацювання відмови і потокової реплікації;
- Підтримка поведінки, подібної до NoSQL, такого як зберігання документів з використанням JSONB і пар ключ-значення за допомогою hstore

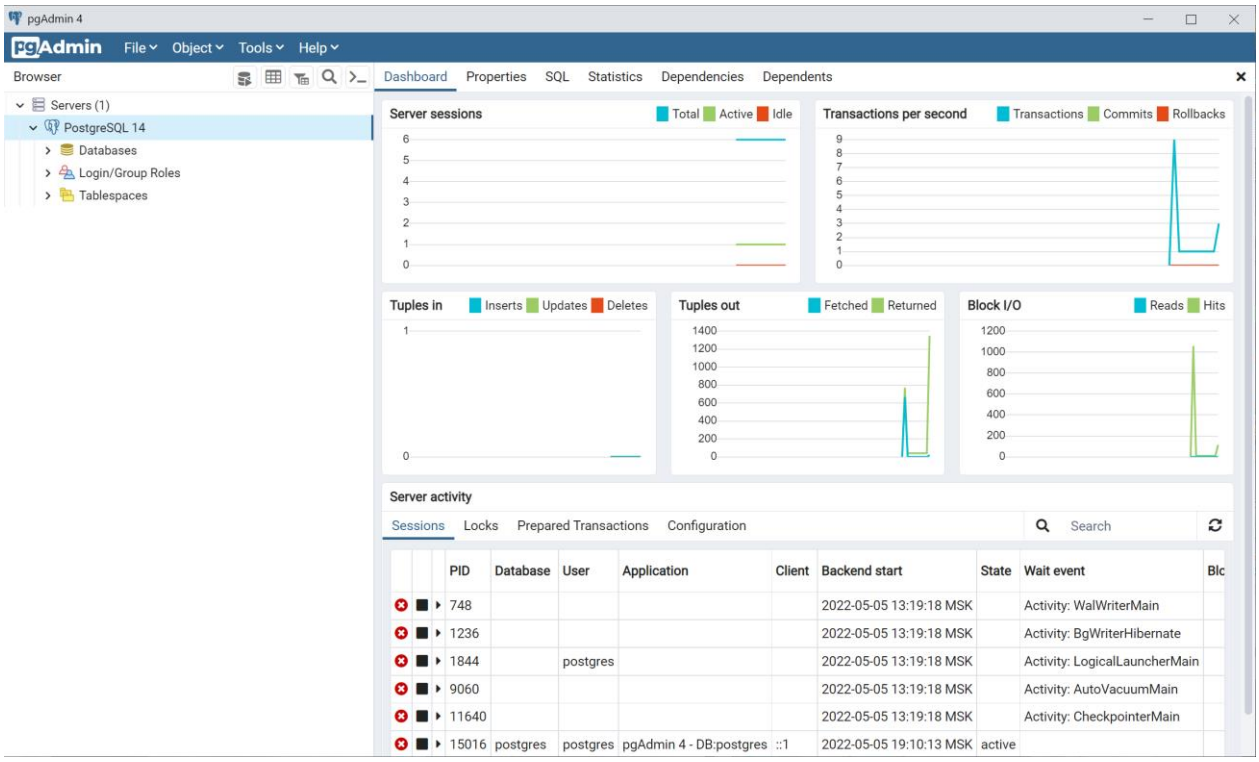


Рисунок 3.2 – Вигляд інтерфейсу PostgreSQL

Також для розширення стандартних можливостей JavaScript й взагалі можливості писати код на TypeScript необхідно ще додатково встановити програмну платформу Node.js.

Node.js - асинхронне середовище виконання JavaScript, кероване подіями, яке призначене для створення масштабованих мережевих додатків.

Основними перевагами Node.js вважаються:

- Легкість засвоєння;
- Потребує менше файлів та коду для написання інтерфейсної і серверної частини в порівнянні з іншими МП;
- Більш швидкий час виходу на ринок;
- Масштабованість;
- Надійність;
- Майже чи не найкращий для розробки MVP;
- Активна спільнота

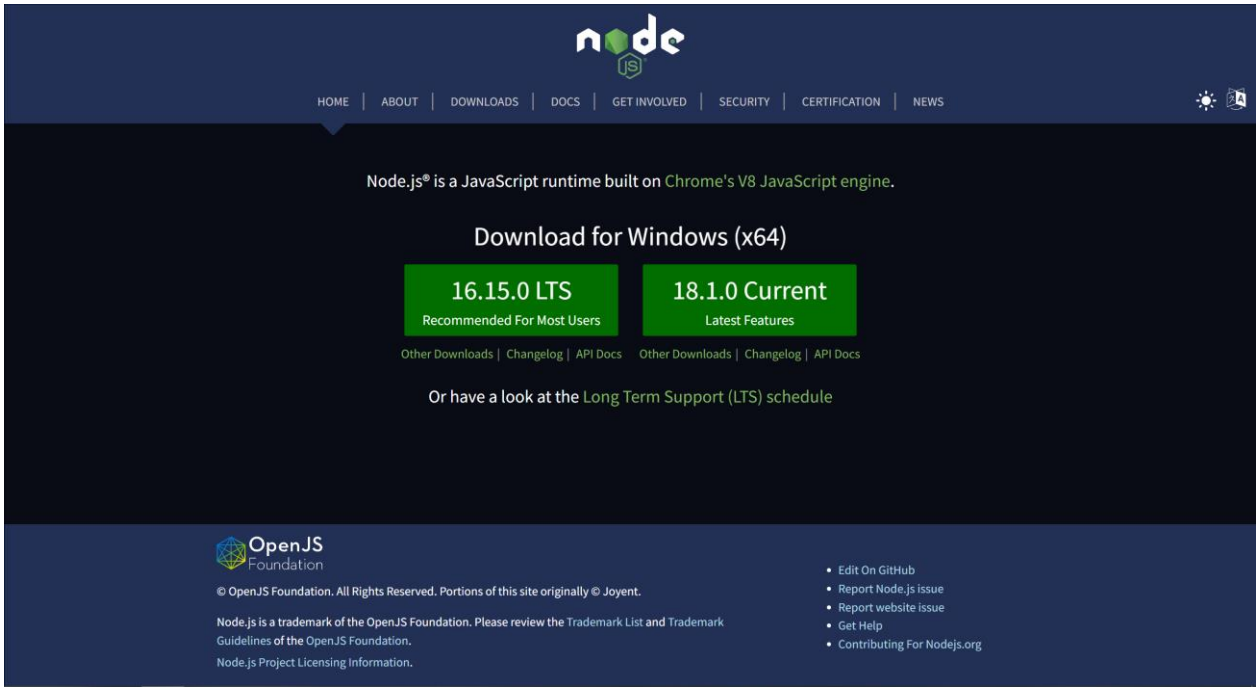


Рисунок 3.3 – Офіційний сайт Node.js

3.3 Функціонал телеграм-боту з метою покращення процесу реалізації продуктів побутової хімії

Оскільки у ході виконання цієї дипломної роботи раніше був розглянутий алгоритм праці телеграм-боту (вже розробленого), то наразі інформація про його функціонал буде розширюватись, проте й частково повторюватися. Загальний вигляд проекту у середовищі розробки Visual Studio Code має наступний вид:

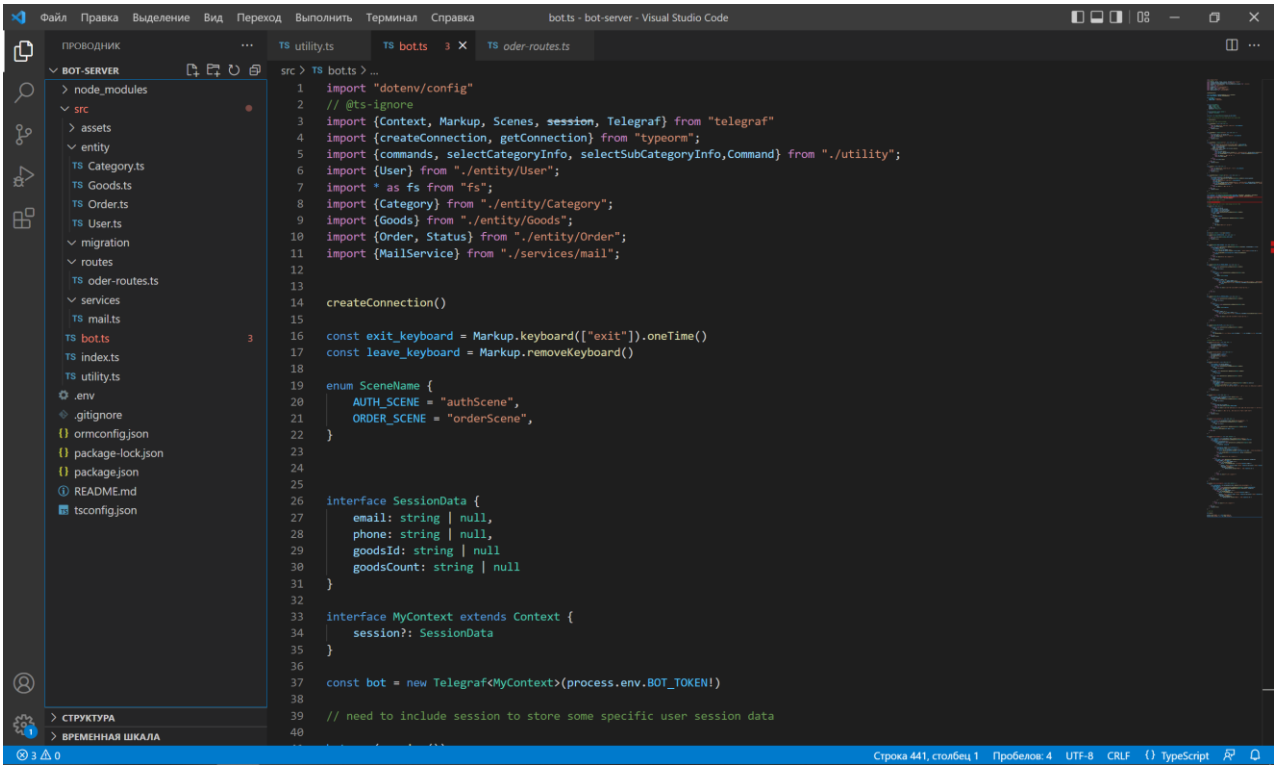


Рисунок 3.4 – Зовнішній вигляд розробленого телеграм-боту у IDE Visual Studio Code

