Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук
(повна назва)

Кафедра програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження методів прогнозування при створенні Телеграм-бота для допомоги покупцям при пошуку ігрових цінностей
(тема)

Виконав:
студент 2 курсу, групи ІПЗм-22-2

Сидоренко О.О.
(прізвище, ініціали)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)
Тип програми освітньо-наукова

Керівник к.т.н., доц. Голян В.В.
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри _____ З.В.Дудар
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерних наук
Кафедра програмної інженерії
Рівень вищої освіти другий (магістерський)
Спеціальність 121 – Інженерія програмного забезпечення
Тип програми освітньо-наукова програма
Освітня програма Інженерія програмного забезпечення
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«___» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Сидоренко Олексію Олександровичу
(прізвище, ім’я, по батькові)

1. Тема роботи «Дослідження методів прогнозування при створенні Телеграм-бота для допомоги покупцям при пошуку ігрових цінностей»
Затверджена наказом по університету від 29.03.2024р. № 250 Ст
2. Термін подання студентом роботи до екзаменаційної комісії 25.06.2024
3. Вихідні дані до роботи наукові статті та публікації за темою роботи, відкриті дані у різних форматах, використання мови Python для розробки
4. Перелік питань, що потрібно опрацювати в роботі
вступ, аналіз предметної галузі, аналіз стану проблеми, постановка задачі, опис можливості використання отриманих результатів у науковій і практичній діяльності, експериментальні дослідження, висновки, перелік джерел посилання, додатки

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Теоретичний аналіз предметної області та постановка задачі	29.02 – 10.03.24	*виконано*
2	Проектування програмного продукту	11.03 – 26.03.24	*виконано*
3	Розробка програмного продукту	27.03 – 13.04.24	*виконано*
4	Планування експериментів	14.04 – 10.05.24	*виконано*
5	Тестування програмного продукту	11.05 – 23.05.23	*виконано*
6	Підготовка пояснювальної записки	24.05 – 16.06.24	*виконано*
7	Підготовка презентації та доповіді	16.06 – 17.06.24	*виконано*
8	Перевірка роботи на плагіат та нормоконтроль	18.06 – 20.06.24	*виконано*
9	Рецензування	20.06 – 21.06.24	*виконано*
10	Занесення диплома в електронний архів	22.06.2024	*виконано*
11	Допуск до захисту у зав. кафедри	23.06.2024	*виконано*

Дата видачі завдання 29 лютого 2024р.

Студент (ка) _____ Сидоренко О.О.
(підпис)

Керівник роботи _____ к.т.н., доц. Голян В.В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Пояснювальна записка містить: 70 с., 12 рис., 2 табл., 37 джерел.

БІРЖА, БОТ, ІГРОВІ ЦІННОСТІ, ПОКУПЕЦЬ, СПОВІЩЕННЯ, FUNPAY, PYTHON, TELEGRAM.

Об'єктом дослідження даної роботи є різні методи прогнозування, які можуть бути використані при створенні Telegram-бота для поліпшення процесу пошуку та придбання ігрових цінностей на платформі FunPay.

Метою даної роботи є дослідження та порівняння різних методів прогнозування, які можуть бути використані при створенні Telegram-бота для покращення процесу пошуку та придбання ігрових цінностей на платформі FunPay.

У ході дослідження проведено аналіз вимог до програмного забезпечення Telegram-бота, розглянуті різноманітні архітектурні підходи та інструменти, придатні для реалізації даного програмного продукту..

У результаті роботи повинно бути проведено проектування та розробка програмного забезпечення для Telegram-бота.

EXCHANGE, BOT, GAME VALUES, BUYER, NOTIFICATIONS, FUNPAY, PYTHON, TELEGRAM.

The subject of investigation in this work is various forecasting methods that can be employed in creating a Telegram bot to enhance the process of searching for and acquiring in-game assets on the FunPay platform.

The aim of this work is to explore and compare different forecasting methods that can be utilized in the development of a Telegram bot to improve the process of searching for and purchasing in-game assets on the FunPay platform.

Throughout the research, an analysis of the software requirements for the Telegram bot has been conducted, and diverse architectural approaches and tools suitable for the implementation of this software product have been examined.

The ultimate goal of the project is to design and develop the software for the Telegram bot based on the identified optimal methods and tools uncovered during the course of the research.

Заява щодо самостійного виконання кваліфікаційної роботи та можливості її публікації в електронному архіві відкритого доступу ElArKhNURE.

Я, Сидоренко Олексій Олександрович, студент гр. ІПЗм-22-2, здобувач вищої освіти на другому (магістерському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Дослідження методів прогнозування при створенні Телеграм-бота для допомоги покупцям при пошуку ігрових цінностей», що буде представлена в екзаменаційну комісію для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElArKhNURE. Всі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

API - Application Programming Interface

HTTP – Hypertext Transfer Protocol

JSON - JavaScript Object Notation

RMT - Real Money Trading

SQL - Structured query language

UML - Unified Modeling Language

XML – Extensible Markup Language

ВСТУП

Сучасний розвиток технологій визначає нові вимоги та можливості для взаємодії людей у віртуальному ігровому середовищі [1]. Проект, який розглядається у цьому дослідженні, націлений на вирішення актуальної проблеми – покращення процесу пошуку ігрових цінностей для користувачів біржі FunPay [2].

Сучасний ігровий ландшафт, насичений різноманітністю віртуальних світів та ігор, не тільки розважає, але й створює власні екосистеми та економічні реалії. В цьому контексті виникає термін RMT (Real Money Trading), що визначає обмін віртуальними активами на реальні гроші. RMT дозволяє гравцям заробляти на своєму хобі, а також отримувати доступ до бажаних ігрових ресурсів без затрат часу та зусиль.

Заохочений зростанням популярності RMT, FunPay стає ключовим гравцем у сфері торгівлі віртуальними цінностями. Як інноваційна платформа, FunPay забезпечує зручний та безпечний обмін ігровими активами, створюючи унікальний простір для ігрової економіки.

Цей проект орієнтований на галузь віртуальних ігор та торгівлі ігровими активами. Він спрямований на створення Telegram-бота, який стане надійним помічником для гравців у пошуку та придбанні цінних ігрових ресурсів.

Головною метою розробки є дослідження та впровадження ефективних методів прогнозування для оптимізації процесу взаємодії гравців з ігровими платформами через Telegram-бота. Задачі включають аналіз вимог до програмного забезпечення, вибір архітектурних рішень та визначення кращих методів прогнозування.

Зростання популярності віртуальних ігор та платформи FunPay, спрямованої на обмін ігровими активами, визначає високу актуальність цього дослідження.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

Перед тим як розпочати проектування та розробку програмного продукту, необхідно детально дослідити існуючі аналоги і визначити їх переваги та недоліки. У цьому розділі здійснено огляд наявних теоретичних та практичних розробок, що стосуються автоматизації процесу пошуку необхідного лоту, висвітлено актуальність розв'язання даної проблеми, проведений аналіз використаної літератури, визначена мета цієї роботи та конкретизовані основні завдання. Отримані результати огляду допоможуть визначити наявні прогалини і визначити напрямки подальших досліджень та розробок у даній області.

1.1 Аналіз взаємодії покупця з біржею при покупці товару

FunPay – це велика геймерська біржа, де можна придбати та продати різноманітні товари для ігор, такі як зброя, золото, внутрішньоігрову валюта, скіни та інші внутрішньоігрові товари. Сайт FunPay був заснований у 2015 році і має оригінальний дизайн у вигляді всесвіту з планетами і типографікою. Кожна гра представлена своєю планетою з ілюстрацією, а категорії товарів – супутниками (див. рис. 1.1).

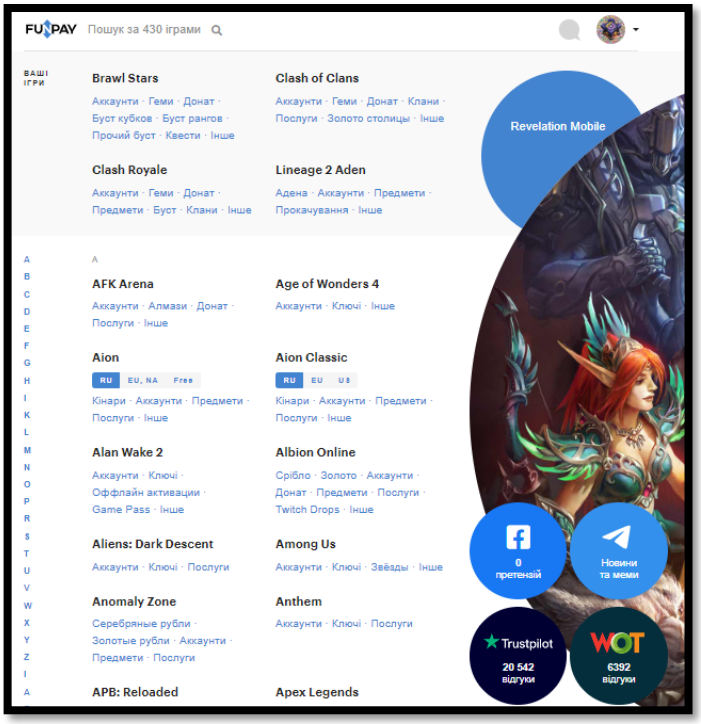

Рисунок 1.1 – Головна сторінка FunPay (рисунок створено самостійно)

Для того, щоб почати торгівлю на FunPay, потрібно зареєструватися на сайті і вибрати товар, який ви хочете купити або продати. При оплаті товару комісія залежить від обраної платіжної системи та комісії певного розділу сайту.

Після успішної угоди гроші на сайті заморожуються до 2 днів, а комісія при виведенні становить 3%. Проте, на FunPay є можливість моментального виведення коштів для довірених продавців.

Тож, щоб покупцеві придбати товар на біржі спочатку варто обрати необхідну гру на головній сторінці сайту. Потім покупцеві потрібно з таблиці обрати необхідний лот (див. рис. 1.2) та зв`язатися з продавцем через внутрішній чат сайту.

