МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Державний університет "Житомирська політехніка"
Кафедра інженерії програмного забезпечення

ЗВІТ З ВИРОБНИЧОЇ ПРАКТИКИ

Студента (ки) 3 курсу групи ІПЗ-21-5
Галузь знань 12 «Інформаційні технології»
спеціальність 121 «Інженерія програмного забезпечення»
ступінь «**бакалавр»**
Волківський Я.С.
(прізвище ініціали, підпис)

Керівник практики: старший викладач кафедри ІПЗ
Локтікова Тамара Миколаївна

Кількість балів: _ Національна оцінка: _____ ECTS:__

Члени комісії:

_____ Н.О. Кушнір
(підпис) (прізвище та ініціали)

_____ Т.М. Локтікова
(підпис) (прізвище та ініціали)

_____ В.П. Полторак
(підпис) (прізвище та ініціали)

_____ І.В. Мовчан
(підпис) (прізвище та ініціали)

Житомир – 2024

ЗМІСТ

1. Характеристика бази виробничої практики:

База виробничої практики представлена у вигляді ПриватБанку, який є одним з найбільших комерційних банків в Україні та має значний вплив на фінансову сферу країни.

АТ КБ «Приватбанк» - реєстраційний №92 від 19.03.1992р. у Державному реєстрі банків. Банківська ліцензія НБУ №22 від 05.10.2011р.

Юридична адреса: вул. Грушевського, 1д, м. Київ 01001, Україна

Адреса для кореспонденції: вул. Набережна Перемоги, 30, м. Дніпро, 49094, Україна.

Поштова скринька: пошта клієнтської підтримки help@pb.ua та пошта призначена для офіційних письмових запитів/листів/заяв, що мають містити підпис/електронний підпис та/або печатку cancelyaria@privatbank.ua.

1.1. Опис основної діяльності підприємства:

ПриватБанк надає широкий спектр фінансових послуг для різних категорій клієнтів, включаючи фізичних осіб, підприємства та установи. Основні види діяльності банку включають, але не обмежуються:

- Розрахунково-касове обслуговування
- Кредитування (включаючи іпотечні кредити, автокредити тощо)
- Інвестиційний бізнес та управління активами
- Обмін валют та інші операції на валютному ринку
- Депозитарне обслуговування
- Електронні платежі та інтернет-банкінг
- Страхування та інші фінансові послуги.

1.2. Опис підрозділу проходження практики:

Студенти, які проходять виробничу практику в ПриватБанку, можуть вибирати різні підрозділи відповідно до своїх інтересів та спеціалізації. Це можуть бути відділи з роботи з клієнтами, кредитування, інвестиційний бізнес, робота з корпоративними клієнтами, ризик-менеджмент, технологічні підрозділи та інші. Важливо, щоб студент обрав підрозділ, що відповідає його професійним і кар'єрним цілям.

2. ОГЛЯД І АНАЛІЗ АПАРАТНОГО ЗАБЕЗПЕЧЕННЯ БАЗИ ВИРОБНИЧОЇ ПРАКТИКИ

На підприємстві бази виробничої практики використовуються комп’ютери, але я використовував свої, бо мені так зручніше та маю змогу дистанційної роботи.

Розглянемо основні технічні характеристики 2-х ПК, з яких один це десктоп рішення, а інший це лептоп Спочатку, що стосується системного блоку.

Спочатку розберемо основну машину, котра є десктопним рішенням. Почнемо з його Центрального Процесору представлені в таблиці 2.1

Таблиця 2.1

Характеристика	Тип, значення
Модель	AMD Ryzen 5 3600
Архітектура (кодова назва)	Zen2 (Matisse)
Тип сокета	PGA AM4
Кількість ядер	6
Кількість потоків	12
Базова тактова частота	3,6GHz
Частота в режимі Boost	4,2GHz
Графічний процесор	Відсутній
Кеш L1-L3	384KB, 3MB, 32MB
Техпроцес	TSMC 7nm FinFET
TDP (Теплопакет)	65W

Характеристики материнської плати основного ПК в таблиці 2.2

Таблиця 2.2

Характеристика	Тип, значення
Модель	MSI B450M Pro-VDH MAX
Сокет	AM4
Процесорна сувмісність	AMD Ryzen ™ / Ryzen ™ 1-ї, 5-ї генерацій з графікою Radeon ™ Vega та 2-ї генерації, 5-ї AMD Ryzen ™ з графікою Radeon ™
Слоти розширення	1 x PCIe 3.0 x16, 2 x PCIe 2.0 x1
Чипсет	AMD B450
Оперативна пам’ять	DDR4
Частоти оперативної пам’яті	2667/ 2800/ 2933/ 3000/ 3066/ 3200/ 3466/ 3733/ 3866 MHz (by A-XMP OC MODE
Кількість слотів оперативної пам’яті	4 шт.
Аудіоконтролер	Realtek® ALC892 Codec
Мережевий контролер	Realtek® 8111H Gigabit LAN controller
Зовнішні порти	PS/2, VGA Port, 4xUSB 2.0 Port, 4x USB 3.2 Gen1, 1xLAN Port, 3xHD Audio Connectors, 1xDVI-D Port, 1xHDMI™ Port

Внутрішні з’єднувачі та порти	1 x 24-pin ATX головне живлення 1 x 8-pin ATX 12V живлення ЦП 4 x SATA 6Gb/s 2 x USB 2.0 (додатково 4 USB 2.0 ports) 1 x USB 3.1 Gen1 (додатково 2 USB 3.1 Gen1 ports) 1 x 4-pin живлення вентилятора ЦП 2 x 4-pin живленнях корпусних вентиляторів 1 x Роз’єм передньох звукової 2 x Роз’єм передньої панелі 1 x TPM 1 x Chassis Intrusion connector 1 x Serial port connector 1 x Parallel port connector 1 x RGB LED strip connector 1 x Clear CMOS jumper
Розміри	9.6 in. x 9.6 in. (24.4 cm x 24.4 cm) m-ATX

Характеристика оперативної пам’яті таблиця 2.3

Таблиця 2.3

Характеристика	Тип, значення
Тип пам’яті	DDR4
Форм-фактор	DIMM
Виробник одиниці пам’яті	Micron
Виробник планки пам’яті	Cricical
Серія	Crucial Ballistix
Об’єм (в сумі)	8GB (16GB)
Кількість, шт	2
Частота	3200MHz
Таймінги	CL16-18-18-36
Напруга, в.	1.35

Характеристика накопичувача під ОС таблиця 2.4

Таблиця 2.4

Характеристика	Тип, значення
Тип	SSD M.2
Виробник	A-DATA XPG
Модель	XPG Spectrix S40G
Об’єм	256GB
Форм-фактор	M.2 2280
Інтерфейс	PCIE 3.0 x4
Швидкість читання, МБ/с	3500
Швидкість запису, МБ/с	3000

Характеристика накопичувача під програми таблиця 2.5

Таблиця 2.5

Характеристика	Тип, значення
Тип	SSD
Виробник	Patriot
Модель	Patriot P210
Об’єм	2TB
Форм-фактор	2,5”
Інтерфейс	SATA 3
Швидкість читання, МБ/с	550
Швидкість запису, МБ/с	500

Характиристика накопичувача під документи та інші файли таблиця 2.6

Таблиця 2.6

Характеристика	Тип, значення
Тип	HDD
Виробник	Western Digital
Кількість обератнь, rpm	5400
Об’єм	1TB
Форм-фактор	2.5”
Інтерфейс	SATA 3
Швидкість читання, МБ/с	125
Швидкість запису, МБ/с	120

Характеристика відеокарти таблиця 2.7

Таблиця 2.7

Характеристика	Тип, значення
Виробник	NVIDIA
Вендор	ASUS
Модель	NVIDIA GeForce GTX 1070 OC
Графічний процесор	GTX 1070 (Boost 1911MHz)
Об’єм VRAM	8GB
Тип VRAM	GDDR5 (4007MHz)
Система охолодження	Asus Expedition
Теплопакет (TDP)	150W
Інтерфейс	PCIE 3.0 x16
Конектор живлення, pin	8
Виходи	2xHDMI, 2xDisplayPort, DVI-D
Архітектура графічного процесору	Pascal

Характеристика блока живлення таблиця 2.8

Таблиця 2.8

Характистика	Тип, значеня
Виробник	Chieftec
Модель	CHIEFTEC 600W PROTON (BDF-600S)
Потужність	600W
Сертифікат	80 PLUS BRONZE
Форм-фактор	ATX