Тут є можливість побачити усю структуру розробленого телеграм-боту, яка складається з різних папок та файлів, які мають різні розширення. Для початку має сенс пройтися поверхнево, щоб краще зрозуміти саму структуру:

Папка «node_modules» містить усі необхідні файли та модулі для коректної праці програмної платформи Node.js.

Папка «entity» містить таблиці, які задіяні для зберігання даних щодо товарів, користувачів та замовлень з їх статусом.

Папка «routers» містить єдиний файл з логікою відправки листа про створені та підтверджені замовлення користувачів для робочого персоналу.

Папка «services» містить єдиний файл зі створенням та зовнішнім оформленням листа, який буде формуватися при появі замовлення користувача й нести детальну інформацію про його створене замовлення.

Зовнішні файли, такі як bot.ts, index.ts, utility.ts будуть розглянуті далі.

Почнемо з головного файлу проекту, який містить увесь функціонал розробленого телеграм-боту з метою покращення процесу реалізації продуктів побутової хімії – файлу bot.ts.

Перше, що при відкритті файлу бросається в очі, це перелік того, що було імпортовано:

```
import "dotenv/config"
// @ts-ignore
import {Context, Markup, Scenes, session, Telegraf} from "telegraf"
import {createConnection, getConnection} from "typeorm";
import {commands, selectCategoryInfo, selectSubCategoryInfo,Command} from "./utility";
import {User} from "./entity/User";
import * as fs from "fs";
import {Category} from "./entity/Category";
import {Goods} from "./entity/Goods";
import {Order, Status} from "./entity/Order";
import {MailService} from "./services/mail";
```

Рисунок 3.5 – Імпортовані файли до bot.ts

Як можна побачити, тут є імпорт з усіх «куточків» проекту: з частини з таблицями для БД; з частини, де реалізован поштовий сервіс для відправки листа; з телеграфу, який формує зовнішній UI бота та інше.

Далі вже безпосередньо йдуть інтерфейси та асинхронні функції, які несуть у собі функціонал бота. Нижче будуть розглянуті основні функціональності даного телеграм-боту, які є найважливіші з точки зору користувача.

```
interface SessionData {
    email: string | null,
    phone: string | null,
    goodsId: string | null
    goodsCount: string | null
}

interface MyContext extends Context {
    session?: SessionData
}
```

За допомогою наведених інтерфейсів іде обозначення полів та типів даних користувача, які надалі будуть використані для подальшої праці цього конкретного користувача з телеграм-ботом.

```
const authAskEmail = async (ctx: any) => {
    try {
        await ctx.reply("Введіть Вашу електронну пошту:", exit_keyboard)
        return ctx.wizard.next()
    } catch (e) {
        console.error(e)
    }
}
```

Асинхронна функція «authAskEmail», яка запитує у користувача його електрону пошту при авторизації для запису його сесіоних даних й подальшої праці з ботом для оформлення замовлення.

```
const handleEmail = Telegraf.on("text", async (ctx: any) => {
    try {
        ctx.session.email = ctx.message.text
        await ctx.reply("Введіть Ваш номер телефону:", exit_keyboard)
        return ctx.wizard.next()
    } catch (e) {
        console.error(e)
    }
})
```

Асинхронна функція «handleEmail», яка спрацьовує після «authAskEmail». «handleEmail» запитує у користувача його номер телефону, після того, які він вказав свою електрону пошту для запису його сесіоних даних для подальшої праці з ботом щодо оформлення замовлення.

```
const handlePhone = Telegraf.on("text", async (ctx: any) => {
    try {
        await ctx.reply("Дякую за внесення повної інформації!", leave_keyboard)
        const res = await getConnection().createQueryBuilder().update(User).set({
```

```
            email: ctx.session.email,
            phone: ctx.message.text
        }).where("tgId= :tgId", {tgId: ctx.from.id}).execute()
        if (ctx.session.goodsId) {
            if (res) {
                await ctx.scene.leave()
                const buttons = [Markup.button.callback("Hi", "continue_decline"),
Markup.button.callback("Так", `order_${ctx.session.goodsId}`)]
                await ctx.replyWithHTML("Повернутись до замовлення?",
Markup.inlineKeyboard([buttons]))
            } else {
                await ctx.reply("Щось пішло не так...")
                return ctx.scene.leave()
            }
        } else {
            return ctx.scene.leave()
        }
    } catch (e) {
        console.error(e)
    }
})
```

Асинхронна функція «handlePhone», яка спрацьовує після введення користувачем усієї необхідної контактної інформації. У разі, якщо, користувач намагався оформити замовлення без попередньої авторизації, вона спитає у користувача, чи бажає він повернутися до оформлення замовлення чи ні.

```
const askCount = async (ctx: any) => {
    try {
        await ctx.reply("Введіть бажану кількість товару:", exit_keyboard)
        return ctx.wizard.next()
    } catch (e) {
        console.error(e)
    }
}
```

Асинхронна функція «askCount», яка запитує у користувача кількість обраного їм товару для подальшого формування замовлення.

```
const handleOrderCount = Telegraf.on("text", async (ctx: any) => {
    try {
        ctx.session.orderCount = +ctx.message.text
        const goods = await
getConnection().getRepository(Goods).findOne(ctx.session.goodsId)
        if (goods && goods.id) {
            await ctx.reply("Кількість збережено!", leave_keyboard)
            ctx.scene.leave()
            const buttons = [Markup.button.callback("Відхилити", "decline_order"),
Markup.button.callback("Підтвердити", "accept_order")]
            await ctx.replyWithHTML(`Ваше замовлення - <b>категорія товару:
${goods.category.name}</b>, <b>назва: ${goods.name}</b>, <b>кількість:
${ctx.message.text}</b>, <b>сумма: ${+ctx.message.text * goods.price} грн.</b>`,
                Markup.inlineKeyboard([buttons]))
        } else {
            await ctx.reply("Щось пішло не так...")
        }
    } catch (e) {
        console.error(e)
    }
})
```

Асинхронна функція «handleOrderCount», яка підтверджує спробу оформити замовлення й пропонує користувачу зробити його кінцевий вибір, щодо створення його замовлення й надає інформацію про попередньо створене замовлення.

```
bot.start(async (ctx: any) => {
    try {
        const tgId = ctx.from.id
        const userName = ctx.from.username
        const firstName = ctx.from.first_name
        const lastName = ctx.from.last_name
        const user = await getConnection().getRepository(User).findOne({
            where: {tgId}
        })
        if (user) {
            ctx.reply(commands)
```

```
        } else {
            const res = await getConnection().getRepository(User).insert({
                tgId,
                userName,
                firstName,
                lastName
            })
            if (res) {
                ctx.reply("Успішно авторизовано!")
            }
        }
    } catch (e) {

    }
})
```

Асинхронна функція, яка містить у собі реалізацію команди «/start» у телеграм-боті. При початку взаємодії між користувачем та ботом цю команду необхідно прописати. Після її прописання бот візьме початкову інформацію про користувача, яка йому буде доступна (ім’я, прізвище, ім’я користувача (якщо є) та створить унікальне айді для цього користувача в БД).

```
bot.command(Command.GOODS_CATEGORY, async (ctx: Context) => {
    try {
        const res = await getConnection().getRepository(Category).find({where:
{isSubCategory: false}})
        if (res.length > 0 && res) {
            const buttons: any[] = []
            for (let i = 0; i < res.length; i++) {
                buttons.push(Markup.button.callback(`${res[i].name}`,
`select_category_${res[i].id}`))
            }
            await ctx.replyWithHTML(selectCategoryInfo, Markup.inlineKeyboard([
                buttons
            ]))
        } else {
            await ctx.reply("Категорій не знайдено!")
```

```
        }
    } catch (e) {
        console.error(e)
    }

})
```