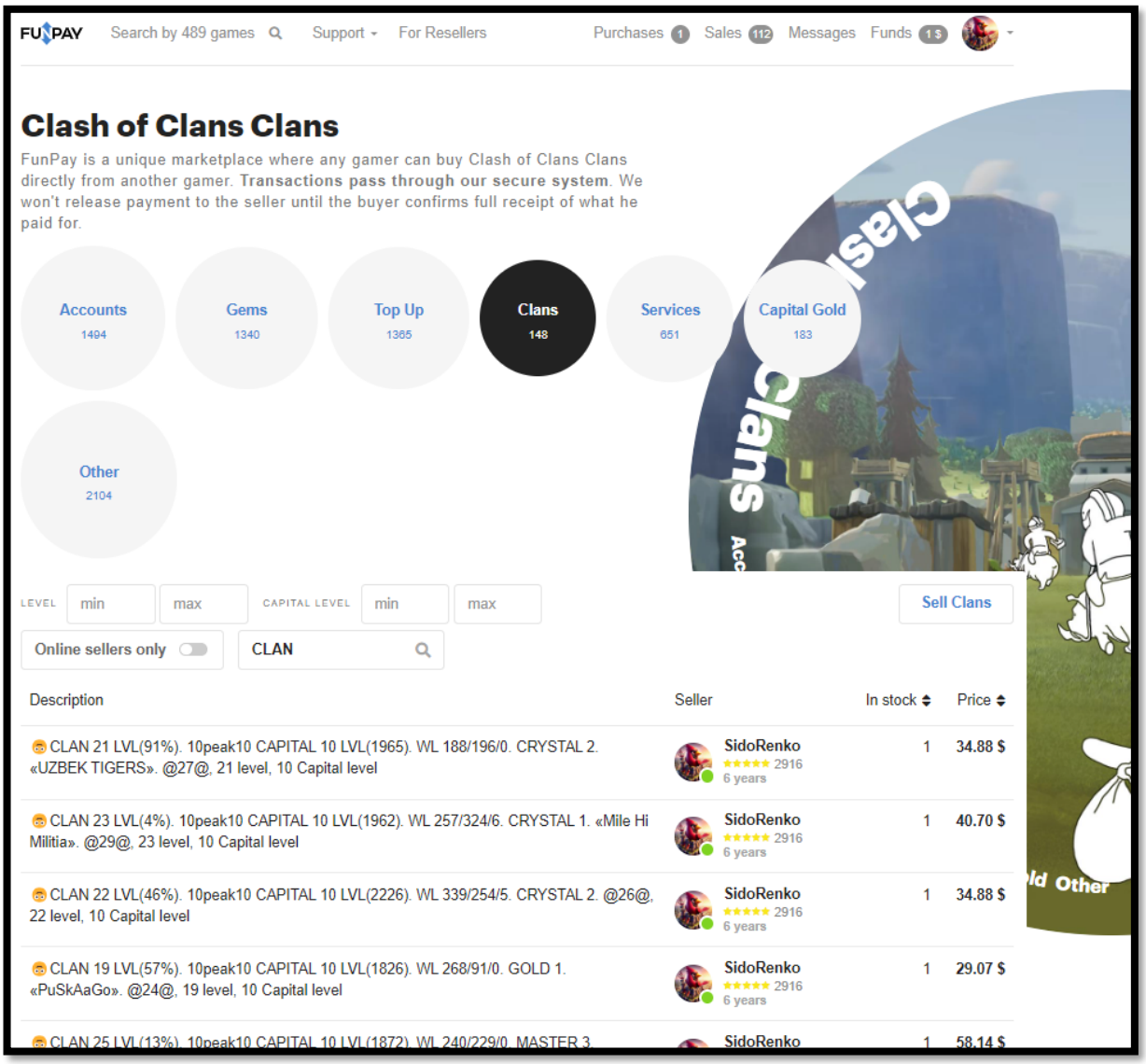

Рисунок 1.2 – Вигляд таблиці пропозицій (рисунок створено самостійно)

Після обговорення усіх деталей (термінів виконання замовлення, можливих ризиків, тощо) з продавцем покупець здійснює оплату через сайт (див. рис. 1.3).

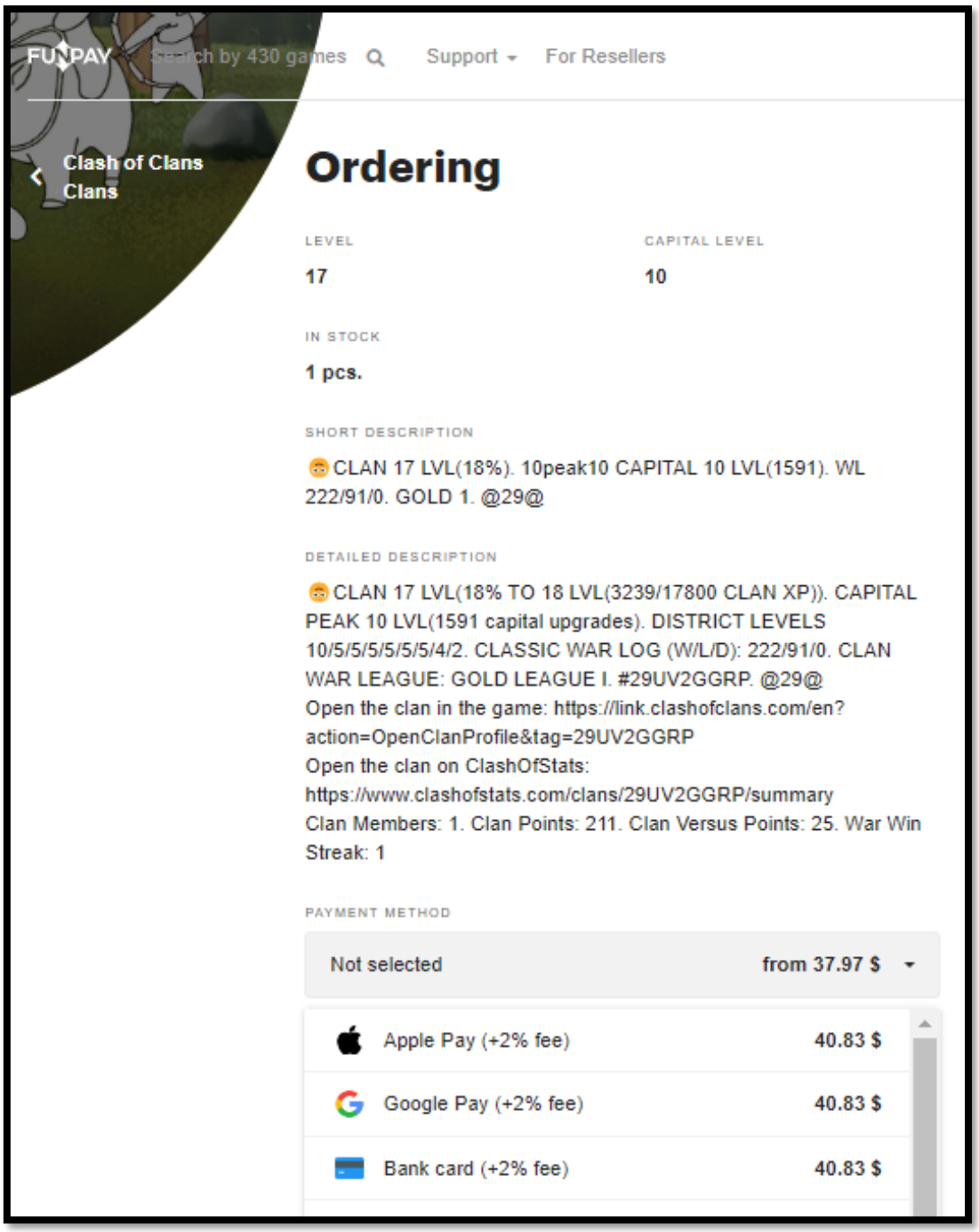

Рисунок 1.3 – Сторінка лоту (рисунок створено самостійно)

Після оплати замовлення покупцем продавець виконує усі свої зобов'язання – передає ресурси, акаунт, надає певні послуги, допомагає покупцю з усіма його запитаннями, тощо. Покупець підтверджує виконання замовлення і тим самим завершує переказ коштів продавцю (див. рис. 1.4).

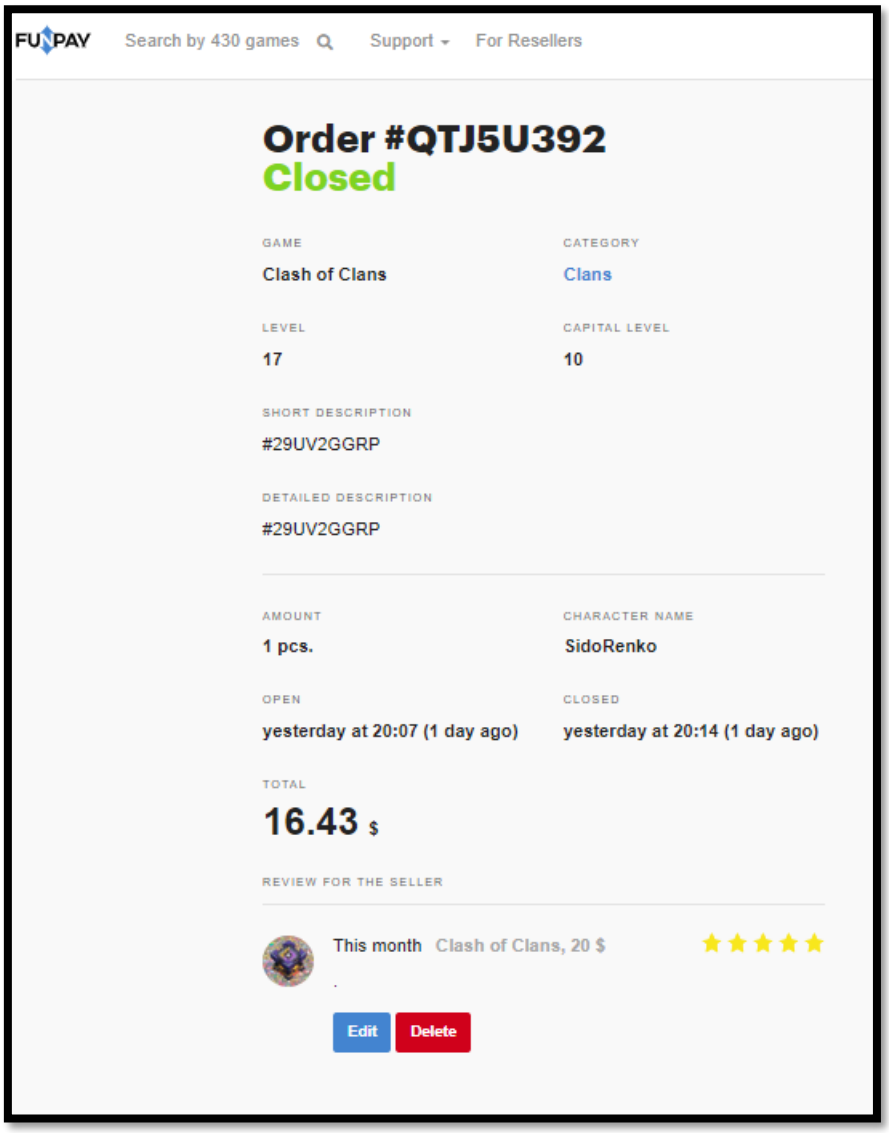

Рисунок 1.4 – Сторінка завершеного замовлення (рисунок створено самостійно)

Таким чином, всі замовлення на FunPay захищені: продавець не може отримати гроші доти, доки замовлення не буде завершено. У разі будь-яких розбіжностей (наприклад, якщо продавець не виконав своїх зобов'язань), FunPay рекомендує звернутися до служби підтримки [3].