Характеристика корпусу ПК таблиця 2.9

Таблиця 2.9

Характеристика	Тип, значення
Модель	1st Player Firerose F5-R1 Color Led
Тип	Miditower
Колір	Чорний
Кількість вентиляторів	3xПередня панель(120мм), 1xЗадня панель(120мм), 1xЗверху(120мм)
Підтримувані мат-плати	ATX, microATX, miniATX
Наявність БП	ні
Розташування БП	знизу
Зовнішні порти	2xUSB2.0, 1xUSB3.0, 1xAudio(навушники), 1xAudio(мікрофон)

Охарактерезуємо тепер пристрої ввода та виведення (таблиці 2.10-2.12)

Клавіатура:

Таблиця 2.10

Характеристика	Тип, значення
Виробник	Dark Project
Модель	Dark Project KD87A
Тип	Механічна
Підсвітка	RGB
Модель перемикачів	Gateron Optic Red 2.0
Модульний кабель	Так, Type-C
Колір	Чорний

Миша:

Таблиця 2.11

Характеристика	Тип, значення
Виробник	A4Tech
Модель	A4Tech Bloody V9M
Тип підключення	Провід
Сенсор	Оптичний
Частота	125 – 1000Гц
Кількість кнопок	9
Пам’ять	256КБ
Інтерфейс	USB 2.0/3.0

Монітор:

Таблиця 2.12

Характеристика	Тип, значення
Виробник	ASUS
Модель	Asus VG249Q
Діагональ	23,8’’
Роздільна здатність	1920x1080
Частота оновлення	144Гц
Час реакція	1ms MPRT / 4ms GtG
Тип матриці	IPS
Кількість кольорів	16.7 млн (6Bit+FRC)
Покриття	матове
Відношення сторін	16:9
Динаміки	2x2W Stereo RMS

Характеристика мережі

Таблиця 2.13

Характеристика	Тип, значення
Роутер	Archer C20
Швидкість мережі	100Мбіт/с
LAN-порти	100Мбіт/с
Тип мережі	2.4/5GHz

Як бачимо, комп’ютер є досить сучасним рішенням і є універсальним рішенням, котре прекрасно підходить до майже всіх задач.

Далі розберемо рішення лептоп, яке являє собою ноутбук від Apple, а саме Apple Macbook Air 13 2015. В таблицях 2.14-2.16 буде описаний кожен його компонент

Процесор:

Таблиця 2.14

Характеристика	Тип, значення
Модель	Intel Core™ i5 5250U
Кількість ядер	2
Кількість потоків	4
Базова тактова частота	1,6ГГц
Частота в Boost	2,7ГГц
Графічний процесор (VRAM)	Intel HD 6000 (1,5GB)
Теплопакет (TDP)	15W
Техпроцес, нм	14
Кодова назва архітектури	Broadwell

Оперативна пам’ять:

Таблиця 2.15

Характеристика	Тип, значення
Об’єм	8GB
Тип каналу роботи	Dual-Chanell
Частота роботи	1600MHz
Тип пам’яті	DDR3

Основний накопичувач:

Таблиця 2.16

Характеристика	Тип, значення
Виробник	SAMSUNG
Об’єм	256GB
Швидкість читання	1500 МБ/с
Швидкість запису	1000МБ/с
Тип (форм-факторо)	SSD (M.2 2280)

Дисплей:

Таблиця 2.17

Характеристика	Тип, значення
Тип	Вбудований
Роздільна здатність	1440x900
Діагональ	13,3”

Клавіатура та трекпад є вбудованими пристроями вводу даних, виготовлені компанією Apple.

Рішення типу лептоп є достатньо застарілим (2015 рік), але прекрасно виконує поставлені йому завдання, також і прекрасно впорався з виконням виробничої практики.

3.1 АНАЛІЗ ЗАДАЧІ

Банківський сектор активно використовує месенджери для надання послуг клієнтам. Метою даного проекту є створення бота для Telegram, який надасть клієнтам ПриватБанку доступ до різноманітних банківських послуг та інформації.

Проблема: Зростаюча популярність банківських послуг через месенджери створює потребу в ефективних інструментах взаємодії з клієнтами.

Мета проекту: Розробка бота для ПриватБанку в Telegram з метою надання клієнтам доступу до банківських послуг та інформації через зручний інтерфейс месенджера.

Поставлені вимоги до бота:

- Функціональні вимоги: перегляд історії транзакцій, перегляд балансу, Підтримка користувачів в реальному часі через бота, інформація про валютні курси та інша корисна інформація.
- Технічні вимоги: швидка відповідь на запити, безпека даних, масштабованість для обслуговування великої кількості користувачів.

Порівняння з іншими рішеннями:

ПриватБанк бот:

Переваги: наявність широкого спектру послуг, доступних через бота, інтеграція з існуючими банківськими системами та послугами.

Недоліки: обмеження функціональності та складність взаємодії з ботом.

Боти інших банків:

Переваги: швидка відповідь на запити користувачів, доступ до актуальної інформації про банківські послуги.

Недоліки: недостатня функціональність порівняно з клієнтським сервісом.

ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Програмне забезпечення (ПЗ) для бота в Telegram для ПриватБанку є складним технічним проектом, який вимагає уваги до деталей на кожному етапі розробки. Цей розділ описує методику проектування та розробки ПЗ для досягнення поставлених цілей і вимог.

Вимоги до проекту були описані в попередньому розділі (Функціональні та Технічні вимоги).

Архітектура визначає структуру системи та взаємозв'язки між її компонентами. У випадку бота в Telegram для ПриватБанку, можна розглянути таку архітектуру:

Мова програмування: розширена версія JavaScript з підтримкою типізації TypeScript.

Клієнт Telegram API: Взаємодія з користувачами через месенджер Telegram та використання бібліотеки Telegraf для мови TypeScript.

Сервіс бота: Обробка та відповідь на запити користувачів.

Серверна частина: Логіка бізнес-процесів, інтеграція зі зовнішніми системами (API ПриватБанку та API Отримання даних про стан валют).

Бази даних: Використання Redis для швидкого збору інформації під час використання бота та MongoDB для зберігання важливої інформації про користувачів, таких як, ім’я, номер телефону, номер картки, баланс картки, історії транзакцій і також для зберігання даних для функції «Лайв підтримка».

3.2 ОПИС РОБОТИ З ПРОГРАМНИМ ПРОДУКТОМ ТА ЙОГО ТЕСТУВАННЯ

Основні функції: перегляд балансу та історії транзакції; перегляд курсу валют та робота з ними; підтримка в реальному часі через бот, котра з'єднує працівника (оператора) та клієнта; поширені питання, котрі взяті з офіційного сайту АТ «ПриватБанк».

Розглянемо кожну функцію більш детально:

Перегляд балансу та історії транзакцій: надає змогу перегляду поточного балансу користувача та 5 останніх транзакцій, в силу політики конфендиціаональності АТ «ПриватБанк» було використана система генерації транзакцій, котра в свою чергу також впливає на баланс картки.

Перегляд курусу валют та робота з ними: надає можливості перегляду курсу 161 валюти, перевести в конкретну валюту, котру потрібно вибрати зі списку та можливість переводити конкретну суму в конкрутну валюту.

Підтримка в реальному часі: надає змогу користувачеві та оператору спілкуватися між собою через телегам чат-бот. Користувач обирає функцію «Лайв підтримка» після чого запит йде всім вільним операторам, після того, як один з операторів прийняв запит, для всіх інших операторів цей запит стає недоступним. Далі користувач з оператором можуть спілкуватися та надсилати від фото до геолокації. Коли питання було вирішено або користувач або оператор мають змогу закрити активний канал за допомогою кнопки «ВІДМІНА» та користуватися далі функціями бота. Після закриття каналу оператор стає доступним до інших запитів. Якщо запит прийнято іншим оператором, то інші не взмозі його прийняти, але повідомлення про запит буде, але кнопка буде недоступна і з повідомленням, що тей запит було прийнято ішним оператором і також коли канал був закритий, кнопка також змінюється на «Канал закритий» і також не доступна.

Поширені питання: надає користувачу змогу перегляду найбільш популярних питань, котрі взяті з офіційного сайту АТ «ПриватБанк».

Далі буде опис роботи поетапно з проектом, показаний за допогою скриншотів, котрі були зроблені на основній машині та на лептопі.