Асинхронна функція, яка містить у собі реалізацію команди «/goods_category» у телеграм-боті. За допомогою цієї команди користувач має можливість продивитися можливі на актуальний момент часу категорії товарів, які представлені у вигляді кнопок.

```
bot.command(Command.GET_ALL_PENDING_ORDERS, async (ctx: any) => {
    try {
        const user = await getConnection().getRepository(User).findOne({
            where: {
                tgId: ctx.from.id
            }
        })
        if (user) {
            const orders = await getConnection().getRepository(Order).find({
                where: {
                    user,
                    status: Status.Pending
                }
            })
            if (orders) {
                if (orders.length > 0) {
                    for (let i = 0; i < orders.length; i++) {
                        const button = [Markup.button.callback("Відправити нове
підтвердження!", `send_new_approval_${orders[i].id}`)]
                        await ctx.reply(`Дата створення:
${orders[i].createdDate.toLocaleDateString()}\n\nКатегорія товару:
${orders[i].goods.name}\nНазва товару: ${orders[i].goods.category.name}\nКількість:
${orders[i].count}\nЦіна: ${orders[i].goods.price} грн.\nСумма: ${orders[i].count *
orders[i].goods.price} грн.`, Markup.inlineKeyboard([
                            button
                        ]))
```

```
                    }
                }else{
                    await ctx.reply(`Наразі у Вас немає непідтвердженних замовлень.`)
                }
            }
        }
    } catch (e) {

    }
})
```

Асинхронна функція, яка містить у собі реалізацію команди «/get_all_pending_orders» у телеграм-боті. За допомогою цієї команди користувач може подивитися свої замовлення, які не були їм підтверджені через пошту й при бажані, може їх підтвердити за допомогою відправки листа з підтвердженням обраного замовлення знову.

```
bot.command(Command.GET_ALL_APPROVED_ORDERS, async (ctx: any) => {
    try {
        const user = await getConnection().getRepository(User).findOne({
            where: {
                tgId: ctx.from.id
            }
        })
        if (user) {
            const orders = await getConnection().getRepository(Order).find({
                where: {
                    user,
                    status: Status.Approved
                }
            })
            if (orders) {
                if(orders.length > 0) {
                    for (let i = 0; i < orders.length; i++) {
                        await ctx.reply(`Дата створення:
${orders[i].createdDate.toLocaleDateString()}\n\nКатегорія товару:
${orders[i].goods.name}\nНазва товару: ${orders[i].goods.category.name}\nКількість:
```

```
${orders[i].count}\nЦіна: ${orders[i].goods.price} грн.\nСумма: ${orders[i].count *
orders[i].goods.price} грн.`)
                    }
                }else{
                    await ctx.reply(`Наразі у Вас немає підтвердженних замовлень.`)
                }
            }
        }
    } catch (e) {

    }
})
```

Асинхронна функція, яка містить у собі реалізацію команди «/get_all_approved_orders» у телеграм-боті. За допомогою цієї команди користувач може подивитися свої підтверджені замовлення та отримати інформацію щодо них.

```
bot.command(Command.GET_MY_INFO, async (ctx: any) => {
    try {
        const user = await getConnection().getRepository(User).findOne({
            where: {
                tgId: ctx.from.id
            }
        })
        if (user) {
            ctx.replyWithHTML(`Ім'я користувача: ${user.firstName + " " +
user.lastName} \nПошта: ${user.email ? user.email : "Наразі пошта не прив'язана!"}
\nТелефон: ${user.phone ? user.phone : "Наразі телефон не прив'язано!"}`)
        }
    } catch (e) {
        console.error(e)
    }
})
```

Асинхронна функція, яка містить у собі реалізацію команди «/get_my_info» у телеграм-боті. За допомогою цієї команди користувач може додати або змінити свої контактні дані, які він надавав боту раніше.

```
bot.action("accept_order", async (ctx: any) => {
    try {
        const goodsId = ctx.session.goodsId
        const orderCount = ctx.session.orderCount
        const user = await getConnection().getRepository(User).findOne({
            where: {
                tgId: ctx.from.id
            }
        })
        const goods = await getConnection().getRepository(Goods).findOne({
            where: {
                id: goodsId
            }
        })
        const res = await getConnection().getRepository(Order).insert({
            user,
            goods,
            count: orderCount
        })
        if (res && user) {
            console.log(user.email)
            const newOrder = await
getConnection().getRepository(Order).findOne(res.raw[0].id)
            if (newOrder) {
                MailService.confirmOrder(user.email, newOrder)
                ctx.reply("Дякуємо за оформлення замовлення! Лист з підтвердженням
вже відправленно на Вашу пошту! Відкрийте його та підтвердіть Ваше замовленя, після
чого наш менеджер зв'яжеться з Вами найближчим часом!")
            }
        }
    } catch (e) {
        console.error(e)
    }
})
```

Асинхронна функція, у якій реалізована дія бота після того як користувач підтвердив свій намір щодо оформлення замовлення. Бот повідомляє користувача

щодо його успішного оформлення замовлення й про те, що на електрону пошту, яку користувач вказав раніше, було надіслано лист із підтвердженням замовлення.

```
bot.action(/order_+/, async (ctx: any) => {
    try {
        const goodsId = ctx.callbackQuery!.data!.replace("order_", "")
        ctx.session.goodsId = goodsId
        const user = await getConnection().getRepository(User).findOne({
            where: {
                tgId: ctx.from.id
            }
        })
        if (user) {
            if (user.email && user.phone) {
                await ctx.scene.enter(SceneName.ORDER_SCENE)
            } else {
                await ctx.reply(`Наразі у Вас заповненна не вся інформація. Будь-
ласка додайте свою пошту та номер телефону! \nЯкщо ви хочете авторизуватись:
/${Command.AUTHENTICATION}`)
            }
        } else {
            await ctx.reply('Щось пішло не так... Будь-ласка використайте команду
/start')
        }
    } catch (e) {
        console.error(e)
    }
})
```

Асинхронна функція, у якій реалізована дія бота після того як користувач спробував оформити замовлення, завчасно не вказавши свої контактні дані. Бот пропонує ввести цьому користувачу його пошту та номер телефону, щоб надалі користувач мав змогу підтвердити свій намір щодо оформлення замовлення.

Наступним у розгляді є файл index.ts. Якщо казати просто і коротко, то цей файл слугує для створення серверу на якому телеграм-бот з метою покращення

процесу реалізації продуктів побутової хімії буде працювати.

Рисунок 3.6 – Код файлу index.ts у середовищі розробки Visual Studio Code

Файл utility.ts містить у собі всі доступні на даний момент часу команди для розробленого телеграм-боту з метою покращення процесу реалізації продуктів побутової хімії, більша частина з яких вже була розглянута та описана вище та також пару змінних.

src > TS utility.ts > ...

```ts
export enum Command {
    AUTHENTICATION = "authentication",
    GOODS_CATEGORY = "goods_category",
    GET_ALL_PENDING_ORDERS = "get_all_pending_orders",
    GET_ALL_APPROVED_ORDERS = "get_all_approved_orders",
    GET_MY_INFO = "get_my_info"
}

const commands = `
/start - команда для запуску бота
/help - команда щоб побачити список наявних команд
/${Command.AUTHENTICATION} - команда для аутентифікації (реєстрація/оновлення користувацьких даних)
/${Command.GOODS_CATEGORY} - команда для отримання списка наявних категорій товарів
/${Command.GET_ALL_PENDING_ORDERS} - команда для отримання списка не підтверджених замовлень
/${Command.GET_ALL_APPROVED_ORDERS} - команда для отримання списка підтверджених замовлень
/${Command.GET_MY_INFO} - команда для отримання користувацьких даних
`

const selectCategoryInfo = "Оберіть категорію товарів які вас цікавлять:"
const selectSubCategoryInfo = "Оберіть під-категорію товарів які вас цікавлять:"

export {commands, selectCategoryInfo, selectSubCategoryInfo}
```

Рисунок 3.7 – Код файлу utility.ts у середовищі розробки Visual Studio Code

3.4 Зовнішній вигляд розробленого телеграм-боту

У цьому підрозділі буде розглянутий зовнішній вигляд телеграм-боту з метою покращення процесу реалізації продуктів побутової хімії. Почнемо з зовнішнього вигляду боту у вікні діалогів месенджеру Telegram. Він має наступний вигляд:

Рисунок 3.8 – Вигляд телеграм-боту у виді окремого діалогу с користувачем

Як можна побачити, розроблений телеграм бот має аватарку, назву й позначку робота, що підтверджує, що це дійсно бот.