Однак, самостійний моніторинг таблиць пропозицій має певні недоліки, серед яких важливо враховувати:

- витрати часу: пошук і відстеження вигідних пропозицій вручну може займати значний час, особливо якщо таблиця пропозицій має велику кількість лотів;

- ризик пропуску вигідних пропозицій: відсутність автоматичних повідомлень може призвести до пропуску вигідних угод або знижок, оскільки користувач не завжди може вчасно відслідковувати всі оновлення;
- можливість помилок: при ручному моніторингу існує ризик допущення помилок при введенні або розумінні інформації про пропозиції, що може призвести до невдалих транзакцій;
- неефективність фільтрів: таблиці можуть містити велику кількість даних, і неефективність фільтрації може ускладнити пошук конкретних товарів;
- брак системи сповіщень: відсутність системи автоматичних сповіщень про зміни в пропозиціях або цінах може призвести до втрати можливостей або затримки в реакції на зміни ринку.

Водночас, наявність системи сповіщень вирішила б усі ці проблеми, однак, у біржі є більш пріоритетні задачі та навіть їх вона виконує досить повільно (навіть така потрібна функція, як автоматична видача товару з`явилася лише 9 вересня 2023 року, через 8 років після створення біржі [4]).

1.2 Аналіз аудиторії

Аналіз аудиторії користувачів месенджерів та користувачів біржі FunPay важливий для розуміння їхніх характеристик, потреб та вподобань. Нижче наведено загальний аналіз обох аудиторій.

Аудиторія користувачів біржі FunPay:

- геймерська спрямованість: користувачі біржі FunPay в основному складаються з геймерів та ентузіастів ігор, які шукають можливості для купівлі та продажу ігрових ресурсів;
- вікова група: в аудиторії FunPay переважають молоді та середнього віку гравці, оскільки геймінг є основною зайнятістю для цієї категорії користувачів;
- інтерес до ігор: користувачі цікавляться різноманітними іграми і активно взаємодіють з FunPay для полегшення процесу обміну та покупки ігрових активів;

- економічні мотиви: багато користувачів FunPay використовують платформу як засіб заробітку або для економії грошей на придбанні ігрових ресурсів.

Аудиторія користувачів месенджерів:

- вікова категорія: більшість користувачів месенджерів охоплюють широкий віковий спектр, від підлітків до дорослих;
- цільова група: месенджери використовуються як для особистого спілкування, так і для бізнесу. Вони популярні серед студентів та молоді;
- функціональність: користувачі месенджерів шукають швидкого та зручного способу спілкування, обміну файлами, мультимедійним контентом та ведення групових чатів;
- мобільність: основна частина користувачів месенджерів використовує їх на мобільних пристроях, що підкреслює важливість мобільності та зручності в їхньому використанні.

Статистика месенджера Telegram:

- у 2024 році аудиторія Telegram складає 800 мільйонів активних користувачів на місяць. Це означає, що Telegram використовує 9.9% населення Землі щомісяця [5];
- щодня в Telegram реєструється 2,5 млн нових користувачів [6];
- починаючи з 2013 року аудиторія Telegram зростає не менше ніж на 40% щороку [7];
- за останні п'ять років аудиторія Telegram зросла у 3.5 рази;
- у січні 2021 року аудиторія Telegram зросла до 500 млн. активних користувачів на місяць. За словами Павла Дурова, тоді до месенджера приєдналося 25 млн нових користувачів протягом 3-х днів;
- 5 жовтня 2021 року Telegram отримав ще близько 70 млн нових користувачів (це сталося внаслідок збою у Facebook);
- у липні 2023 року Павло Дуров повідомив, що аудиторія месенджера досягла 800 активних користувачів на місяць;

- у жовтні 2023 року Telegram посідав 8-е місце в рейтингу найвідвідуваніших соціальних мереж світу;
- за даними GrabOn [8] у Telegram найбільше користувачів у віці від 25 до 34 років (31.31%). На другому місці – вікова група віком від 18 до 24 років (24.57%).

1.3 Аналіз існуючих рішень та постановка задачі

Наразі існує низка ботів для спрощення взаємодії користувачів біржі FunPay з сайтом. Однак, здебільшого, вони створені для продавців, а не для покупців.

З найбільш популярних можна відмітити FunPay Cardinal [9], FunPay Vertex [10], FunPay Server [11], FunPay Helper, FunPay Manager [12].

Функціонал цих програм, здебільшого, повторює один одного та має такі переваги перед звичайним користуванням сайтом:

- автоматичне підняття пропозицій;
- автоматичну видачу товарів;
- автоматичну відповідь на повідомлення;
- якийсь збір статистики профілю;
- можливість налаштування чорного списку;
- відповідь на підтвердження замовлення;
- певний функціонал для отримання сповіщень у Telegram.

Усі ці програми вимагають від користувача тримати застосунок увімкненим для роботи. Однак, увесь цей функціонал не охоплює потреби покупців.

Не було знайдено жодної повноцінної програми, що задовільнила б потреби покупців (сповіщення про зміни у таблицях пропозицій). На GitHub було знайдено лише 2 репозиторії з певними напрацюваннями з цієї теми – FunPay Parser від користувача JustLike420 [13] та від користувача chyornyy [14], однак наразі їх неможливо використовувати для отримання сповіщень.

Для досягнення визначеної мети роботи, є необхідним проведення теоретичного аналізу існуючих методів прогнозування в області розробки Телеграм-ботів, аналіз ринку ігрових цінностей та вимог користувачів до таких

систем, розробка і тестування програмного забезпечення бота, що здатний ефективно допомагати в пошуку ігрових ресурсів, та оцінка ефективності запропонованого рішення.

2 ПРОЕКТУВАННЯ ПРОГРАМНОГО ПРОДУКТУ

2.1 Концептуальна модель програмного продукту

Щоб пояснити, як працює програмне забезпечення, потрібно створити концептуальну модель системи. Ця модель повинна відображати основні аспекти предметної області, тобто містити інформацію про всіх основних учасників, які взаємодіють у процесах системи.

Тож, актори, що взаємодіють з програмним продуктом такі:

- користувач: здебільшого, постійні користувачі біржі FunPay, які шукають товари для покупки чи перепродажу.
- адміністратор: особа, яка відповідає за контроль та управління функціоналом системи.

Основними прецедентами є:

а) Виконавець – користувач. Реалізовано у додатку Telegram. Опис дій:

1) запуск бота;
2) налаштування заготовок для пошуку (вибір гри, вибір характеристик лотів тощо);
3) можливість одноразового пошуку з вивантаженням усієї доступної інформації про лоти у Excel-таблицю;
4) можливість налаштування сповіщень про зміни у таблицях пропозицій (зміна ціни, характеристик лоту тощо);
5) отримання сповіщень.

б) Виконавець – адміністратор. Реалізовано у додатку Telegram. Обмежує та надає доступ користувачам до Telegram-бота.

Діаграма прецедентів в UML (Unified Modeling Language) - це структурний вид діаграм, який використовується для візуалізації функціональності системи та взаємодій між її компонентами. Вона дозволяє моделювати функціональність системи з точки зору користувачів та їх взаємодій з системою [15].

Опис прецедентів та взаємодія акторів з програмним продуктом представлені графічно на діаграмі прецедентів (див. рис. 2.1).

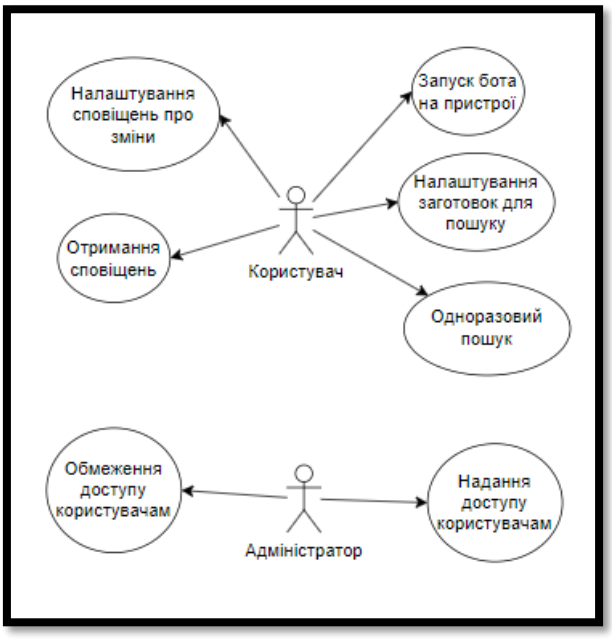

Рисунок 2.1 – Діаграма прецедентів системи (рисунок створено самостійно)

Основні елементи діаграми прецедентів включають:

- прецеденти (Use Cases): Представляють конкретні функціональні можливості системи, які можуть бути використані користувачами або іншими системами;
- актори: Представляють ролі, які взаємодіють з прецедентами. Актор може бути конкретним користувачем, групою користувачів або іншою системою;
- відносини між прецедентами і акторами: Показують, як актори взаємодіють з прецедентами.

Діаграма прецедентів допомагає визначити функціональні вимоги до системи, з'ясувати, які функції доступні різним акторам і як вони взаємодіють між собою. Це допомагає команді проектувальників та розробників краще розуміти потреби користувачів та створювати систему, яка відповідає цим потребам.

Процес створення заготовки для пошуку може бути представлено діаграмою діяльності (див. рис. 2.2).

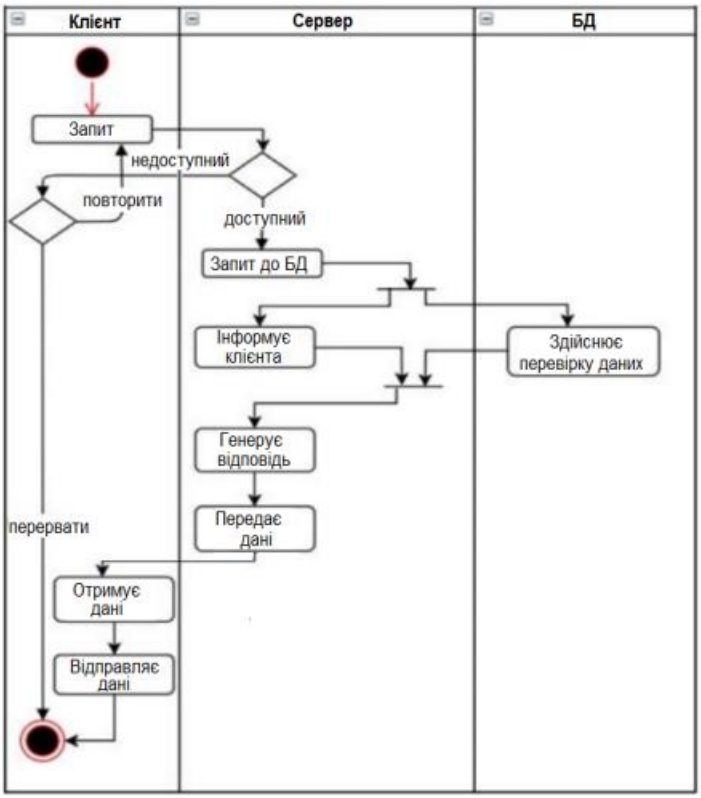

Рисунок 2.2 – Діаграма діяльності (рисунок створено самостійно)

Діаграма діяльності в UML є графічним інструментом для моделювання процесів та алгоритмів. Вона використовується для візуального представлення послідовності дій, які відбуваються в системі або процесі. Діаграми діяльності надають можливість уявно відобразити кроки або дії, взаємодії між суб'єктами системи, об'єктами та зовнішніми сутностями. [16]

Основні елементи діаграми діяльності включають:

- дії (Actions): Кроки або операції, які виконуються в процесі;
- рішення (Decisions): Умовні галузі, які визначають різні шляхи в процесі в залежності від умов;
- вершини (Nodes): Точки в діаграмі, які представляють дії або стан системи;
- флов-чарти (Flowcharts): Стрілки або лінії, які вказують на порядок виконання дій та переходи між станами;
- об'єкти (Objects): Сутності або об'єкти, які взаємодіють в процесі.