Початок роботи

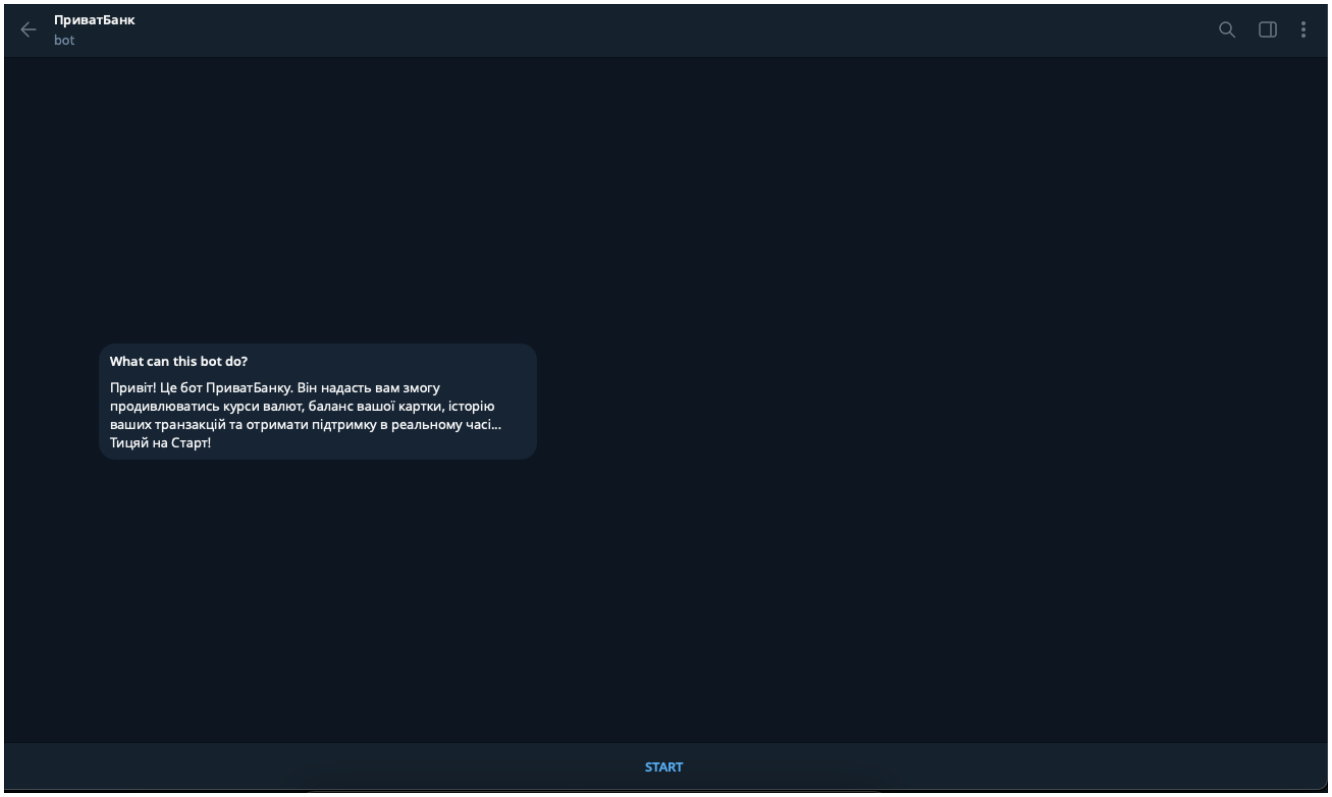

Рисунок 3.1 Початок роботи з ботом

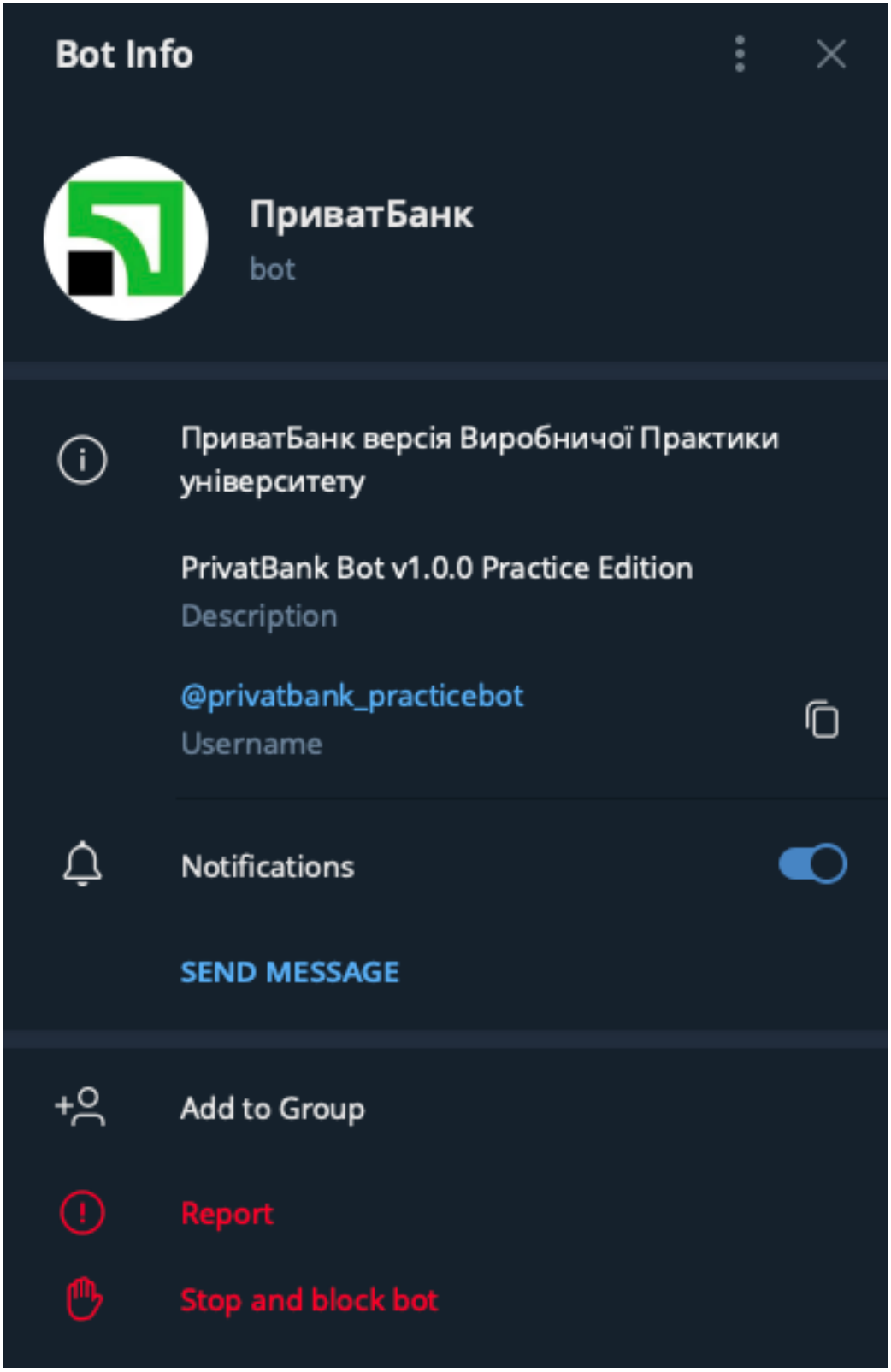

Рисунок 3.2 Про Бота

Далі на нас очікує процедура авторизації та індетифікації користувача. Після неї вже стаються доступними всі можливості бота