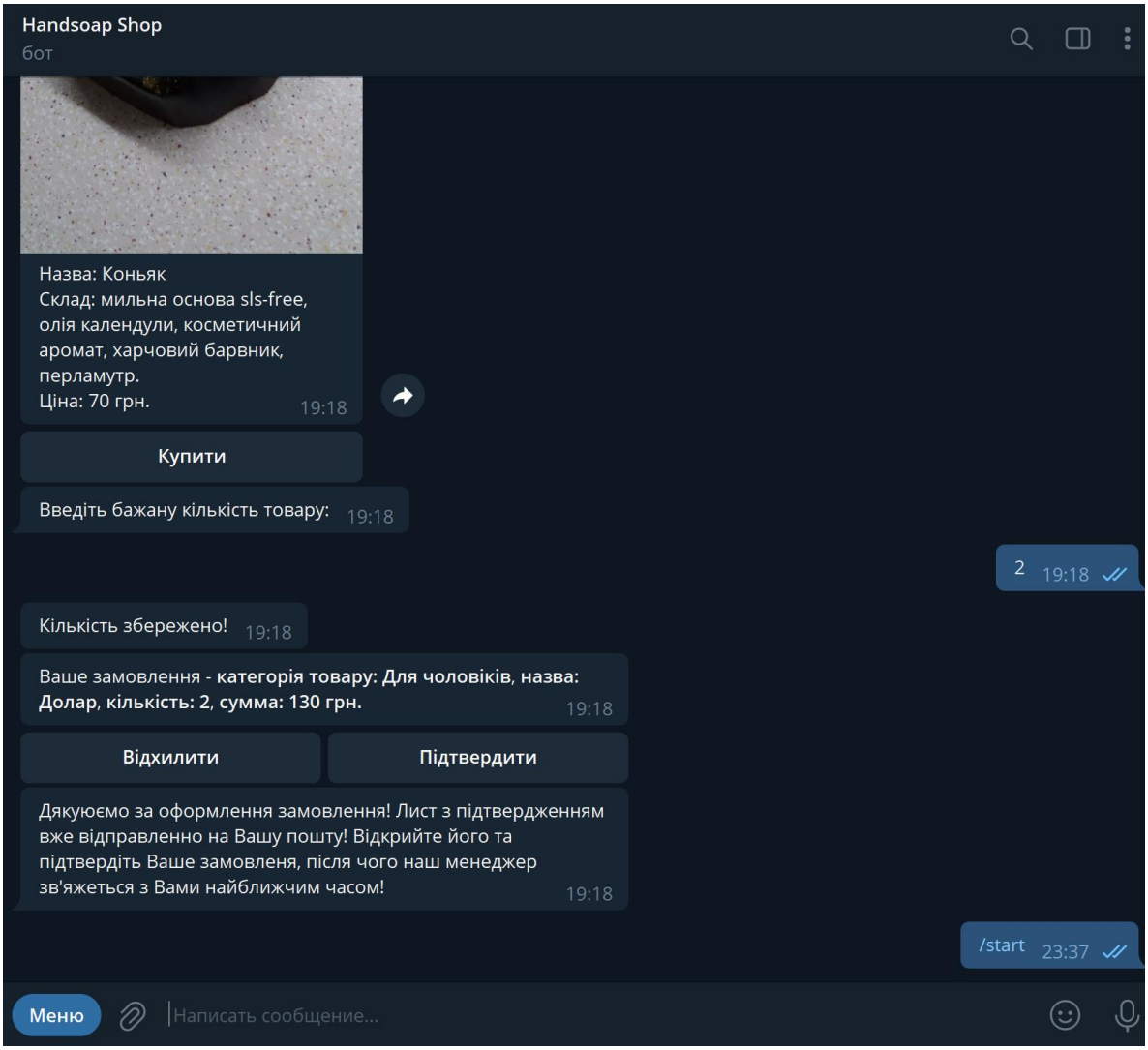


Рисунок 3.9 – Вигляд телеграм-боту в діалозі с користувачем

З рисунку вище можна побачити, що під назвою боту, де у звичайних користувачів вказана їх остання діяльність, або, простіше кажучі, їх час, коли останній раз вони були в мережі, є напис «бот», що також підтверджує, що наразі відкритий діалог з ботом.

Якщо натиснути на ім’я бота, то відкриється додаткова інформаційна панель, яка дозволяє побачити «ім’я користувача» у бота, усі вкладення та додаткові функції з ним:

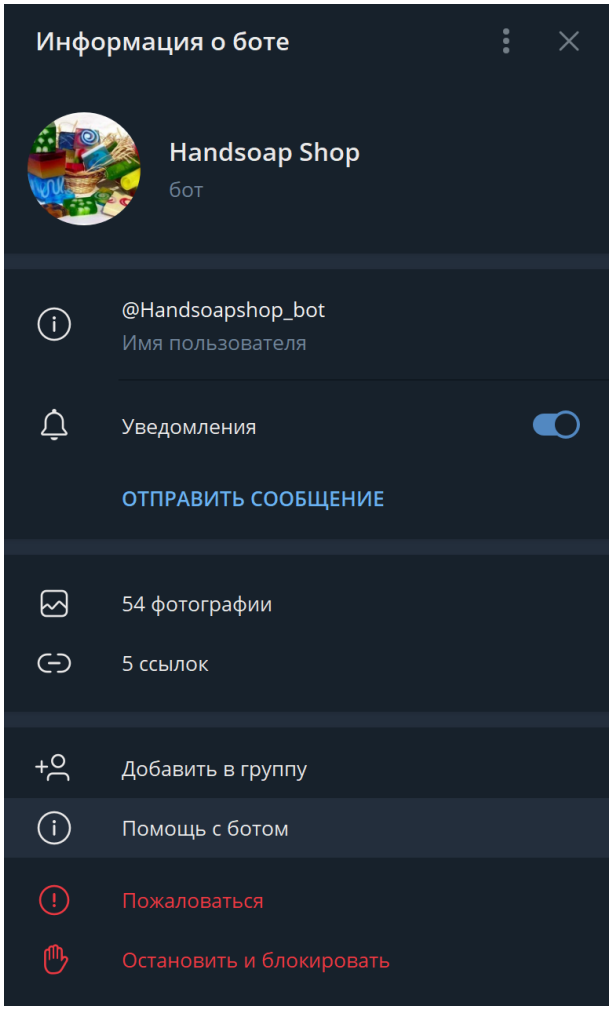


Рисунок 3.10 – Вигляд вікна з додатковою інформацією про телеграм-бот в Telegram

Наостанок, потрібно ще розглянути зовнішній вигляд меню взаємодії користувача з ботом усередині вікна діалогу:

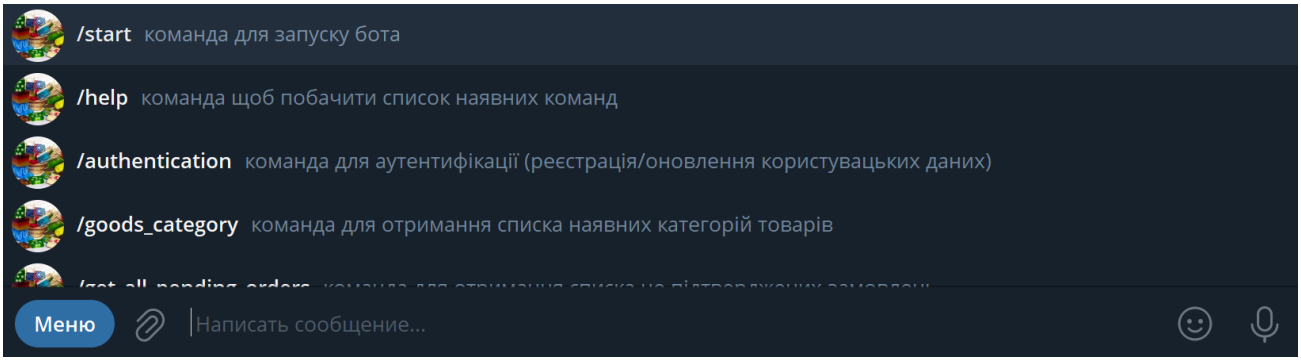


Рисунок 3.11 – Вигляд робочої області «Меню» в діалозі з ботом

Як можна побачити по рисунку вище, у «Меню» знаходяться команди за допомогою яких можна взаємодіяти з ботом. У розділі 3.3 були розглянуті з точки зору реалізації коду більшість з них, проте хотілось би також наголосити про єдину команду про котру раніше не було сказано. Це команда /help. Вона слугує лише для однієї мети – виведення інформації про можливі команди, які розуміє бот у вигляді текстового повідомлення:

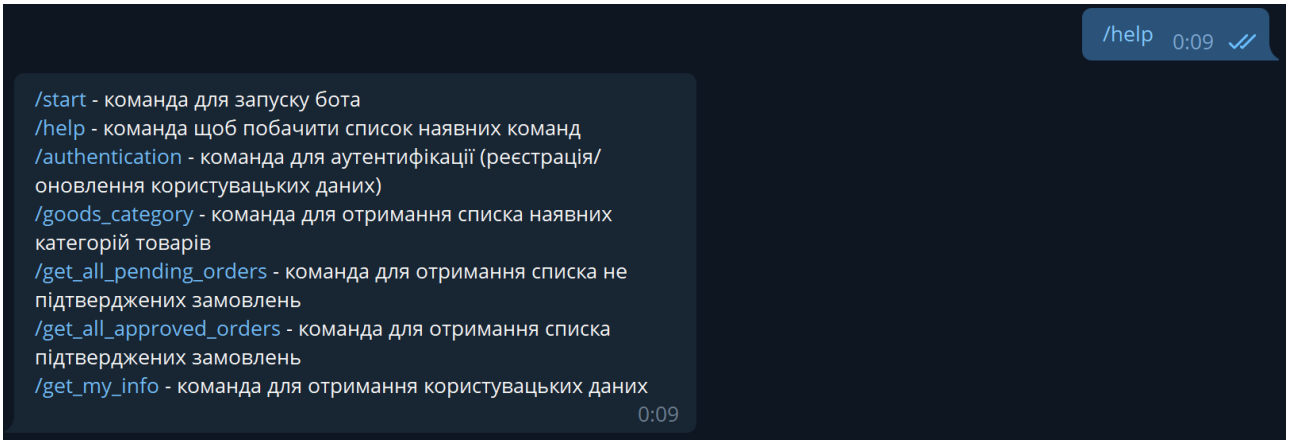
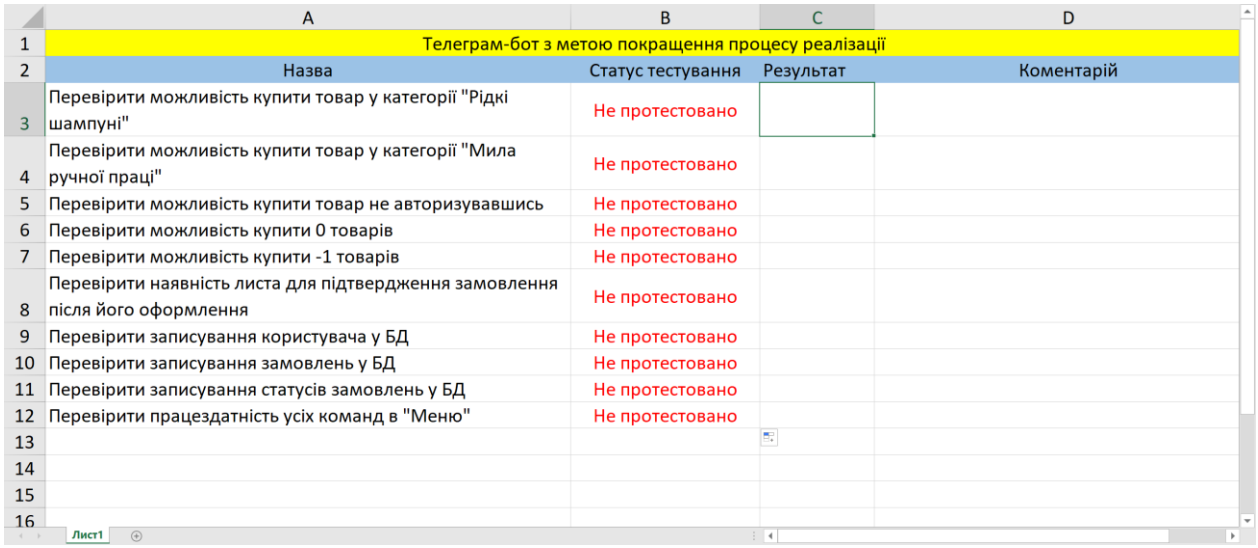


Рисунок 3.12 – Вигляд повідомлення після написання команди «/help» в діалозі з ботом

3.5 Тестування телеграм-боту з метою покращення процесу реалізації продуктів побутової хімії

Оскільки, дійсно якісним ПЗ вважається лише те ПЗ, яке було не тільки грамотно розроблено з технічної сторони, а ще й добре протестовано, наразі є сенс перейти як раз до тестування розробленого телеграм-боту.

Слідуючи другому принципу тестування, яка наголошує, що «Вичерпне тестування неможливе», у даному підрозділі не буде спроби повністю протестувати абсолютно весь телеграм-бот, адже це просто неможливо. Набагато доцільніше буде провести «смоук-тестування» за допомогою попередньо розробленого чек-ліста, який передбачає тестування найосновнішого функціоналу боту:



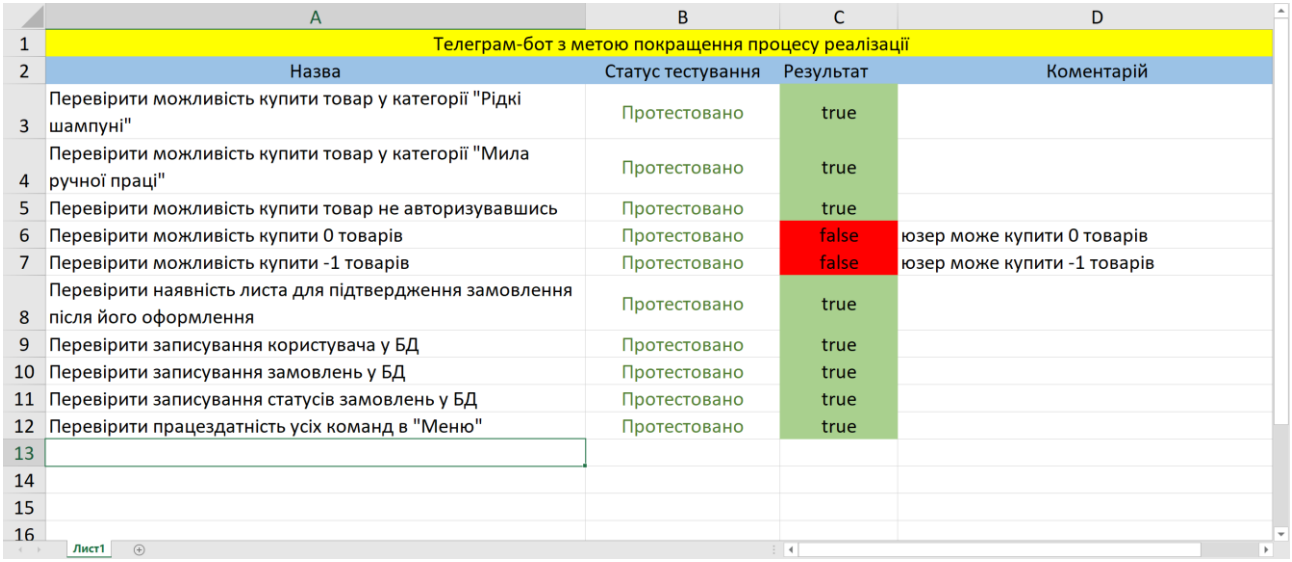
	A	B	C	D
1	Телеграм-бот з метою покращення процесу реалізації			
2	Назва	Статус тестування	Результат	Коментарій
3	Перевірити можливість купити товар у категорії "Рідкі шампуні"	Не протестовано		
4	Перевірити можливість купити товар у категорії "Мила ручної праці"	Не протестовано		
5	Перевірити можливість купити товар не авторизувавшись	Не протестовано		
6	Перевірити можливість купити 0 товарів	Не протестовано		
7	Перевірити можливість купити -1 товарів	Не протестовано		
8	Перевірити наявність листа для підтвердження замовлення після його оформлення	Не протестовано		
9	Перевірити записування користувача у БД	Не протестовано		
10	Перевірити записування замовлень у БД	Не протестовано		
11	Перевірити записування статусів замовлень у БД	Не протестовано		
12	Перевірити працездатність усіх команд в "Меню"	Не протестовано		
13				
14				
15				
16				

Лист1

Рисунок 3.13 – Вигляд імпровізованого чек-ліста для тестування телеграм-боту

Як можна побачити вище на рисунку 3.13 є 10 коротких перевірок, які націлені на функціональну перевірку можливостей бота. Оскільки даний телеграм-бот розроблений у рамках MVP цих перевірок для смоук тестування буде достатньо.

Після проведення смоук-тестування телеграм-боту з метою покращення процесу реалізації продуктів побутової хімії, отримані такі результати:



	A	B	C	D
1	Телеграм-бот з метою покращення процесу реалізації			
2	Назва	Статус тестування	Результат	Коментарій
3	Перевірити можливість купити товар у категорії "Рідкі шампуні"	Протестовано	true	
4	Перевірити можливість купити товар у категорії "Мила ручної праці"	Протестовано	true	
5	Перевірити можливість купити товар не авторизувавшись	Протестовано	true	
6	Перевірити можливість купити 0 товарів	Протестовано	false	юзер може купити 0 товарів
7	Перевірити можливість купити -1 товарів	Протестовано	false	юзер може купити -1 товарів
8	Перевірити наявність листа для підтвердження замовлення після його оформлення	Протестовано	true	
9	Перевірити записування користувача у БД	Протестовано	true	
10	Перевірити записування замовлень у БД	Протестовано	true	
11	Перевірити записування статусів замовлень у БД	Протестовано	true	
12	Перевірити працездатність усіх команд в "Меню"	Протестовано	true	
13				
14				
15				
16				

Лист1

Рисунок 3.13 – Вигляд імпровізованого чек-ліста для тестування телеграм-боту після «смоука»

Виходячи з рисунку вище, можна зробити висновок, що смоук-тестування телеграм-боту з метою покращення процесу реалізації продуктів товару завершився успішно лише на 80%, адже частина перевірок показала актуальний результат не такий, який очікувався, тобто, під час тестування були знайдені баги.

ВИСНОВКИ

Telegram - це популярний месенджер, який ефективно використовується для бізнесу. Такі інструменти як боти, чати і канали легко застосовуються для розсилок, перенаправлення клієнтів до менеджерів. На відміну від соціальних мереж та інших платформ, додаток дає можливість безпечного обміну повідомленнями з дотриманням повної конфіденційності.

Враховуючи усе вищеописане щодо Telegram-ботів, було прийнято рішення розробити власного комерційного бота, який у ході цієї дипломної був розроблений. Головною метою розробленого телеграм-боту є покращення процесу реалізації продуктів побутової хімії. Завдяки товариському інтерфейсу Telegram при роботі з ботами, вдалося реалізувати «спілкування» між користувачем та ботом за допомогою кнопок, команд та повідомлень (наприклад, кількість товару, який користувач хоче придбати).

Телеграм-бот «HandSoap Shop» був розроблений за допомогою власної авторської архітектури, яка попередньо була спроектована у вигляді mind-map.