Діаграми діяльності допомагають візуалізувати, розуміти та аналізувати послідовності операцій у системі. Вони широко використовуються під час аналізу та проектування систем для опису бізнес-процесів, алгоритмів та внутрішньої логіки програм.

2.2 Компонентна архітектура

Компонентна архітектура — це стиль побудови програмних систем, в якому система розбивається на окремі компоненти або модулі. Кожен компонент відповідає за певну функціональність та може бути реалізований, тестований та підтримуваний незалежно від інших компонентів. [17]

Основні характеристики компонентної архітектури:

- розширюваність (Scalability): Здатність системи легко змінювати та розширюватися за допомогою додавання або заміни компонентів;
- підтримка підходу "розділення на рівні сервісів" (Service Level Separation): Кожен компонент може надавати окремий сервіс або функціональність;
- незалежність компонентів (Component Independence): Компоненти можуть функціонувати незалежно один від одного, і зміни в одному компоненті не повинні впливати на інші;
- модульність (Modularity): Система розділена на модулі (компоненти), що полегшує розробку, тестування та підтримку;
- інтерфейси (Interfaces): Компоненти взаємодіють між собою через чітко визначені інтерфейси, що спрощує їхню інтеграцію;
- широка використаність (Reusability): Можливість використовувати компоненти в інших частинах системи або в інших програмах;
- підтримка різноманітності технологій (Technology Diversity): Кожен компонент може бути реалізований з використанням різних технологій, які найкраще підходять для конкретного завдання.

Компонентна архітектура дозволяє розробникам створювати більш гнучкі, легко змінювані та розширювані системи, що є важливим у сучасному програмному проектуванні.

Розглянемо два підходи для реалізації програмного забезпечення для покращення взаємодії покупців з біржею FunPay:

- монолітна архітектура;
- трирівнева архітектура.

2.2.1 Монолітна архітектура

Монолітна архітектура (також відома як однокомпонентна архітектура) — це підхід до розробки програмного забезпечення, при якому весь функціонал системи об'єднується в єдиному компоненті чи модулі [18]. Всі частини програми взаємодіють одна з одною без зовнішніх залежностей.

Основні риси однокомпонентної архітектури:

- єдиний модуль чи додаток: Весь функціонал програми розташований в єдиному модулі чи додатку. Всі компоненти та модулі взаємодіють безпосередньо один з одним;
- простота: Монолітні системи можуть бути простими у розумінні та розробці, оскільки весь код знаходиться в одному місці;
- швидка розробка: Розробка, тестування та впровадження може бути прискорене через концентрацію всіх функцій у єдиному модулі;
- легкість утримання: Утримання однокомпонентної системи може бути простішим, оскільки не потрібно враховувати взаємодію між різними компонентами;
- обмежена гнучкість: Внесення змін або розширення системи може бути складним, оскільки всі елементи пов'язані між собою;
- одноразовий випуск: Зазвичай, оновлення або зміни вносяться в одну цілісну систему;
- одна база даних: Часто монолітні додатки використовують одну базу даних для всіх їх компонентів;
- високий рівень взаємозалежності: Компоненти в однокомпонентній системі можуть сильно взаємодіяти один з одним.

Хоча однокомпонентна архітектура може бути простою для початкового розвитку, вона може стати обмеженням для масштабування та розширення великих проектів. Багатокомпонентні архітектури, такі як мікросервісна архітектура, стали популярнішими для розробки великих та складних систем.

2.2.2 Трирівнева архітектура

Трирівнева архітектура (Three-Tier Architecture) — це структурна модель для побудови і розвитку програмного забезпечення, яка розділяє систему на три основні рівні (див. рис. 2.3). Кожен з цих рівнів виконує певні функції та взаємодіє з іншими рівнями за допомогою чітко визначених інтерфейсів. Така архітектурна модель використовується для створення гнучких, розширюваних та легко супроводжуваних систем. [19]

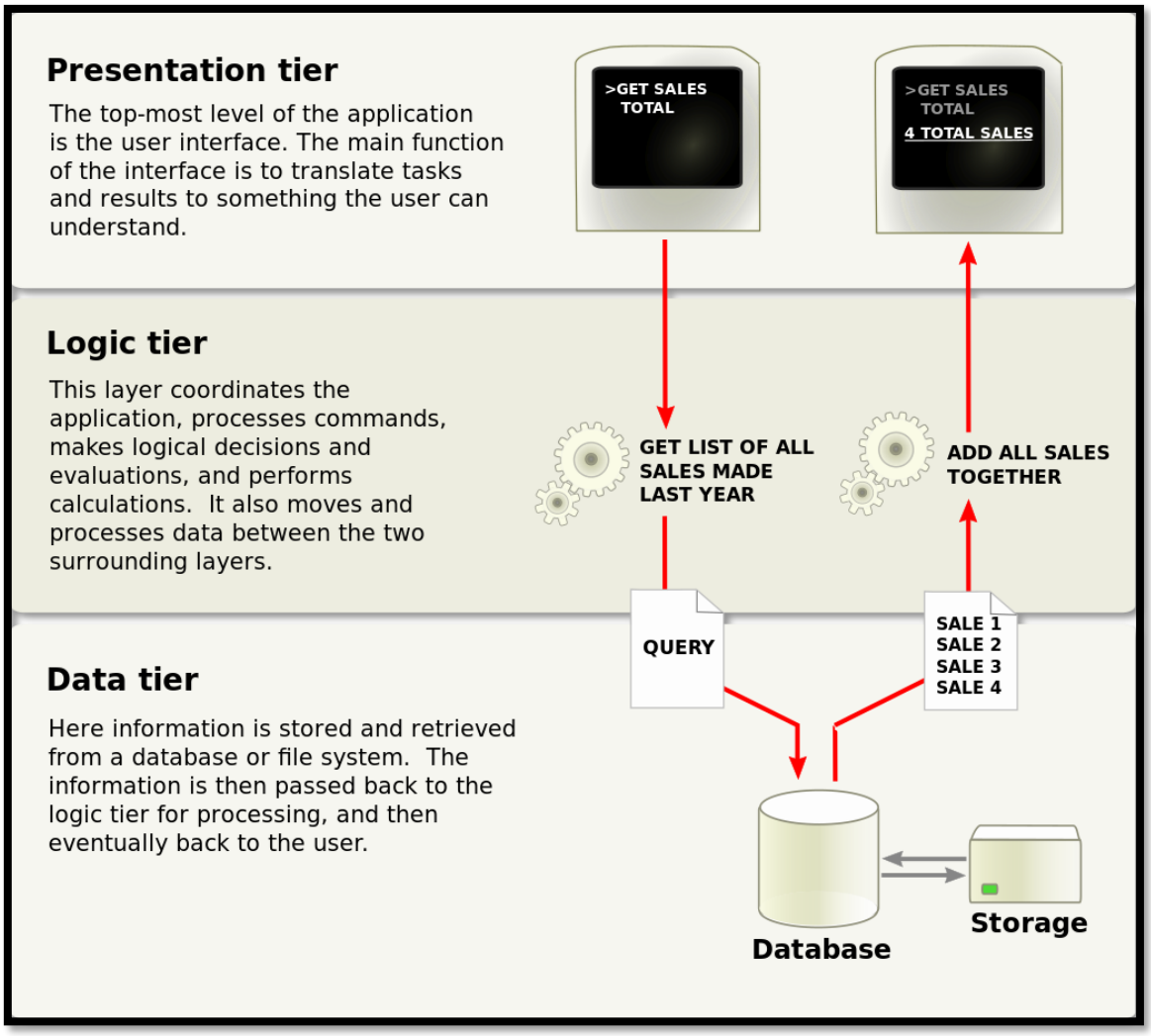

Рисунок 2.3 – Рівні трирівневої архітектури (за даними [20])

Основні рівні трирівневої архітектури:

- представлення (Presentation Tier): Цей рівень відповідає за взаємодію з користувачем і відображення інформації. Тут знаходиться користувацький інтерфейс, і все, що стосується візуального представлення даних. Це може бути веб-інтерфейс, мобільний додаток чи інші способи взаємодії з користувачем;
- бізнес-логіка (Business Logic Tier): На цьому рівні зосереджена бізнес-логіка, яка визначає логіку обробки даних, бізнес-правила та операції. Тут приймаються рішення, пов'язані з обробкою інформації та бізнес-процесами;
- доступ до даних (Data Tier): Рівень доступу до даних відповідає за зберігання та обробку даних. Це може бути база даних або інші джерела даних. Рівень забезпечує взаємодію з базою даних, зберігає та отримує інформацію.

Основні переваги трирівневої архітектури включають модульність (кожен рівень може розвиватися та масштабуватися незалежно), полегшене управління, більшу гнучкість та зручність у супроводженні. Також ця архітектура дозволяє відокремити відповідальності різних компонентів системи.

У контексті створення Telegram-бота для сповіщень про зміни у таблицях пропозицій FunPay трирівнева архітектура може бути організована наступним чином:

- представлення: Користувацький інтерфейс, через який користувач отримує сповіщення та може взаємодіяти з системою. У цьому випадку, Telegram-бот виступає в якості представлення;
- бізнес-логіка: В цьому компоненті реалізована логіка обробки змін у таблицях пропозицій FunPay та відправка сповіщень. Він визначає, які зміни є суттєвими для сповіщення певного користувача;
- доступ до даних: Тут знаходиться вся необхідна інформація про пропозиції, зміни яких слід відслідковувати та налаштування. Серверний компонент взаємодіє з базою даних для виявлення змін.

Ця архітектура дозволяє відокремити логіку користувацького інтерфейсу від бізнес-логіки та обробки даних. Кожен рівень може розвиватися і супроводжуватися незалежно, що полегшує розширення та підтримку системи.