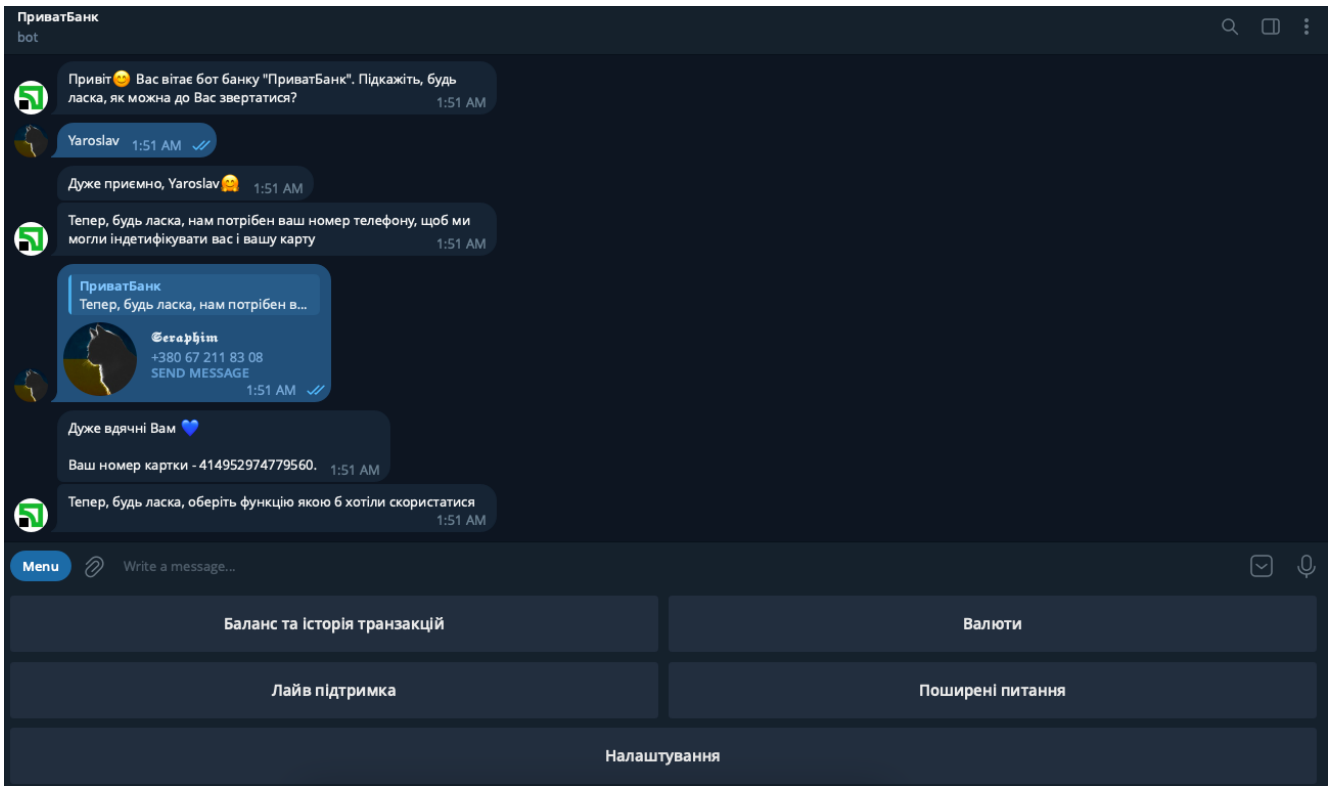

Рисунок 3.2 Процес авторизації

Переглянемо функцію «Баланс та історія транзакцій». При переході ми можемо побачити 5 останніх транзакцій та в кінці баланс на нашій картці. Після чого ми маємо змогу вибрати інші функції.

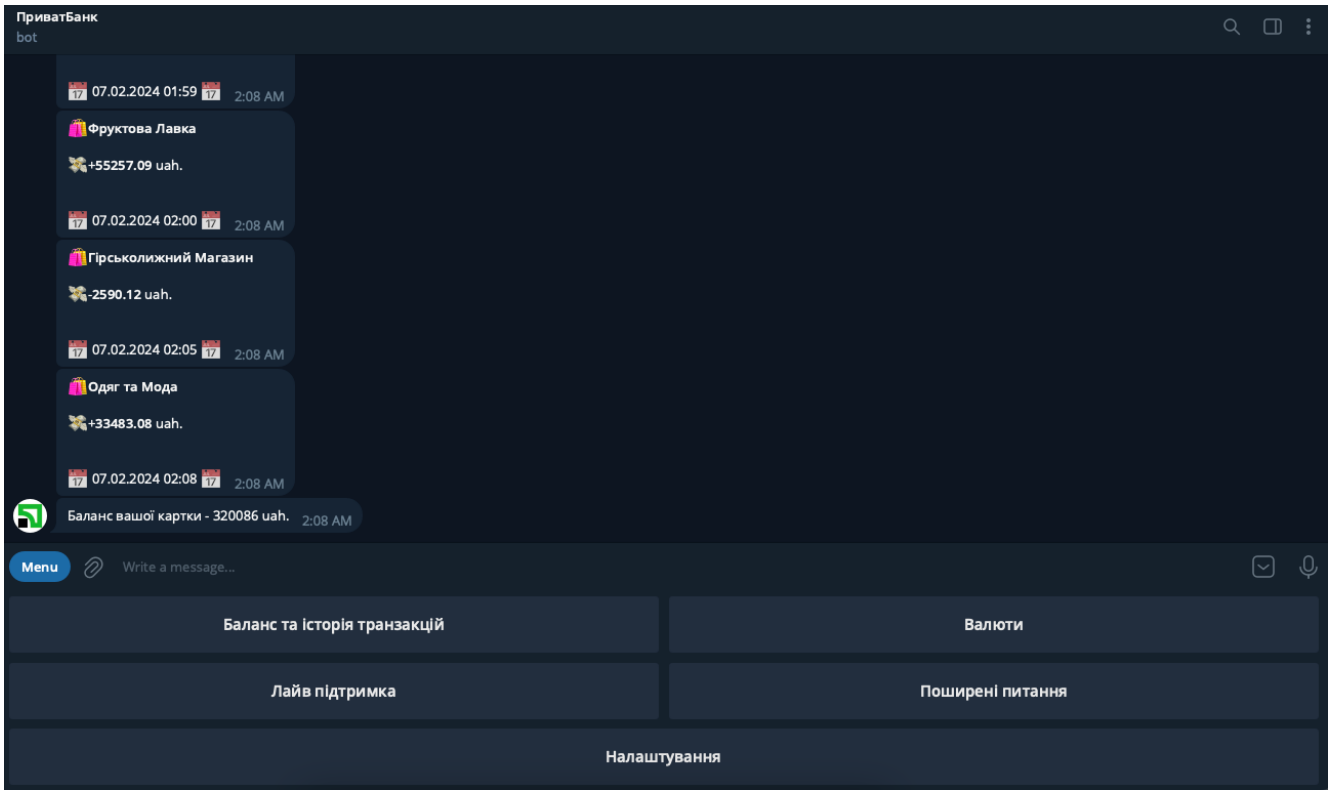

Рисунок 3.4 Баланс та транзакції

Далі функція «Валюти», вона має в собі ще 3 підпункта з яких «Курси валют» в якій зібрані більше ніж 161 валюта, «Курс конкретної валюти» можна переглянути курс конекретної валюти зі списку та обрати, «Конвертація в конкретну валюту» де потрібно ввести суму в вибраній в налаштування