Бот несе в собі зручність, інформативність, привітність щодо використання, можливість авторизації користувача, можливість оформлення замовлення через бота, можливість передивитися підтверджені й не підтверджені замовлення та ін. Серед асортименту товару представлені різновиди шампунів та мила ручної праці, які розділені на декілька категорій (чоловічі, жіночі, дитячі та святкові). Придбати товари можливо виключно безготівково.

З технічної сторони телеграм-бот «HandSoap Shop» написаний за допомогою мови програмування TypeScript із необхідною для розробки телеграм-боту та створення локального серверу програмної платформи Node.js. Для зберігання даних про оформлення замовлень користувачів та їх даних була використана об'єктно-реляційна система управління базами даних PostgreSQL. Також телеграм-бот був протестований за допомогою «смоук-тестування» у вигляді імпровізованого чек-ліста з функціональними перевірками.

ПЕРЕЛІК ПОСИЛАНЬ

1. Telegram FAQ [Електронний ресурс]: [Веб–сайт]. – електронні дані. – Режим доступу: https://telegram.org/faq#q-what-is-telegram-what-do-i-do-here
2. The Evolution of Telegram [Електронний ресурс]: [Веб–сайт]. – електронні дані. – Режим доступу: https://telegram.org/evolution#october-2013
3. Telegram launched two new web clients for mobile and desktop devices [Електронний ресурс]: [Веб–сайт]. – електронні дані. – Режим доступу: https://telegramm.app/telegram-launched-two-new-web-clients-for-mobile-and-desktop-devices/
4. Telegram reaches 1 billion downloads globally [Електронний ресурс]: [Веб–сайт]. – електронні дані. – Режим доступу: https://9to5mac.com/2021/08/30/telegram-reaches-1-billion-downloads-globally/
5. Why Telegram has become the hottest messaging app in the world [Електронний ресурс]: [Веб–сайт]. – електронні дані. – Режим доступу: https://www.theverge.com/2014/2/25/5445864/telegram-messenger-hottest-app-in-the-world
6. Telegram Bot Platform [Електронний ресурс]: [Веб–сайт]. – електронні дані. – Режим доступу: https://telegram.org/blog/bot-revolution
7. Gaming Platform 1.0 [Електронний ресурс]: [Веб–сайт]. – електронні дані. – Режим доступу: https://telegram.org/blog/games

8. Как использовать Telegram для бизнеса [Електронний ресурс]: [Веб–сайт]. – електронні дані. – Режим доступу: https://umom.biz/kak-ispolzovat-telegram-dlya-biznesa/

9. Чат-бот в Телеграм для бизнеса: зачем он и как создать [Електронний ресурс]: [Веб–сайт]. – електронні дані. – Режим доступу: https://profi-soft.kz/articles/zachem-biznesu-chat-bot-telegram/

10. Про нас | ROZETKA [Електронний ресурс]: [Веб–сайт]. – електронні дані. – Режим доступу: https://rozetka.com.ua/ua/pages/about/

11. Рейтинг популярних сайтів за січень 2022 [Електронний ресурс]: [Веб–сайт]. – електронні дані. – Режим доступу: https://web.archive.org/web/20220316193405/https://tns-ua.com/news/reyting-populyarnih-saytiv-za-sichen-2022

12. Про Банк [Електронний ресурс]: [Веб–сайт]. – електронні дані. – Режим доступу: https://privatbank.ua/about

13. Telegram для бизнеса [Електронний ресурс]: [Веб–сайт]. – електронні дані. – Режим доступу: https://blog.admobispy.com/reklamnye_seti/telegram_dlya_biznesa

14. Telegram Business: The Ultimate Guide (Nov 2020) [Електронний ресурс]: [Веб–сайт]. – електронні дані. – Режим доступу: https://respond.io/blog/telegram-for-business

15. TypeScript for the New Programmer [Електронний ресурс]: [Веб–сайт]. – електронні дані. – Режим доступу: https://www.typescriptlang.org/docs/handbook/typescript-from-scratch.html

16. Use Case Diagram Tutorial (Guide with Examples) [Електронний ресурс]: [Веб–сайт]. – електронні дані. – Режим

доступу: https://creately.com/blog/diagrams/use-case-diagram-tutorial/

17. The Easy Guide to UML Activity Diagrams [Електронний ресурс]: [Веб–сайт]. – електронні дані. – Режим доступу: https://creately.com/blog/diagrams/activity-diagram-tutorial/

18. Алгоритм – ВУЕ [Електронний ресурс]: [Веб–сайт]. – електронні дані. – Режим доступу: https://vue.gov.ua/Алгоритм

19. PostgreSQL: Documentation: 14: 2. A Brief History of PostgreSQL [Електронний ресурс]: [Веб–сайт]. – електронні дані. – Режим доступу: https://www.postgresql.org/docs/current/history.html

20. PostgreSQL: About [Електронний ресурс]: [Веб–сайт]. – електронні дані. – Режим доступу: https://www.postgresql.org/about/

21. The benefits of PostgreSQL [Електронний ресурс]: [Веб–сайт]. – електронні дані. – Режим доступу: https://www.prisma.io/dataguide/postgresql/benefits-of-postgresql

22. About | Node.js [Електронний ресурс]: [Веб–сайт]. – електронні дані. – Режим доступу: https://nodejs.org/en/about/

23. 7 Advantages of Node.js for Startups [Електронний ресурс]: [Веб–сайт]. – електронні дані. – Режим доступу: https://relevant.software/blog/7-benefits-of-node-js-for-startups/

24. TypeScript tutorial with Visual Studio Code [Електронний ресурс]: [Веб–сайт]. – електронні дані. – Режим доступу: https://code.visualstudio.com/docs/typescript/typescript-tutorial

25. Ярош А.О., Телеграм-бот як інструмент у бізнесі / НАУКОВО-ТЕХНІЧНА КОНФЕРЕНЦІЯ "ЗАСТОСУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ В

ІНФОКОМУНІКАЦІЙНИХ ТЕХНОЛОГІЯХ". Збірник тез 20.04.2022, ДУТ, м. Київ – К.: ДУТ, 2022. С. 162-164.

26. Ярош А.О., Переваги телеграм-боту як комерційної одиниці / ВСЕУКРАЇНСЬКА НАУКОВО-ТЕХНІЧНА КОНФЕРЕНЦІЯ "СУЧАСНІ ІНТЕЛЕКТУАЛЬНІ ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ В НАУЦІ ТА ОСВІТІ". Збірник тез 05.04.2022, ДУТ, м. Київ – К.: ДУТ, 2022. С. 162-164.

ДОДАТОК А

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

РОЗРОБКА ТЕЛЕГРАМ-БОТУ З МЕТОЮ ПОКРАЩЕННЯ ПРОЦЕСУ РЕАЛІЗАЦІЇ ПРОДУКТІВ ПОБУТОВОЇ ХІМІЇ МОВОЮ JAVASCRIPT

Виконав студент 4 курсу
Групи ПД-44
Ярош А.О.
Керівник роботи
Жебка В.В.

Київ - 2022

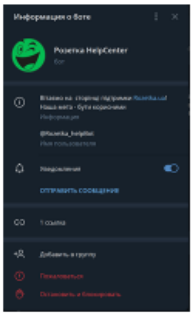
МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- **Мета дослідження** - покращення процесу реалізації товарів побутової хімії за допомогою розробленого телеграм-боту мовою JavaScript.
- **Об'єкт дослідження** - процес реалізації товарів побутової хімії за допомогою застосування Телеграм-ботів.
- **Предмет дослідження** - телеграм-бот на мові програмування JavaScript.

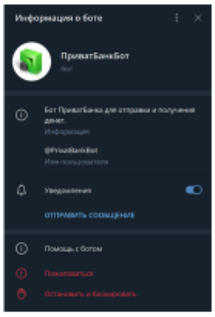
АКТУАЛЬНІСТЬ РОБОТИ

- Актуальність роботи полягає в створенні Telegram-боту, котрий буде призначений для покращення об'єму продажу товарів, а не тільки інформування. Наразі така практика дуже рідка та саме тому це має вигляд «свіжого» погляду щодо використання ботів у Telegram, використовуючи його можливості.
- Даний бот може бути використаний як основний приклад для інших Telegram-ботів у сфері бізнеса, де власник бізнеса бажає розширити розпізнаваність власних товарів та збільшення прибутку своєї бізнес діяльності.

НАЙБІЛЬШ ПОДІБНІ ДО АНАЛОГІВ



Telegram-бот Розетки



Telegram-бот ПриватБанка

ПОРІВНЯННЯ З АНАЛОГАМИ

Назва	Переваги	Недоліки
Телеграм-бот від Розетки	• Можливість зворотнього зв'язку	• Відсутність можливості придбати товар
	• Інформативність	• Незручна архітектура бота
	• Налаштування особистого кабінету	
Телеграм-бот від ПриватБанку	• Багатофункціональність	• Відсутність інформативності
	• Можливість онлайн транзакцій безпосередньо через бота	• Незручність в користуванні

ТЕХНІЧНЕ ЗАВДАННЯ

- Дослідити розвиток, можливості та актуальність ведення бізнесу в Telegram.
- Розробити архітектуру телеграм-боту у вигляді mind-map.
- Розробити структуру бази даних.
- Розробити діаграму діяльності та діаграму прецедентів.
- Реалізувати можливість авторизації користувача безпосередньо у боті.
- Реалізувати можливість для користувача продивлятися його замовлення.
- Реалізувати можливість оформляти замовлення користувачу.
- Реалізувати можливість надсилання листа на електрону пошту користувача після оформлення ним замовлення.
- Реалізувати можливість підтвердження замовлення користувача через електрону пошту.