2.3 Взаємодія між компонентами

Розробка та взаємодія програмних продуктів вимагають обрання правильного протоколу комунікації для забезпечення ефективного обміну даними між компонентами систем. Розглянемо та порівняємо три різні підходи:

- REST (Representational State Transfer) [21];
- GraphQL [22];
- gRPC [23]. (див. табл. 2.1)

Таблиця 2.1. Порівняння REST, GraphQL, gRPC (таблиця виконана самостійно)

	REST	GraphQL	gRPC
Протокол	Використовує HTTP/HTTPS для передачі даних	Також використовує HTTP/HTTPS, але один URL для всіх запитів та використовує POST-запити	Використовує HTTP/2 для передачі бінарних даних
Методи	Заснований на стандартних HTTP-методах (GET, POST, PUT, DELETE)	Використовує лише POST-запити, всі операції обробляються одним URL	Використовує Remote Procedure Call (RPC)
Формати даних	Зазвичай використовує JSON або XML	Використовує JSON	Використовує Protocol Buffers (ProtoBuf) для серіалізації даних

Кінець таблиці 2.1

	REST	GraphQL	gRPC
Архітектура	Орієнтована на ресурси, кожен ресурс має свій унікальний URI	Заснований на типах даних та їх відносинах, клієнт визначає необхідні дані	Заснований на визначенні сервісів та методів, використовує схему для опису інтерфейсів
Гнучкість запитів	Зазвичай фіксовані точки доступу, клієнт отримує фіксований набір даних	Гнучка модель запитів, клієнт визначає, які дані йому потрібні	Визначає жорсткий інтерфейс сервісу
Швидкість	Зазвичай повільніший через великий обсяг текстових даних	Залежить від того, як клієнт формулює свої запити, але може бути більш ефективним в порівнянні з REST	Швидший завдяки використанню бінарних даних та HTTP/2
Складність конфігурації	Зазвичай простий у використанні і розумінні	Зазвичай складніший через гнучкість запитів	Може бути складніший через визначення сервісів та методів, але надає сильну типізацію та автоматичну документацію

REST використовує стандартні HTTP-методи та протоколи, що робить його простим для використання та розуміння. Велика кількість розробників знає та розуміє REST, що полегшує співпрацю та обмін знаннями в команді.

REST використовує текстові формати даних, такі як JSON або XML, які легко читати та відлажувати. Це особливо важливо для відладки та розробки.

REST є широко підтримуваним та інтегрованим у багато систем. Це дозволяє легко взаємодіяти з іншими сервісами та інфраструктурою.

Тож, з огляду на вищезазначену таблицю, REST найкращий для даної задачі.

2.4 Визначення вимог до програмного продукту

Загальні вимоги до програмного продукту, який включає в себе Telegram-бот для сповіщень про зміни у таблицях пропозицій FunPay:

- простий та зрозумілий інтерфейс Telegram-бота для комфортної взаємодії користувачів;
- можливість отримання сповіщень користувачами про зміни у таблицях пропозицій FunPay;
- можливість налаштовувати типи та обсяги отримуваних сповіщень;
- персональні налаштування для кожного користувача;
- швидка обробка та надсилання сповіщень про зміни для забезпечення оперативності та актуальності інформації;
- захист особистої інформації користувачів;
- забезпечення безпеки передачі даних та взаємодії з Telegram;
- здатність системи ефективно працювати при збільшенні обсягів даних та користувачів;
- можливість взаємодії з таблицями пропозицій на FunPay для виявлення та відстеження змін;
- система логування подій та можливість створення звітів для аналізу роботи системи;
- забезпечення стабільної та безперебійної роботи сервісу;

- доступна та зрозуміла документація до продукту;
- підтримка та можливість вирішення технічних проблем.

Функціональні вимоги до Telegram-бота для сповіщень про зміни у таблицях пропозицій FunPay:

- реєстрація та авторизація: можливість користувачів реєструватися в боті та авторизовуватися для отримання персоналізованих сповіщень;
- підписка на категорії: можливість вибору категорій товарів або ігор, за якими користувач хоче отримувати сповіщення;
- фільтрація за характеристиками: можливість вказати конкретні характеристики товарів, за якими користувач хоче отримувати сповіщення (ціна, рейтинг, кількість тощо);
- повідомлення про зміни в режимі реального часу: забезпечення миттєвого сповіщення користувачів про будь-які зміни в таблицях пропозицій FunPay;
- перегляд деталей пропозицій: можливість отримання детальної інформації про конкретні пропозиції;
- відписка та пауза: можливість користувачів відписатися від отримання сповіщень або встановити паузу на певний період;
- система повідомлень про нововведення: можливість повідомляти користувачів про нові можливості, зміни в правилах або оновлення функціоналу;
- локалізація: підтримка різних мов для більшого комфорту користувачів.

2.5 Інтерфейс Telegram-бота

GetUpdates та setWebhook - це два різні методи в Telegram API [24], які використовуються для отримання оновлень (повідомлень) для ботів Telegram. (див. рис. 2.4)

Рисунок 2.4 – getUpdates та setWebhook (за даними [25])

Метод getUpdates використовується для запиту оновлень (повідомлень) через додатковий запит на сервер Telegram. Коли ви викликаєте цей метод, сервер Telegram негайно повертає вам список доступних оновлень. Ви можете використовувати цей метод, якщо не хочете встановлювати власний вебхук, але вам потрібно регулярно запитувати сервер Telegram для отримання нових повідомлень.

Метод setWebhook використовується для встановлення вебхука для бота. Вебхук - це механізм, за допомогою якого сервер Telegram автоматично відправляє нові повідомлення на певний URL, який ви вказали. Ви вказуєте URL, на який Telegram буде відправляти POST-запити з новими оновленнями. Встановлення вебхука дозволяє боту отримувати оновлення миттєво, без необхідності регулярних запитів.

Переваги getUpdates:

- Метод getUpdates є досить простим для реалізації, оскільки вам просто потрібно здійснювати GET-запити на сервер Telegram.
- Гнучкість: Ви можете запитувати оновлення в будь-який момент, контролюючи частоту запитів.

Недоліки getUpdates:

- Низька ефективність: Регулярні запити можуть призводити до зайвого трафіку і використання ресурсів сервера, особливо при великій кількості ботів.
- Затримка у взятті оновлень: Оновлення стають доступними для бота тільки в момент запиту, тобто може бути деяка затримка між появою оновлення та його отриманням ботом.

Переваги setWebhook:

- Миттєва відповідь: Завдяки вебхуку бот миттєво отримує оновлення, що дозволяє отримувати повідомлення миттєво після їхнього надходження.
- Ефективність ресурсів: Вебхук дозволяє уникнути зайвих запитів і, таким чином, економити ресурси сервера.

Недоліки setWebhook:

- Потреба в HTTPS: Для використання вебхуку потрібен безпечний (HTTPS) сервер.
- Складніше налаштування: Встановлення вебхуку може вимагати налаштувань та деякої інфраструктури.

Відповідно, якщо більша частота отримання оновлень не є критичною, і простота реалізації є важливою, то getUpdates може бути зручним. Якщо ж потрібні миттєві повідомлення та ефективність ресурсів важлива, вебхук - кращий варіант.

3 РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

У цьому розділі буде розглянуто процес розробки програмного продукту і виявлено проблеми і методи їх рішення.

3.1 Обрані технології розробки

3.1.1 Аналіз платформи для розробки

При розробці програмного продукту, що включає парсинг сайту FunPay та взаємодію з користувачами через Telegram, важливо обрати мову програмування, яка оптимально задовольнить вимоги проекту. Вибір мови може суттєво вплинути на швидкість розробки, легкість масштабування, здатність до інтеграції з іншими системами, а також на зручність подальшого утримання програмного продукту. З урахуванням цих факторів, розглядаємо чотири популярні мови програмування — Python, JavaScript (Node.js), Java та Ruby — щоб оцінити їхню придатність для заданих завдань.

Кожна з цих мов має свої особливості, які роблять її ідеальною для певних типів проектів. Python відомий своєю гнучкістю та широким спектром використання від веб-розробки до машинного навчання [26], тоді як JavaScript є основою сучасної веб-розробки, особливо у комбінації з Node.js для серверної частини[27]. Java залишається стандартом в корпоративній розробці завдяки своїй високій продуктивності та надійності [28], а Ruby пропонує швидкість та простоту розробки з використанням фреймворку Rails[29].

Залежно від специфіки проекту, важливо враховувати такі аспекти як швидкість виконання, доступність відповідних бібліотек та фреймворків, активність та підтримка спільноти, а також досвід команди з певною мовою. У таблиці нижче наведено детальний порівняльний аналіз цих мов, який допоможе зробити обґрунтований вибір (див. табл. 3.1).

Таблиця 3.1. Порівняння мов програмування (таблиця виконана самостійно)

	Python	JavaScript (Node.js)	Java	Ruby
Синтаксис	Простий та читабельний.	Асинхронний, підтримка сучасних стандартів.	Вербозний, строгий.	Виразний та гнучкий.
Виконання	Інтерпретована, підходить для скриптів та автоматизації.	Інтерпретована, асинхронна обробка дозволяє ефективно обробляти I/O.	Компільована, висока продуктивність, ідеальна для великих систем.	Інтерпретована, підходить для швидкої розробки веб-застосунків.
Бібліотеки для API	Requests, BeautifulSoup, Scrapy.	Axios, Express, Puppeteer для парсингу динамічних сайтів.	HttpClient, Jsoup, Spring для розробки великих корпоративних застосунків.	HTTParty, Nokogiri для веб-скрапінгу.
Бібліотеки для ботів	python-telegram-bot, Telepot, aiogram	node-telegram-bot-api, Telegraf.	Java Telegram Bot API.	telegram-bot-ruby.
Підтримка спільноти	Величезна, активна у розвитку бібліотек і інструментів.	Велика, особливо в розробці веб-додатків.	Дуже велика, зокрема в корпоративному секторі.	Менша, але дуже віддана і активна.
Підтримка парсингу	Відмінна, зокрема для статичних сайтів.	Добра, особливо для динамічних сайтів за допомогою Puppeteer.	Добра, включно з розробкою стабільних веб-сервісів.	Добра, зокрема для швидкого скрапінгу.

Кінець таблиці 3.1

	Python	JavaScript (Node.js)	Java	Ruby
Масштабованість	Добра для веб-застосунків, можливі обмеження для великих дистриб'юторів.	Відмінна, асинхронність допомагає у масштабованих застосунках.	Відмінна, підходить для складних систем з високим навантаженням.	Добра для веб-застосунків, легке масштабування.
Легкість розробки	Висока, широкий вибір готових рішень.	Висока, особливо при використанні JavaScript на фронтенді і бекенді.	Середня, потрібен час на налаштування середовища.	Висока, багато вбудованих функцій для швидкої розробки.
Продуктивність	Достатня для більшості веб-застосунків, може бути нижчою за компільовані мови.	Достатня, оптимізована для I/O операцій.	Висока, підходить для великих корпоративних систем.	Достатня, але може вимагати оптимізації для великих об'ємів даних.
Дебагінг і відладка	Зручні інструменти і детальні помилки.	Підтримка сучасних інструментів, NPM.	Висока завдяки потужним IDE, як IntelliJ IDEA.	Інтуїтивні помилки, хороша підтримка від спільноти.

Підсумовуючи порівняльний аналіз мов програмування для проекту з парсингом сайту FunPay та взаємодією з користувачами через Telegram, можна з впевненістю сказати, що Python виділяється серед інших мов своїми унікальними перевагами. Через свою універсальність, зручність використання та широку підтримку бібліотек, Python є відмінним вибором для даного проекту. Велика кількість наявних бібліотек для парсингу веб-сайтів та інтеграції з Telegram,

дозволяє швидко і ефективно розвивати функціонал, необхідний для успішної реалізації проекту.