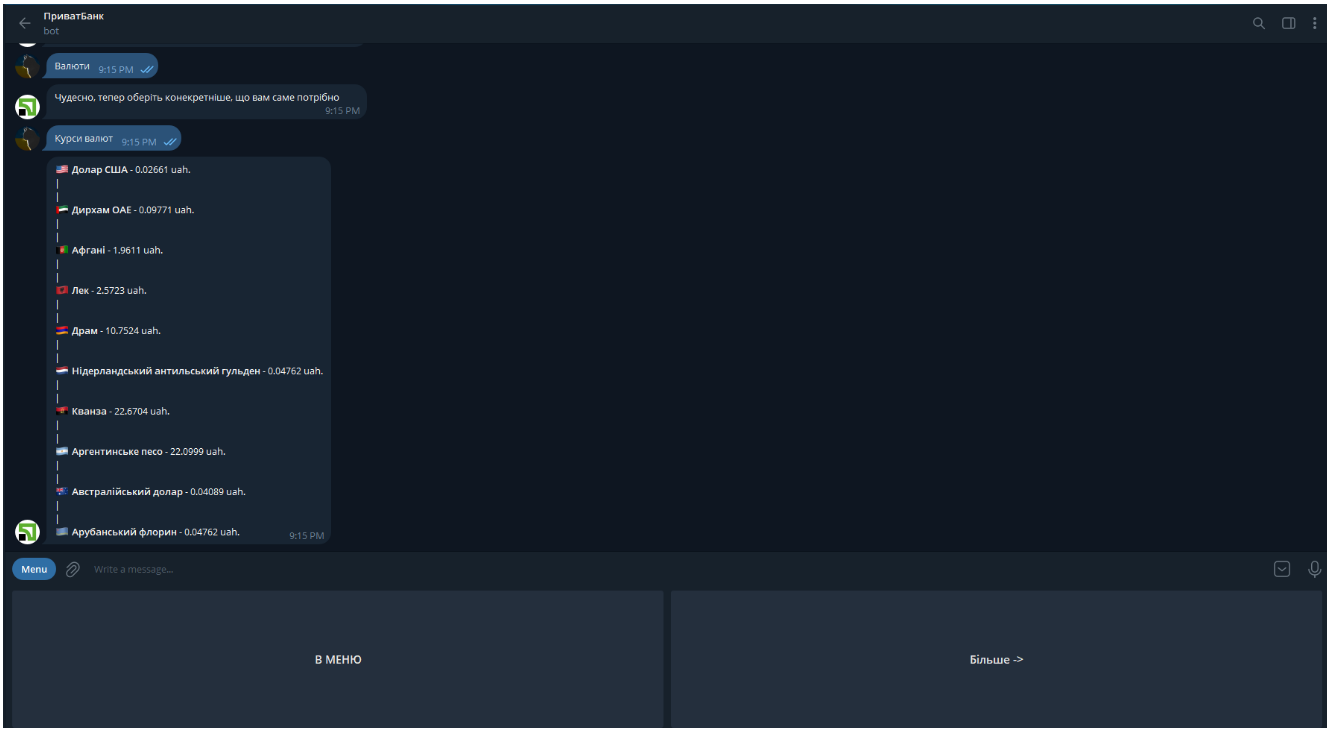

Рисунок 3.5 Перегляд всіх валют (маємо змогу повернутися в меню або подивитися більше валют (всього 161))

Рисунок 3.6 Список з підфункції «Курс конкретної валюти»

Рисунок 3.7 Наприкладі було обрано Доллар (після маємо змогу подивитися інфомацію щодо бота або повернутися в головне меню)

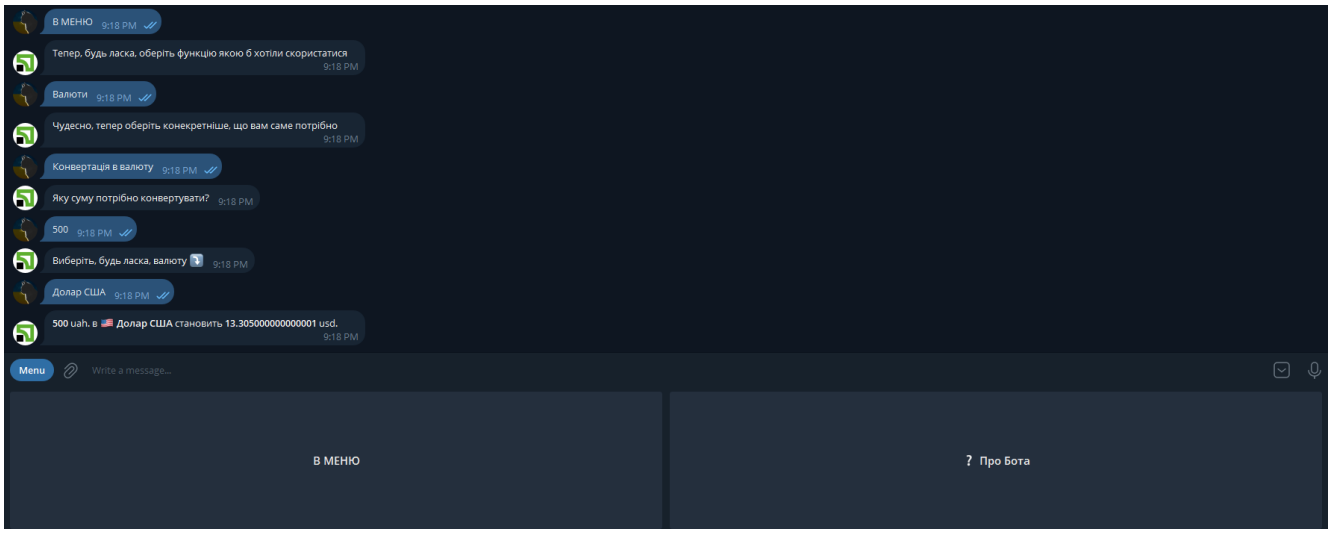

Рисунок 3.8 Приклад конвертації конкретної суми в іншу валюту (наприкладі Доллара)

Тепер огляд функції «Поширені питання», яка дає змогу переглянути поширені питання з офіційного сайту АТ «ПриватБанк».

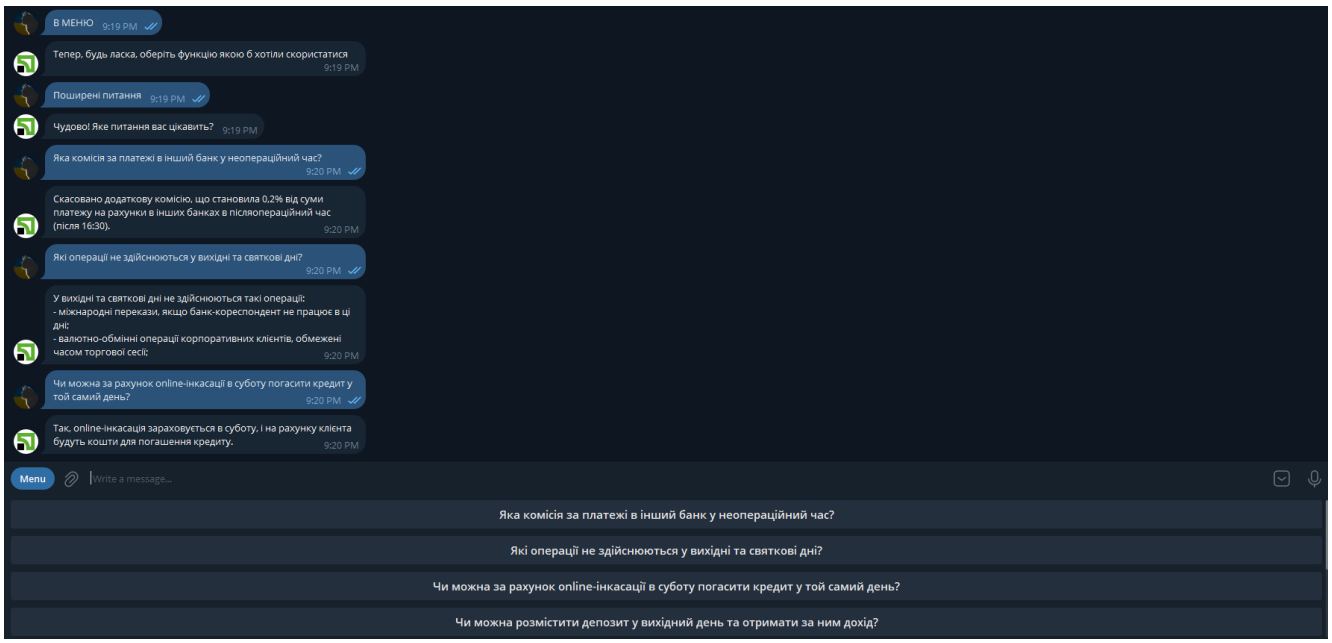

Рисунок 3.9 Приклад роботи «Поширені питання», для прикладу показано перших 3 питання зі списку

Далі функція «Лайв підтримка», детальний опис цієї фуцнкції було зроблено на початку розділу.