ПРОГРАМНІ ТА ТЕХНІЧНІ ЗАСОБИ РЕАЛІЗАЦІЇ

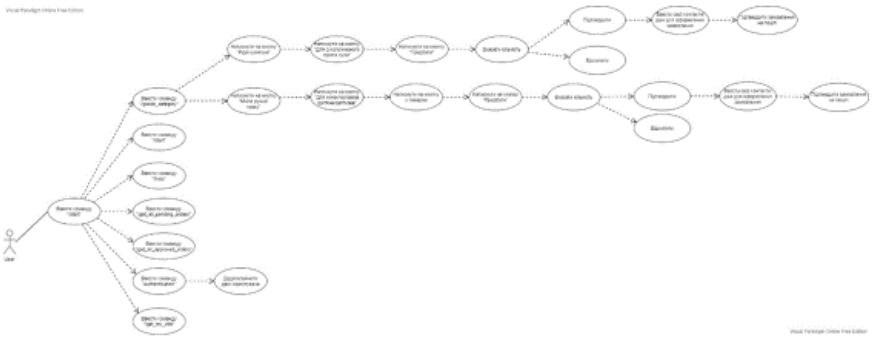
Visual Studio Code

PostgreSQL

node

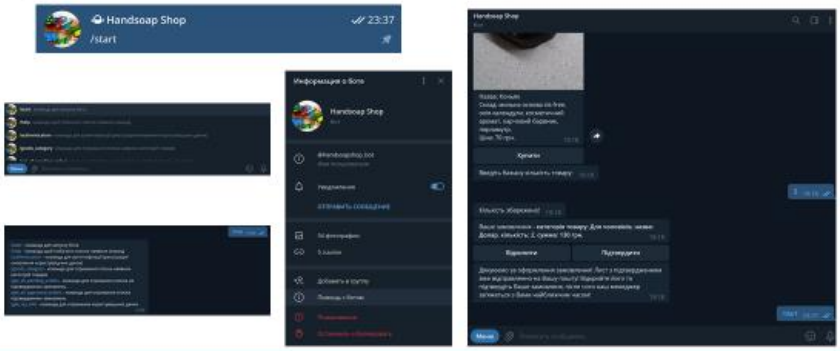
АРХІТЕКТУРА ТЕЛЕГРАМ-БОТУ

ДІАГРАМА ПРЕЦЕДЕНТІВ



ДІАГРАМА ДІЯЛЬНОСТІ

ЗОВНІШНИЙ ВИГЛЯД ТЕЛЕГРАМ-БОТУ



ВИСНОВКИ

Досліджено:

- Розвиток, можливості та актуальність ведення бізнесу в Telegram.

Розроблено:

- Архітектура телеграм-боту у вигляді mind-map.
- Структура бази даних.
- Діаграми діяльності та прецедентів.

Реалізовано:

- Можливість авторизації користувача безпосередньо у боті.
- Можливість для користувача продивлятися його замовлення.
- Можливість оформляти замовлення користувачу.
- Можливість надсилання листа на електрону пошту користувача після оформлення ним замовлення.
- Можливість підтвердження замовлення користувача через електрону пошту.

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1) Ярош А.О., Телеграм-бот як інструмент у бізнесі / НАУКОВО-ТЕХНІЧНА КОНФЕРЕНЦІЯ "ЗАСТОСУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ В ІНФОКОМУНІКАЦІЙНИХ ТЕХНОЛОГІЯХ" Збірник тез 20.04.2022, ДУТ, м. Київ – К.: ДУТ, 2022. С. 162-164.
2) Ярош А.О., Переваги телеграм-боту як комерційної одиниці / ВСЕУКРАЇНСЬКА НАУКОВО-ТЕХНІЧНА КОНФЕРЕНЦІЯ "СУЧАСНІ ІНТЕЛЕКТУАЛЬНІ ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ В НАУЦІ ТА ОСВІТІ " Збірник тез 05.04.2022, ДУТ, м. Київ – К.: ДУТ, 2022. С. 162-164.

ДЯКУЮ ЗА УВАГУ!