Python також підтримується однією з найбільших розробницьких спільнот, що забезпечує доступ до великої кількості ресурсів для навчання і вирішення можливих проблем. Це робить Python ідеальною мовою для команд розробників різного рівня кваліфікації, дозволяючи з легкістю знаходити вирішення навіть для складних задач. Його простий та читабельний синтаксис сприяє швидкому освоєнню і зменшує час, необхідний на розробку і дебагінг програмного забезпечення.

Враховуючи ці аспекти, Python є практичним та гарним вибором для цього проекту.

3.1.2 Аналіз СУБД

Для проекту, який включає розробку Telegram-бота для допомоги покупцям у пошуку ігрових цінностей на FunPay, вибір відповідної СУБД є критичним. Розглянемо основні опції, які можуть підійти для такого типу завдань.

MySQL [30] відома своєю стабільністю та високою надійністю, з відмінною підтримкою складних транзакцій з ACID-сумісністю, але вона стикається з труднощами при горизонтальному масштабуванні та управлінні великими обсягами невпорядкованих даних.

PostgreSQL [31] пропонує розширену підтримку SQL стандартів і здатна зберігати різні типи даних, включаючи JSON та геопросторові дані, з високою продуктивністю і надійністю, але її налаштування може бути складнішим та вимагати більше ресурсів для оптимізації продуктивності.

MongoDB [32], як документо-орієнтована NoSQL база даних, відзначається своєю гнучкістю у зберіганні даних без суворої схеми та відмінними можливостями для горизонтального масштабування, що робить її ідеальною для роботи з великими обсягами невпорядкованих даних. Водночас, вона може мати обмеження в транзакційній надійності, що важливо враховувати при розробці систем.

Redis, хоч і використовується зазвичай як допоміжна система для кешування або зберігання тимчасових даних, пропонує екстремально високу швидкість читання та запису та підтримку різних типів даних [33], що може значно прискорити відгук бота, особливо при високому навантаженні.

Кожна з цих СУБД може знайти своє застосування в залежності від специфічних потреб проекту. MongoDB виглядає як найбільш відповідний варіант для управління даними, що динамічно змінюються та не мають строгої структури, що часто зустрічається при парсингу веб-сайтів. PostgreSQL може слугувати як надійна основа для зберігання транзакційних даних та складних запитів, а Redis може оптимізувати швидкість взаємодії з користувачем завдяки своїй здатності до швидкого кешування.

3.1.3 Telegram Bot API

Telegram Bot API [34] надає потужний інструментарій для розробників, дозволяючи створювати ботів, здатних взаємодіяти з користувачами у багатьох різних форматах і контекстах. Ці боти можуть виконувати широкий спектр функцій, від автоматичних відповідей до складних інтеграцій з базами даних і зовнішніми API, що перетворює їх на високоефективні інструменти для бізнесу, освіти, інформаційних сервісів та багато іншого.

До особливостей та можливостей Telegram Bot API можна віднести:

- відправлення повідомлень: Telegram боти можуть надсилати текст, стікери, відео, аудіо та зображення. Це дозволяє розробникам створювати різноманітні взаємодії, залежно від потреб користувачів;
- інтерактивні клавіатури: Боти можуть включати кастомізовані клавіатури для збору відповідей від користувачів, що значно підвищує їх взаємодію та залученість;
- обробка команд: Розробники можуть налаштувати ботів на реагування на специфічні команди, що робить навігацію та функціональність бота більш інтуїтивно зрозумілими;

- вебхуки та лонг-полінг: Для отримання оновлень від Telegram боти можуть використовувати вебхуки, які дозволяють серверам Telegram відправляти оновлення безпосередньо боту, або лонг-полінг, де бот регулярно запитує Telegram про нові оновлення;
- інтеграція з зовнішніми сервісами: Боти можуть інтегруватися з зовнішніми веб-сервісами через API, що дозволяє збирати дані, виконувати дії або надавати інформацію в реальному часі;
- платіжні функції: Telegram дозволяє інтегрувати платіжні системи, що відкриває можливість для електронної комерції прямо у чатах.

Telegram Bot API взаємодіє через прості HTTP-запити і підтримує всі мови програмування, які можуть робити ці запити. Така універсальність дозволяє розробникам використовувати різні програмні стеки та підходи при створенні ботів. API надзвичайно добре документовано, з чіткими інструкціями та великою кількістю прикладів коду, що робить процес розробки доступнішим і зрозумілішим.

Telegram Bot API є одним з найбільш потужних інструментів для створення ботів на сьогодні, надаючи розробникам гнучкість, широкий функціонал і можливість до інтеграції з майже будь-якими зовнішніми сервісами або системами. Це робить його ідеальним вибором для розробки ботів, які можуть виконувати все, від простих завдань до складних бізнес-операцій.

3.1.4 Аналіз бібліотек для взаємодії з Telegram

PyTelegramBotAPI [35] і aiogram [36] — це дві популярні бібліотеки для створення Telegram ботів на Python, кожна з яких має свої унікальні особливості та підходи, які роблять їх придатними для різних сценаріїв використання.

PyTelegramBotAPI пропонує простий і інтуїтивно зрозумілий інтерфейс для взаємодії з Telegram API. Ця бібліотека виконує операції синхронно, що означає послідовну обробку запитів. Цього може бути достатньо для ботів з невеликим навантаженням або де не критична висока швидкість реакції. Основні переваги PyTelegramBotAPI полягають в її легкості в освоєнні та наявності численних

прикладів і документації, що робить її відмінним вибором для новачків або проектів з менш складною логікою. Проте, синхронна природа бібліотеки може стати обмеженням у сценаріях з високою одночасністю або коли потрібна швидка обробка великої кількості повідомлень.

Aiogram, з іншого боку, є асинхронною бібліотекою, що забезпечує високу продуктивність і здатність до одночасної обробки багатьох запитів без блокування виконання програми. Це досягається завдяки використанню асинхронних функцій Python і asyncio. Aiogram підтримує всі функції Telegram API і надає розробникам гнучкі інструменти для створення складних взаємодій у межах бота, включаючи станові машини для управління різними стадіями діалогу з користувачем. Хоча aiogram вимагає більш глибокого розуміння асинхронного програмування, її використання може значно підвищити ефективність ботів, що опрацьовують численні запити або взаємодіють з великою кількістю користувачів одночасно.

Тож, для сценаріїв інтенсивної обробки даних або високонавантажених ботів, які потребують оптимальної продуктивності та асинхронної обробки, aiogram буде кращим вибором, що надає необхідні технічні переваги для ефективної реалізації проекту.

3.1.5 Середовище розробки

PyCharm від JetBrains є одним з найпопулярніших інтегрованих середовищ розробки (IDE) для Python, забезпечуючи розробників комплексним набором інструментів для оптимізації робочого процесу. Це середовище допомагає збільшувати продуктивність за рахунок автоматизації багатьох рутинних завдань, пропонує інтуїтивний інтерфейс та широкі можливості для професійної роботи з кодом. [37]

PyCharm інтенсивно інтегрується з Python, що дозволяє автоматично виявляти помилки, підтримувати рефакторинг, а також забезпечувати автоматичне завершення коду. Окрім Python, PyCharm підтримує ряд інших мов, таких як HTML, JavaScript і CSS, що робить його ідеальним для розробки веб-додатків.

Інструменти для навігації та пошуку дозволяють швидко пересуватися по проекту і знаходити потрібні файли, класи, методи чи символи.

PyCharm включає в себе інструменти для роботи з базами даних і системами контролю версій, інтегрує термінали, а також надає засоби для налагодження і профілювання. Вбудована підтримка веб-фреймворків таких як Django, Flask і Pyramid, а також інструменти для розробки з використанням JavaScript, Node.js та інших технологій, робить PyCharm особливо зручним для розробників веб-додатків.

Однією з ключових особливостей PyCharm є його підтримка віртуальних середовищ, що забезпечує ізоляцію Python-залежностей для кожного проекту. Також PyCharm має глибоку інтеграцію з фреймворками для тестування, такими як pytest, unittest та Doctest, що дозволяє розробникам виконувати тестування безпосередньо з IDE.

PyCharm пропонує дві версії: Professional і Community. Professional версія, що є платною, включає розширені можливості для веб-розробки та роботи з базами даних. Community версія є безкоштовною і надає достатньо функцій для здійснення багатьох стандартних завдань розробки на Python. Обидві версії підтримують налаштування та розширення функціональності через велику кількість доступних плагінів, що дозволяє користувачам адаптувати IDE під свої унікальні потреби.

Завдяки своїй багатофункціональності, зручності використання та широкій підтримці спільноти, PyCharm є відмінним вибором для розробників, які шукають комплексне рішення для своїх Python проектів.

3.2 Серверна частина програмного продукту

Для розробки серверної частини було використано мову Python з використанням бібліотек aiogram, requests та BeautifulSoup4.

3.2.1 Створення бота у Telegram

Створення чат-бота в Telegram через BotFather є фундаментальним процесом, який включає кілька критичних етапів: реєстрацію, конфігурацію та розгортання.

Цей процес демонструє використання комплексних взаємодій між розробником і Telegram API, що забезпечують гнучкість і розширюваність ботів.

Перший крок полягає в ініціації діалогу з @BotFather — офіційним ботом Telegram, який використовується для створення та управління іншими ботами. Розробник починає з надсилання команди /newbot BotFather, який запитує ім'я та унікальний username бота (див. рис. 3.1). Це ім'я має бути ідентифікатором, який чітко асоціюється з функціональністю бота і закінчується на "bot", що є загальноприйнятною конвенцією. Після вибору імені BotFather генерує унікальний токен доступу. Цей токен є ключем для програмного інтерфейсу API, який забезпечує автентифікацію та авторизацію запитів від бота до Telegram.