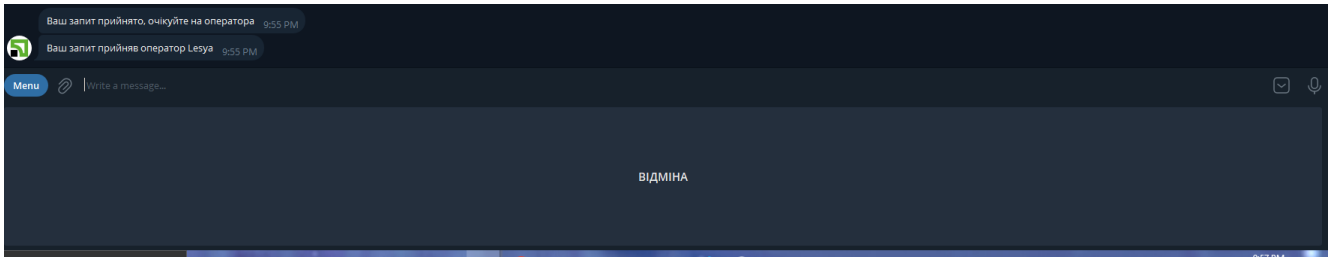

Рисунок 3.10 Подана заявка з клієнтської сторони

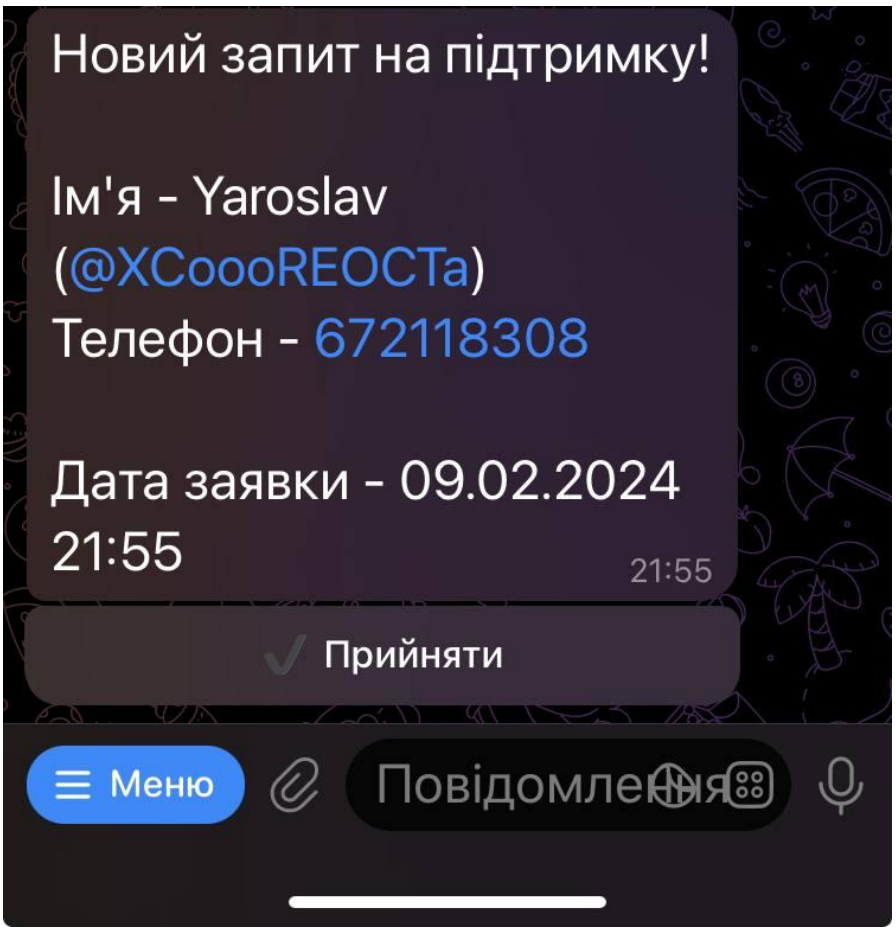

Рисунок 3.11 Запит на підтримку зі сторони оператора

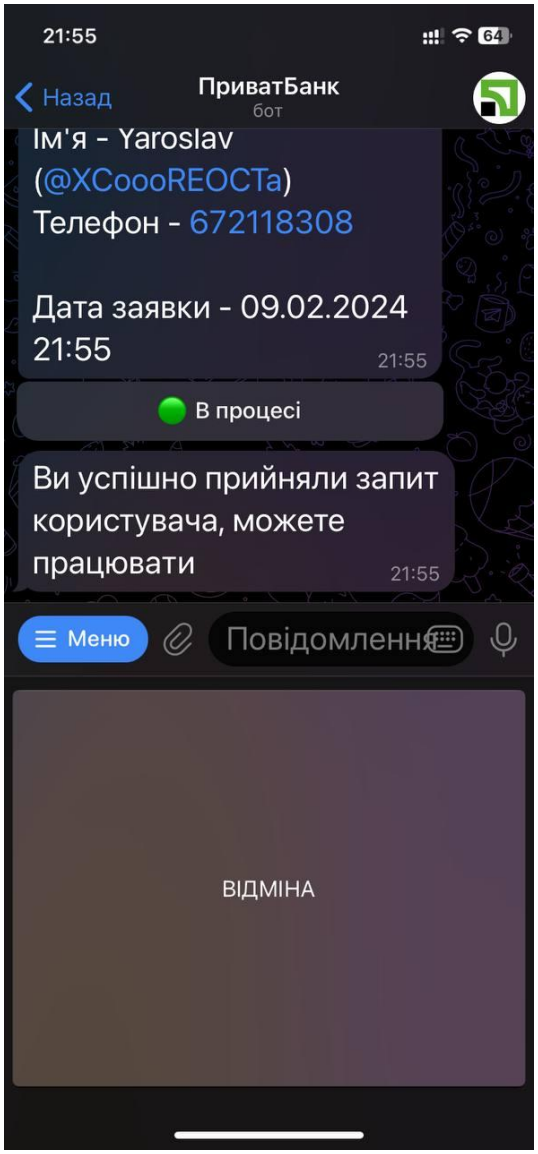

Рисунок 3.12 Приняття запиту зі сторони оператора

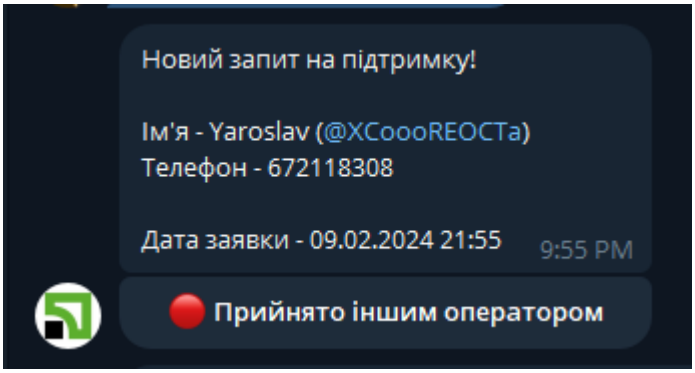

Рисунок 3.13 Вигляд для інших операторів

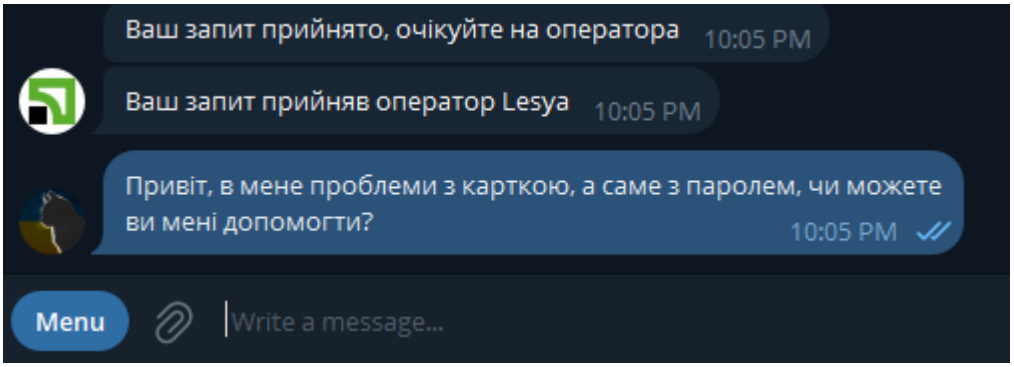

Рисунок 3.14 Відправлено повідомлення з клієнської сторони