ДОДАТОК Б

```
import "dotenv/config"
// @ts-ignore
import {Context, Markup, Scenes, session, Telegraf} from "telegraf"
import {createConnection, getConnection} from "typeorm";
import {commands, selectCategoryInfo, selectSubCategoryInfo,Command} from "./utility";
import {User} from "./entity/User";
import * as fs from "fs";
import {Category} from "./entity/Category";
import {Goods} from "./entity/Goods";
import {Order, Status} from "./entity/Order";
import {MailService} from "./services/mail";


createConnection() //соєдіненіє з бд

const exit_keyboard = Markup.keyboard(["Вихід"]).oneTime()
const leave_keyboard = Markup.removeKeyboard()

enum SceneName {
    AUTH_SCENE = "authScene",
    ORDER_SCENE = "orderScene",
}


interface SessionData {
    email: string | null,
    phone: string | null,
    goodsId: string | null
    goodsCount: string | null
}
interface MyContext extends Context {
    session?: SessionData
}

const bot = new Telegraf<MyContext>(process.env.BOT_TOKEN!)

// need to include session to store some specific user session data

bot.use(session())

// section with stage scenario messages
const authAskEmail = async (ctx: any) => {
    try {
        await ctx.reply("Введіть Вашу електронну пошту:", exit_keyboard)
        return ctx.wizard.next()
    } catch (e) {
```

```
        console.error(e)
    }
}

const handleEmail = Telegraf.on("text", async (ctx: any) => {
    try {
        ctx.session.email = ctx.message.text
        await ctx.reply("Введіть Ваш номер телефону:", exit_keyboard)
        return ctx.wizard.next()
    } catch (e) {
        console.error(e)
    }
})

const handlePhone = Telegraf.on("text", async (ctx: any) => {
    try {
        await ctx.reply("Дякую за внесення повної інформації!", leave_keyboard)
        const res = await getConnection().createQueryBuilder().update(User).set({
            email: ctx.session.email,
            phone: ctx.message.text
        }).where("tgId= :tgId", {tgId: ctx.from.id}).execute()
        if (ctx.session.goodsId) {
            if (res) {
                await ctx.scene.leave()
                const buttons = [Markup.button.callback("Ні", "continue_decline"), Markup.button.callback("Так",
`order_${ctx.session.goodsId}`)]
                await ctx.replyWithHTML("Повернутись до замовлення?", Markup.inlineKeyboard([buttons]))
            } else {
                await ctx.reply("Щось пішло не так...")
                return ctx.scene.leave()
            }
        } else {
            return ctx.scene.leave()
        }
    } catch (e) {
        console.error(e)
    }
})

const askCount = async (ctx: any) => {
    try {
        await ctx.reply("Введіть бажану кількість товару:", exit_keyboard)
        return ctx.wizard.next()
    } catch (e) {
        console.error(e)
    }
}

const handleOrderCount = Telegraf.on("text", async (ctx: any) => {
```

```
    try {
        ctx.session.orderCount = +ctx.message.text
        const goods = await getConnection().getRepository(Goods).findOne(ctx.session.goodsId)
        if (goods && goods.id) {
            await ctx.reply("Кількість збережено!", leave_keyboard)
            ctx.scene.leave()
            const buttons = [Markup.button.callback("Відхилити", "decline_order"),
Markup.button.callback("Підтвердити", "accept_order")]
            await ctx.replyWithHTML(`Ваше замовлення - <b>категорія товару: ${goods.category.name}</b>,
<b>назва: ${goods.name}</b>, <b>кількість: ${ctx.message.text}</b>, <b>сумма: ${+ctx.message.text * goods.price}
грн.</b>`,
                Markup.inlineKeyboard([buttons]))
        } else {
            await ctx.reply("Щось пішло не так...")
        }
    } catch (e) {
        console.error(e)
    }
})

// scene start a chain of user interaction

const authScene = new Scenes.WizardScene(SceneName.AUTH_SCENE, authAskEmail, handleEmail, handlePhone)
const orderScene = new Scenes.WizardScene(SceneName.ORDER_SCENE, askCount, handleOrderCount)
const stage = new Scenes.Stage([authScene, orderScene])
stage.hears('exit', (ctx: any) => ctx.scene.leave())

// need to provide stages to context

bot.use(stage.middleware())

//command triggers when user click or send message with one of them

bot.start(async (ctx: any) => {
    try {
        const tgId = ctx.from.id
        const userName = ctx.from.username
        const firstName = ctx.from.first_name
        const lastName = ctx.from.last_name
        const user = await getConnection().getRepository(User).findOne({
            where: {tgId}
        })
        if (user) {
            ctx.reply(commands)
        } else {
            const res = await getConnection().getRepository(User).insert({
                tgId,
                userName,
                firstName,
```

```
                lastName
            })
            if (res) {
                ctx.reply("Успішно авторизовано!")
            }
        }
    } catch (e) {

    }
})

bot.help((ctx: Context) => ctx.reply(commands))

bot.command(Command.AUTHENTICATION, async (ctx: any) => {
    try {
        await ctx.scene.enter(SceneName.AUTH_SCENE)
    } catch (e) {
        console.error(e)
    }

})


bot.command(Command.GOODS_CATEGORY, async (ctx: Context) => {
    try {
        const res = await getConnection().getRepository(Category).find({where: {isSubCategory: false}})
        if (res.length > 0 && res) {
            const buttons: any[] = []
            for (let i = 0; i < res.length; i++) {
                buttons.push(Markup.button.callback(`${res[i].name}`, `select_category_${res[i].id}`))
            }
            await ctx.replyWithHTML(selectCategoryInfo, Markup.inlineKeyboard([
                buttons
            ]))
        } else {
            await ctx.reply("Категорій не знайдено!")
        }
    } catch (e) {
        console.error(e)
    }

})

bot.command(Command.GET_ALL_PENDING_ORDERS, async (ctx: any) => {
    try {
        const user = await getConnection().getRepository(User).findOne({
            where: {
                tgId: ctx.from.id
            }
```

```
        })
        if (user) {
            const orders = await getConnection().getRepository(Order).find({
                where: {
                    user,
                    status: Status.Pending
                }
            })
            if (orders) {
                if (orders.length > 0) {
                    for (let i = 0; i < orders.length; i++) {
                        const button = [Markup.button.callback("Відправити нове підтвердження!",
`send_new_approval_${orders[i].id}`)]
                        await ctx.reply(`Дата створення:
${orders[i].createdDate.toLocaleDateString()}\n\nКатегорія товару: ${orders[i].goods.name}\nНазва товару:
${orders[i].goods.category.name}\nКількість: ${orders[i].count}\nЦіна: ${orders[i].goods.price} грн.\nСумма:
${orders[i].count * orders[i].goods.price} грн.`, Markup.inlineKeyboard([
                            button
                        ]))
                    }
                }else{
                    await ctx.reply(`Наразі у Вас немає непідтвердженних замовлень.`)
                }
            }
        }
    } catch (e) {

    }
})

bot.command(Command.GET_ALL_APPROVED_ORDERS, async (ctx: any) => {
    try {
        const user = await getConnection().getRepository(User).findOne({
            where: {
                tgId: ctx.from.id
            }
        })
        if (user) {
            const orders = await getConnection().getRepository(Order).find({
                where: {
                    user,
                    status: Status.Approved
                }
            })
            if (orders) {
                if(orders.length > 0) {
                    for (let i = 0; i < orders.length; i++) {
                        await ctx.reply(`Дата створення:
${orders[i].createdDate.toLocaleDateString()}\n\nКатегорія товару: ${orders[i].goods.name}\nНазва товару:
```

```
${orders[i].goods.category.name}\nКількість: ${orders[i].count}\nЦіна: ${orders[i].goods.price} грн.\nСумма:
${orders[i].count * orders[i].goods.price} грн.`)
                    }
                }else{
                    await ctx.reply(`Наразі у Вас немає підтвердженних замовлень.`)
                }
            }
        }
    } catch (e) {

    }
})

bot.command(Command.GET_MY_INFO, async (ctx: any) => {
    try {
        const user = await getConnection().getRepository(User).findOne({
            where: {
                tgId: ctx.from.id
            }
        })
        if (user) {
            ctx.replyWithHTML(`Ім'я користувача: ${user.firstName + " " + user.lastName} \nПошта: ${user.email ?
user.email : "Наразі пошта не прив'язана!"} \nТелефон: ${user.phone ? user.phone : "Наразі телефон не
прив'язано!"}`)
        }
    } catch (e) {
        console.error(e)
    }
})

//action triggers on btn click

bot.action("decline_order", async (ctx: any) => {
    try {
        ctx.session.goodsId = undefined
        ctx.session.orderCount = undefined
        ctx.reply("Замовлення видалено!")
    } catch (e) {
        console.error(e)
    }
})

bot.action("continue_decline", async (ctx: any) => {
    try {
        ctx.session.goodsId = undefined
        ctx.session.email = undefined
        ctx.reply("Замовлення видалено!")
    } catch (e) {
        console.error(e)
```

```
    }
})

bot.action("accept_order", async (ctx: any) => {
    try {
        const goodsId = ctx.session.goodsId
        const orderCount = ctx.session.orderCount
        const user = await getConnection().getRepository(User).findOne({
            where: {
                tgId: ctx.from.id
            }
        })
        const goods = await getConnection().getRepository(Goods).findOne({
            where: {
                id: goodsId
            }
        })
        const res = await getConnection().getRepository(Order).insert({
            user,
            goods,
            count: orderCount
        })
        if (res && user) {
            console.log(user.email)
            const newOrder = await getConnection().getRepository(Order).findOne(res.raw[0].id)
            if (newOrder) {
                MailService.confirmOrder(user.email, newOrder)
                ctx.reply("Дякуємо за оформлення замовлення! Лист з підтвердженням вже відправленно на Вашу пошту! Відкрийте його та підтвердіть Ваше замовленя, після чого наш менеджер зв'яжеться з Вами найближчим часом!")
            }
        }
    } catch (e) {
        console.error(e)
    }
})


bot.action(/order_+/, async (ctx: any) => {
    try {
        const goodsId = ctx.callbackQuery!.data!.replace("order_", "")
        ctx.session.goodsId = goodsId
        const user = await getConnection().getRepository(User).findOne({
            where: {
                tgId: ctx.from.id
            }
        })
        if (user) {
            if (user.email && user.phone) {
```

```
                await ctx.scene.enter(SceneName.ORDER_SCENE)
            } else {
                await ctx.reply(`Наразі у Вас заповненна не вся інформація. Будь-ласка додайте свою пошту та
номер телефону! \nЯкщо ви хочете авторизуватись: /${Command.AUTHENTICATION}`)
            }
        } else {
            await ctx.reply('Щось пішло не так... Будь-ласка використайте команду /start')
        }
    } catch (e) {
        console.error(e)
    }
})


bot.action(/send_new_approval_+/, async (ctx: any) => {
    try {
        const orderId = ctx.callbackQuery!.data!.replace("send_new_approval_", "")
        const user = await getConnection().getRepository(User).findOne({
            where: {
                tgId: ctx.from.id
            }
        })
        const order = await getConnection().getRepository(Order).findOne(orderId)
        if (order && user) {
            MailService.confirmOrder(user.email, order)
        }
    } catch (e) {

    }
})


bot.action(/select_category_+/, async (ctx: Context) => {
    try {
        const categoryId = ctx.callbackQuery!.data!.replace("select_category_", "")
        const category = await getConnection().getRepository(Category).findOne(categoryId)
        if (category) {
            if (category.hasSubCategories) {
                const subCategories = await getConnection().getRepository(Category).find({
                    where: {
                        parentId: categoryId
                    }
                })
                if (subCategories.length > 0 && subCategories) {
                    const buttons: any[] = []
                    for (let i = 0; i < subCategories.length; i++) {
                        buttons.push(Markup.button.callback(`${subCategories[i].name}`,
`select_sub_category_${subCategories[i].id}`))
                    }
```

```
                    await ctx.replyWithHTML(selectSubCategoryInfo, Markup.inlineKeyboard([
                        buttons
                    ]))
                } else {
                    await ctx.reply("Категорій не знайдено!")
                }
            } else {
                const goods = await getConnection().getRepository(Goods).find({where: {category}})
                if (goods.length > 0 && goods) {
                    for (let i = 0; i < goods.length; i++) {
                        await ctx.replyWithPhoto({
                            source: fs.createReadStream(`./src/assets/${goods[i].imgSrc}`)
                        }, {
                            caption: `Назва: ${goods[i].name} \nСклад: ${goods[i].description} \nЦіна:
${goods[i].price} грн.`,
                            parse_mode: 'Markdown',
                            ...Markup.inlineKeyboard([
                                    [Markup.button.callback("Придбати!", `order_${goods[i].id}`)]
                                ]
                            )
                        })
                    }
                } else {
                    await ctx.reply("Товарів не знайдено!")
                }
            }
        }
    } catch (e) {
        console.error(e)
    }
})

bot.action(/select_sub_category+/, async (ctx: Context) => {
    try {
        const subCategoryId = ctx.callbackQuery!.data!.replace("select_sub_category_", "")
        if (subCategoryId) {
            const category = await getConnection().getRepository(Category).findOne(subCategoryId)
            const goods = await getConnection().getRepository(Goods).find({where: {category}})
            if (goods.length > 0 && goods) {
                for (let i = 0; i < goods.length; i++) {
                    await ctx.replyWithPhoto({
                        source: fs.createReadStream(`./src/assets/${goods[i].imgSrc}`)
                    }, {
                        caption: `Назва: ${goods[i].name} \nСклад: ${goods[i].description} \nЦіна:
${goods[i].price} грн.`,
                        parse_mode: 'Markdown',
                        ...Markup.inlineKeyboard([
                                [Markup.button.callback("Придбати!", `order_${goods[i].id}`)]
                            ]
```

```
                        )
                    })
                }
            } else {
                await ctx.reply("Товарів не знайдено!")
            }
        }
    } catch (e) {
        console.error(e)
    }
})

// start bot

bot.launch()

process.once('SIGINT', () => bot.stop('SIGINT'))
process.once('SIGTERM', () => bot.stop('SIGTERM'))
```