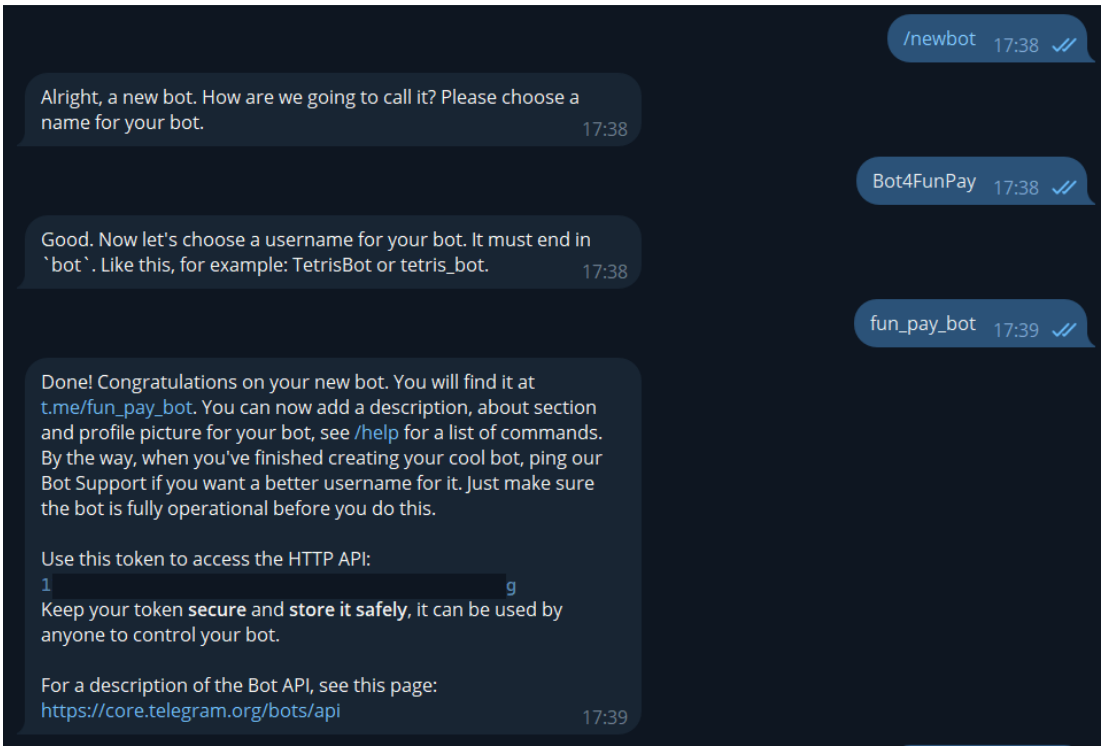

Рисунок 3.1 – Діалог з BotFather (рисунок створено самостійно)

Після створення бота розробник може скористатися додатковими командами для налаштування. Наприклад, команди /setdescription, /setcommands і /setuserpic дозволяють встановити опис бота, визначити список команд, які бот може розпізнати, і завантажити зображення профілю відповідно. Ці параметри покращують інтерактивність бота та сприяють його ідентифікації користувачами.

Після конфігурації розробник використовує токен для програмування бота за допомогою обраної бібліотеки, наприклад python-telegram-bot або aiogram для Python. Ці бібліотеки спрощують реалізацію таких функцій, як обробка повідомлень, керування файлами та інтеграція з зовнішніми API. Розробник пише код, який дозволяє боту реагувати на вхідні команди та ініціювати дії відповідно до логіки, заданої в алгоритмах бота.

Фінальний крок полягає в розгортанні бота на сервері або у хмарі. Це забезпечує боту стабільність і доступність, необхідні для обслуговування кінцевих користувачів. Сервер приймає запити, обробляє їх через бота і відправляє відповіді користувачам.

Реєстрація, конфігурація та розгортання чат-бота в Telegram є інтегрованим процесом, який вимагає від розробників знань як у програмуванні, так і в адмініструванні серверів. Цей процес демонструє потужність та гнучкість Telegram як платформи для розробки чат-ботів.

3.2.2 Проєктування структури зберігання даних

ER-діаграму бази даних представлено у додатку Д цієї роботи. База даних представлена 12-ма таблицями:

- Bot_users – усі зареєстровані користувачі бота;
- New_lot_settings – налаштування для створення нових лотів;
- Notifications – всі налаштування сповіщень;
- Games – ігри, що представлені на біржі;
- Sections – розділи, пов'язані з іграми;
- Filters – фільтри для пошуку лотів (на біржі);
- Filters_settings – налаштування для фільтрів;
- Filters_values – можливі значення атрибутів (для фільтрів з випадаючим списком);
- Attributes – атрибути лотів;
- Notifications_settings – налаштування для сповіщень;
- Changes_settings – налаштування сповіщень для змін в лотах;

- Lots – лоти в системі.

База даних має наступні відношення між таблицями (позначення 1:M – one to many):

- Bot_users - Notifications: 1:M. Один користувач може мати декілька налаштувань для сповіщень;
- Games - Sections: 1:M. Одна гра може мати декілька розділів;
- Sections - Lots: 1:M. Один розділ може містити декілька лотів;
- Filters - Filters_values: 1:M. Один фільтр може мати декілька можливих значень;
- Filters - Notifications_settings: 1:M. Один фільтр може мати декілька налаштувань відносно фільтрів;
- Attributes - Filters_values: 1:1. Одному атрибуту відповідає одне значення фільтру;
- Notifications - Notifications_settings: 1:M. Одне налаштування для сповіщень може мати декілька налаштувань відносно фільтрів;
- Notifications – New_lot_settings: 1:1. Одному налаштуванню для сповіщень відповідає одне налаштування для нових лотів;
- Notifications - Change_settings: 1:M. Одне налаштування для сповіщень може мати декілька налаштувань про зміни.

3.2.3 Реалізація клієнтської частини чат-бота

Клієнтська частина має надавати повний доступ до такого фунцкіоналу:

- повне створення заготовок для пошуку (вибір гри та розділу, вибір фільтрів, зміна короткого та детального опису);
- одноразовий пошук за заготовкою та вивантаженням усіх лотів у excel-файл;
- можливість отримувати сповіщення про зміни у таблицях пропозицій та їх тонкого налаштування;
- можливість дізнаватися про нові розділи та ігри на сайті.

Взаємодія користувачів з чат-ботом у Telegram може відбуватися через різноманітні механізми, які розробник може використовувати для створення зручного та інтуїтивно зрозумілого інтерфейсу.

До основних способів, через які користувачі можуть взаємодіяти з ботами можна віднести:

- команди. Команди — це, можливо, найпростіший і найбільш використовуваний метод взаємодії з ботами. Вони є текстовими стрічками, що починаються зі слеша (наприклад, /start або /help). Розробники можуть налаштувати бота на реакцію на певні команди, що ініціюють певні функції або дії;
- інлайн-режим. Інлайн-боти дозволяють користувачам викликати бота прямо у будь-якому чаті, ввівши його ім'я та запит в інлайн-режимі. Наприклад, користувач може написати “@gif happy dog” у не пов’язаному з ботом чаті, щоб отримати GIF-зображення з веселою собакою для відправки в чат. Цей метод забезпечує велику зручність та швидкість взаємодії;
- звичайні клавіатури. Звичайні клавіатури замінюють стандартну клавіатуру користувача на набір кнопок, які відправляють текстові повідомлення назад до бота. Це може бути корисним для напрямлення користувача через фіксований набір опцій або збирання специфічних відповідей;
- інлайн-клавіатури. Інлайн-клавіатури дозволяють додавати кнопки безпосередньо під повідомленням бота. Вони не замінюють стандартну клавіатуру користувача і можуть містити кнопки, які виконують дії (наприклад, callback-запити), замість відправлення тексту.

Для нашої задачі було вирішено побудувати взаємодію з користувачем через єдину команду – “/menu”, яка викликає повідомлення з інлайн-клавіатурою, аби не захламляти діалог з ботом непотрібними повідомленнями. Так, наприклад, аби створити нову заготовку для пошуку потрібно:

- надіслати команду “/menu”;
- натиснути “Заготовки для пошуку”, “Створити”;
- обрати назву гри та розділ сайту (див. рис. 3.2).

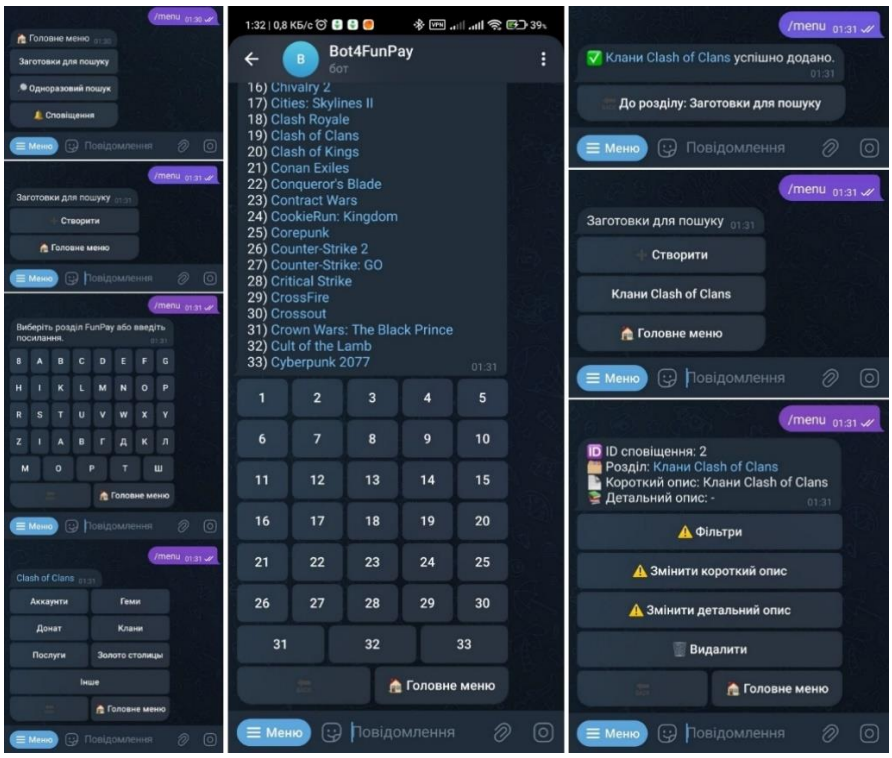

Рисунок 3.2 – Створення заготовки (рисунок створено самостійно)

Останню функцію було вирішено додати шляхом створення окремої Telegram-групи з розділом на теми, щоб не надсилати велику кількість однакових сповіщень різним користувачам (див. рис. 3.3).

Рисунок 3.3 – Сповіщення про зміни розділів (рисунок створено самостійно)

Сповіщення про зміни у таблицях пропозицій приходять у особистих повідомленнях та містять у собі інформацію про тип сповіщення, розділ на сайті, короткий опис, ціну, інформацію про продавця (кількість зірок, час на сайті), кількість товару, посилання на продавця та на лот, теги для більш зручної навігації повідомленнями (див. рис. 3.4).