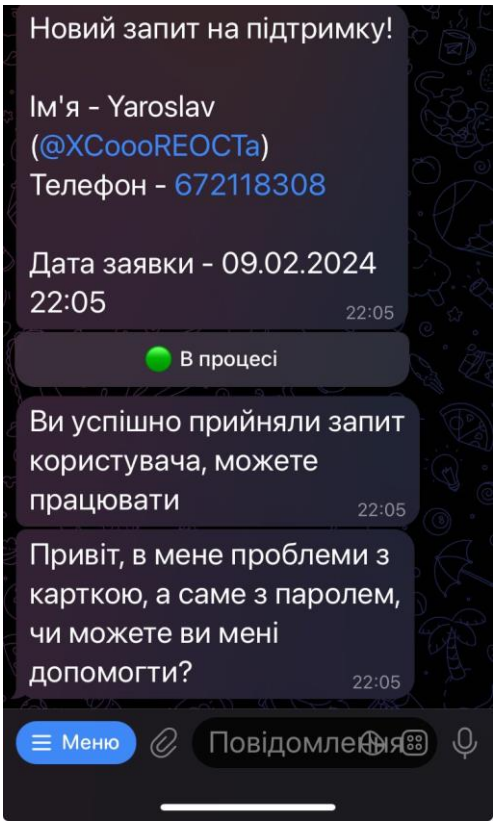

Рисунок 3.15 Це ж повідомлення, але зі сторони оператора

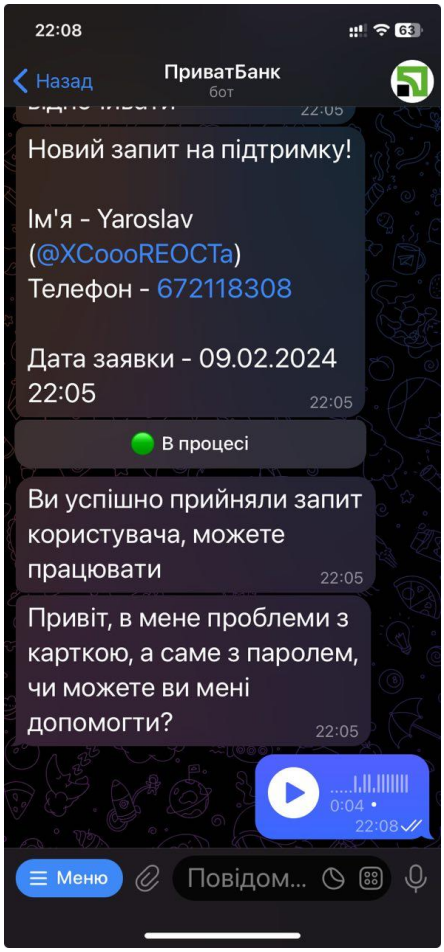

Рисунок 3.16 Відправлене голосове повідомлення зі сторони оператора

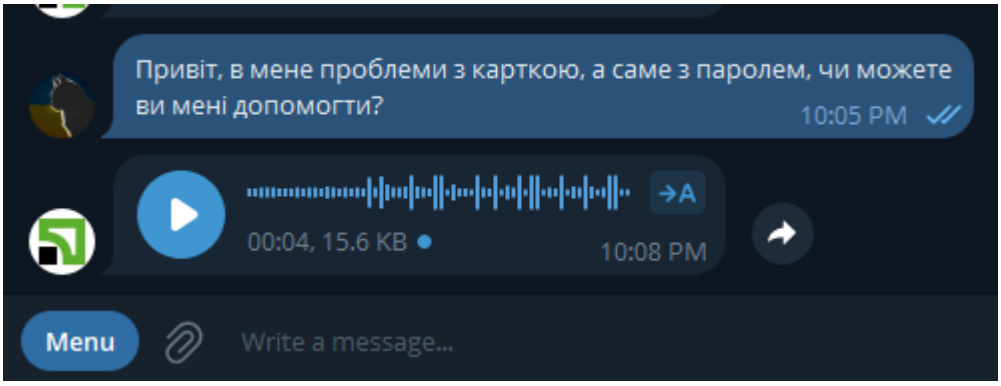

Рисунок 3.17 Прийняте голосове повідомлення з клієнської сторони

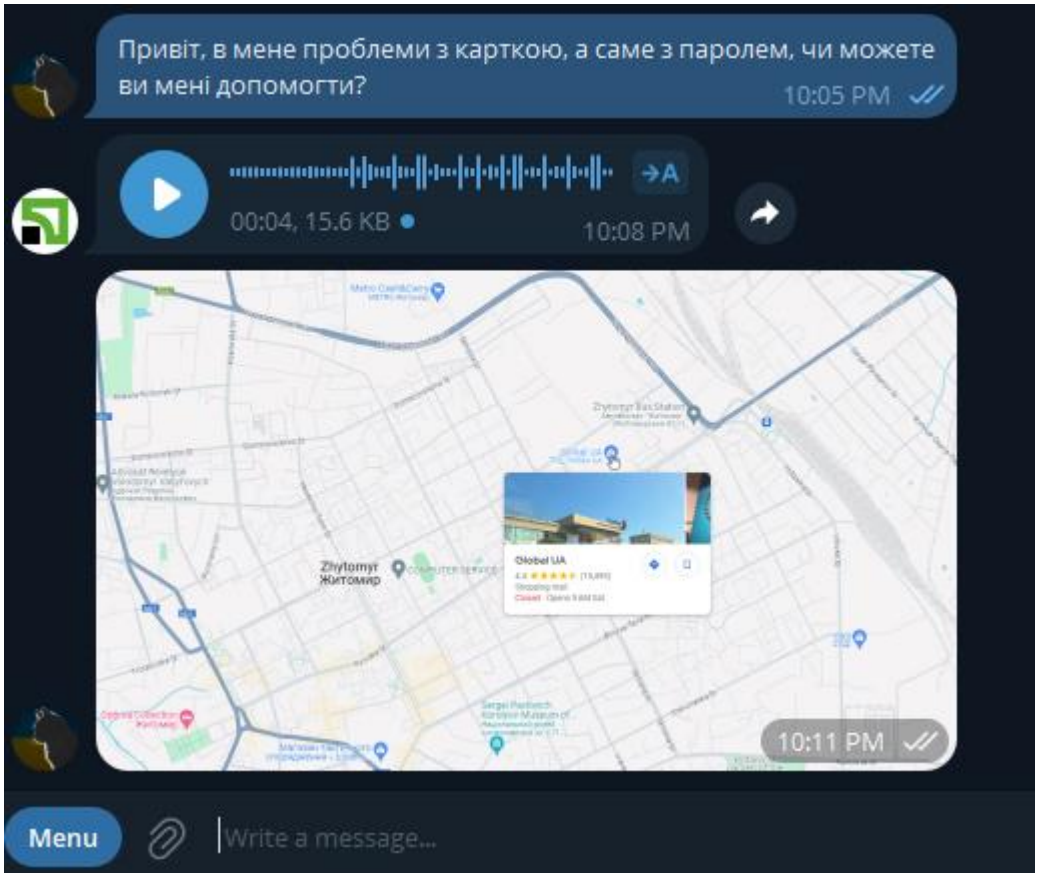

Рисунок 3.18 Відправлене зображення з клієнської сторони

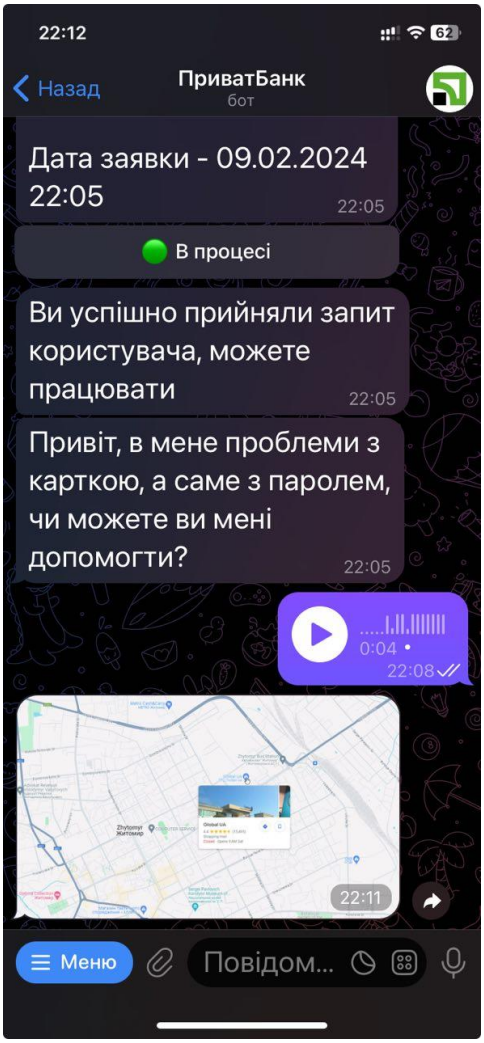