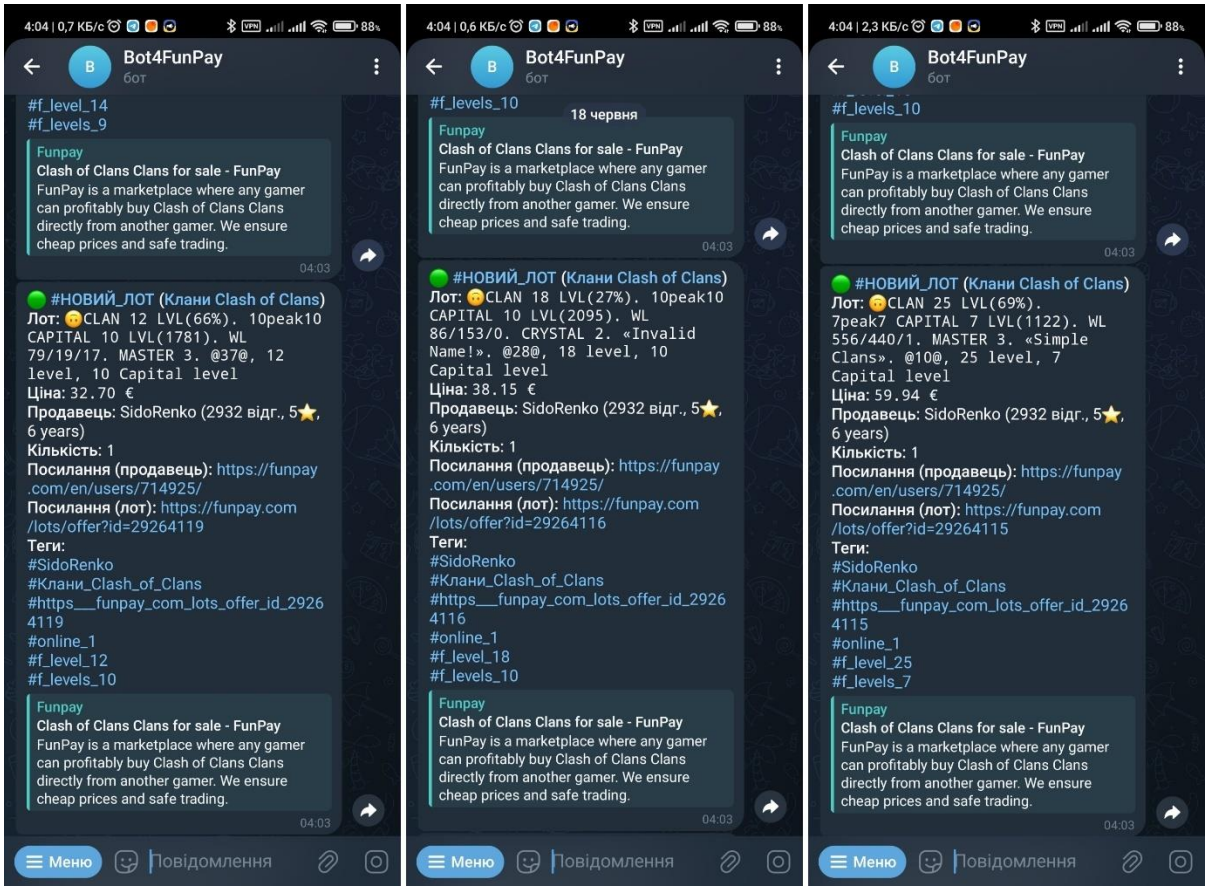

Рисунок 3.4 – Приклади сповіщень (рисунок створено самостійно)

3.2.4 Реалізація серверної частини чат-бота

Серверна частина нашого Telegram-бота виконує комплекс завдань, необхідних для забезпечення його ефективного функціонування. Вона відповідає за прийом і обробку повідомлень від користувачів, аналізуючи їхні запити та визначаючи відповідні дії. Це включає збереження інформації про сесії користувачів, що дозволяє підтримувати безперервність взаємодії і надавати персоналізовані відповіді.

Важливою складовою є взаємодія з базою даних, де зберігаються профілі користувачів, налаштування пошуку і сповіщень, а також історія запитів і

результатів. Сервер підтримує постійний обмін даними з базою, забезпечуючи їх актуальність і доступність.

Також серверна частина інтегрується з зовнішніми сервісами, такими як платформа FunPay, для отримання найсвіжіших даних про ігрові цінності, що включає в себе парсинг інформації про лоти у різних розділах сайту.

Забезпечення безпеки даних є ще однією критичною функцією серверної частини. Вона гарантує захист персональної інформації користувачів і безпечне передавання даних між ботом і сервером. Завдяки цьому, користувачі можуть бути впевнені в конфіденційності своїх даних під час використання Telegram-бота.

ВИСНОВКИ

У ході цього дослідження була проведена детальна аналіз предметної області та вивчення методів прогнозування для створення Telegram-бота, що допомагає покупцям у пошуку ігрових цінностей на платформі FunPay. Основна увага була приділена дослідженню різних методів прогнозування, які можуть значно покращити процес пошуку та придбання ігрових цінностей. Також було розглянуто варіанти архітектурних підходів і інструментів, які оптимально підходять для реалізації такого програмного продукту.

Результати дослідження демонструють, що впровадження Telegram-бота може значно підвищити швидкість та зручність взаємодії користувачів з торговельними платформами. Використання алгоритмів прогнозування та аналізу даних дозволяє боту надавати користувачам актуальну інформацію про найкращі пропозиції в реальному часі, збільшуючи їхні шанси на успішні транзакції. Інтуїтивно зрозумілий інтерфейс, який реалізований через інлайн-клавіатуру та сповіщення про зміни на сайті, роблять процес взаємодії максимально простим та ефективним.

У подальшому планується вдосконалення та адаптація бота під різноманітні ігрові платформи та біржі, що може відкрити нові можливості для його застосування у ширшому контексті ігрової індустрії, стимулюючи технологічний розвиток та інновації. Результати цієї роботи підтверджують важливість подальших досліджень у цій області та вказують на потребу інтеграції сучасних технологічних рішень у повсякденні процеси взаємодії користувачів з цифровими платформами. Отримані знання та досвід можуть бути використані для подальшої оптимізації та розробки нових інструментів, які зроблять віртуальний ігровий світ ще більш доступним та зручним для користувачів по всьому світу.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. O. Mazurova, O. Samantsov, O. Topchii and M. Shirokopetleva, A Study of Optimization Models for Creation of Artificial Intelligence for the Computer Game in the Tower Defense Genre, 2020 IEEE International Conference on Problems of Infocommunications. Science and Technology (PIC S&T), 2020, pp. 491-496, doi: 10.1109/PICST51311.2020.9468057.

2. FunPay — біржа ігрових цінностей. URL: https://funpay.com/uk/ (дата звернення: 29.12.2023).

3. Transactions Protection : FunPay. URL: https://funpay.freshdesk.com/en/support/solutions/articles/103000275205-transactions-protection (дата звернення: 29.12.2023).

4. Правила / FunPay. URL: https://funpay.com/uk/trade/info (дата звернення: 29.12.2023).

5. Digital 2023 October Global Statshot Report — DataReportal – Global Digital Insights. URL: https://datareportal.com/reports/digital-2023-october-global-statshot (дата звернення: 29.12.2023).

6. Telegram: Contact @durov. URL: https://t.me/durov/215 (дата звернення: 29.12.2023).

7. Telegram global MAU 2022 | Statista. URL: https://www.statista.com/statistics/234038/telegram-messenger-mau-users/ (дата звернення: 29.12.2023).

8. Telegram Users Statistics, Data & Facts (2024 Updated). URL: https://www.grabon.in/indulge/tech/telegram-users-statistics/ (дата звернення: 10.06.2024).

9. sidor0912/FunPayCardinal. URL: https://github.com/sidor0912/FunPayCardinal (дата звернення: 29.12.2023).

10. NightStrang6r/FunPayVertex. URL: https://github.com/NightStrang6r/FunPayVertex (дата звернення: 29.12.2023).

11. NightStrang6r/FunPayServer. URL: https://github.com/NightStrang6r/FunPayServer (дата звернення: 29.12.2023).

12. Soft FunPay Manager - auto-withdrawal, bot for FunPay. URL: https://ggsel.net/en/catalog/product/3671647 (дата звернення: 29.12.2023).

13. All_Parsers/funpay at 2c545d8dc8f71ae3db53c75df570ddfc5c1a4956 JustLike420/All_Parsers. URL: https://github.com/JustLike420/All_Parsers/tree/2c545d8dc8f71ae3db53c75df570ddfc5c1a4956/funpay (дата звернення: 29.12.2023).

14. chyornyy/funpay-parser: The Funpay Parser is a web scraper built with Scrapy that extracts information about lots on the https://funpay.com/ marketplace. URL: https://github.com/chyornyy/funpay-parser/tree/main (дата звернення: 29.12.2023).

15. Лекція 3 3UML. Діаграма прецедентів. URL: https://e-tk.lntu.edu.ua/mod/resource/view.php?id=9292 (дата звернення: 30.12.2023).

16. Інструкція, як будувати UML-діаграми | DOU. URL: https://dou.ua/forums/topic/40575/ (дата звернення: 30.12.2023).

17. Component Based Architecture. URL: https://medium.com/clean-code-channel/component-based-architecture-2cb38ea1b247 (дата звернення: 30.12.2023).

18. Монолітна архітектура ПЗ. - QALight. URL: https://qalight.ua/baza-znaniy/shho-take-monolitna-arhitektura/ (дата звернення: 30.12.2023).

19. Лекція 20. Архітектура та проектування комонентних систем (2016 р.). URL: https://victana.lviv.ua/knyhy/konspekty-lektsii/133-kros-platformenne-prohramuvannia-ta-khmarni-servisy/568-lektsiia-20-arkhitektura-ta-proektuvannia-komonentnykh-system-2016-r (дата звернення: 30.12.2023).

20. Бізнес-логіка – Вікіпедія. URL: https://uk.wikipedia.org/wiki/Бізнес-логіка (дата звернення: 18.06.2024).

21. Як правильно працювати з REST API - Блог ITVDN. URL: https://itvdn.com/ua/blog/article/rest-api-18 (дата звернення: 30.12.2023).

22. GraphQL | A query language for your API. URL: https://graphql.org/ (дата звернення: 30.12.2023).

23. gRPC. URL: https://grpc.io/ (дата звернення: 30.12.2023).

24. Telegram Bot API. URL: https://core.telegram.org/bots/api (дата звернення: 30.12.2023).

25. Запускаємо Telegram-бота на Webhook з Docker та ngrok, інтегруємо зовнішні API - YouTube. URL: https://www.youtube.com/watch?v=M8mdtSauOJA (дата звернення: 18.06.2024).

26. Python Courses & Tutorials | Codecademy. URL: https://www.codecademy.com/catalog/language/python (дата звернення: 18.06.2024).

27. Співбесіда з Node.js розробником. 255 запитань для Junior, Middle і Senior | DOU. URL: https://dou.ua/lenta/articles/interview-node-js/ (дата звернення: 18.06.2024).

28. Java Tutorial. URL: https://www.w3schools.com/java/ (дата звернення: 18.06.2024).

29. Ruby on Rails — A web-app framework that includes everything needed to create database-backed web applications according to the Model-View-Controller (MVC) pattern. URL: https://rubyonrails.org/ (дата звернення: 18.06.2024).

30. MySQL. URL: https://www.mysql.com/ (дата звернення: 18.06.2024).

31. PostgreSQL: The world's most advanced open source database. URL: https://www.postgresql.org/ (дата звернення: 18.06.2024).

32. MongoDB: The Developer Data Platform | MongoDB. URL: https://www.mongodb.com/ (дата звернення: 18.06.2024).

33. Shubin, I. , Kozyriev, A. Method for Solving Quantifier Linear Equations for Formation of Optimal Queries to Databases CEUR Workshop Proceedings, 2023, 3396, pp. 449–459

34. Introduction - A guide to Telegram.Bot library. URL: https://telegrambots.github.io/book/ (дата звернення: 18.06.2024).

35. pyTelegramBotAPI 4.19.2 documentation. URL: https://pytba.readthedocs.io/en/latest/ (дата звернення: 18.06.2024).

36. aiogram 3.7.0 documentation. URL: https://docs.aiogram.dev/en/latest/ (дата звернення: 18.06.2024).

37. Pycharm це IDE від JetBrains для програмування на Python. URL: https://foxminded.ua/pycharm-tse/ (дата звернення: 18.06.2024).