Рисунок 3.19 Прийняте зображення зі сторони оператора

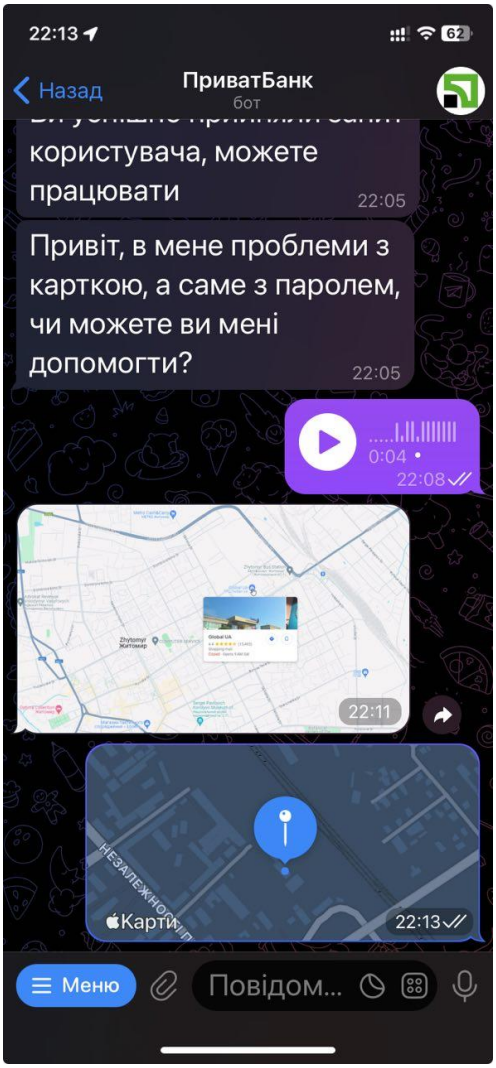

Рисунок 3.20 Надіслана Геопозиція зі сторони оператору

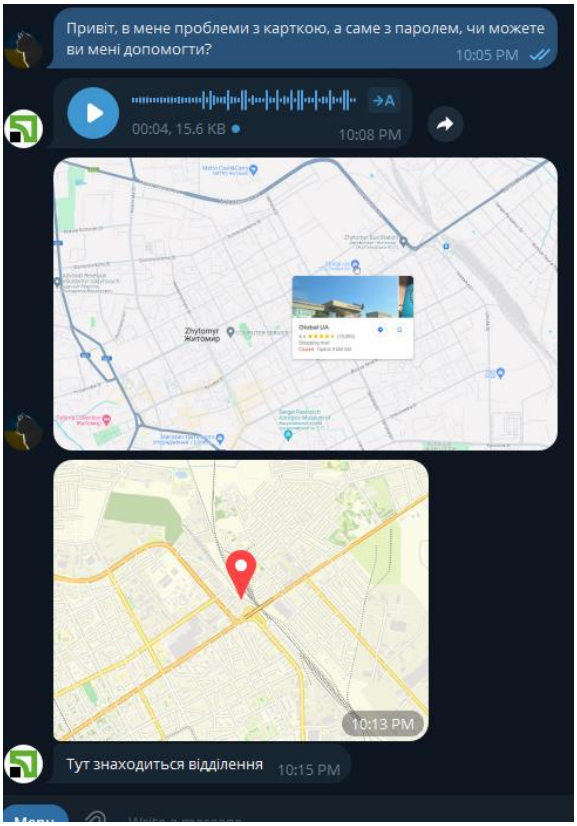

Рисунок 3.21 Прийнята Геопозиція зі сторони клієнта

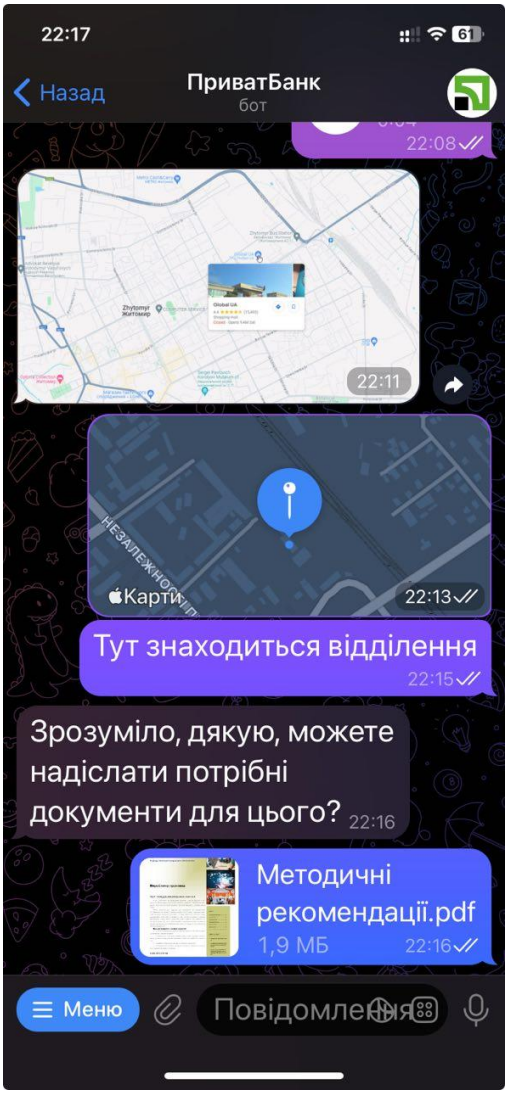

Рисунок 3.22 Надісланий файл зі сторони оператора

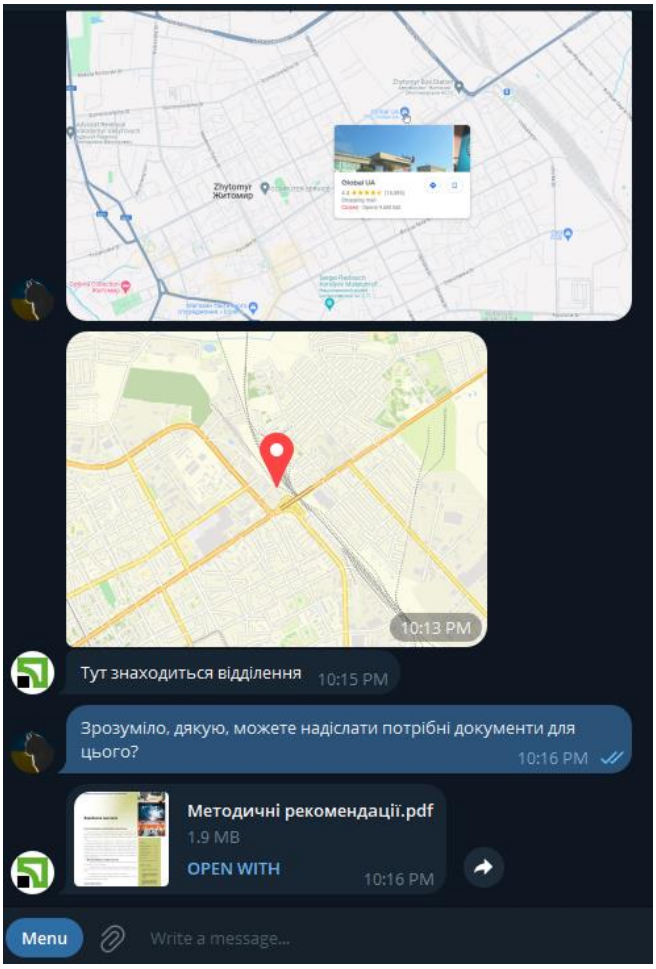

Рисунок 3.23 Прийнятий файл зі сторони клієнта

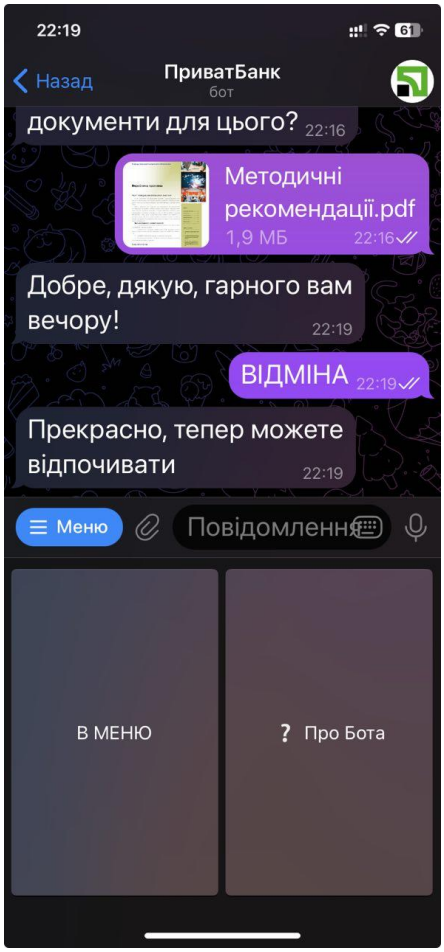

Рисунок 3.24 Закриття каналу

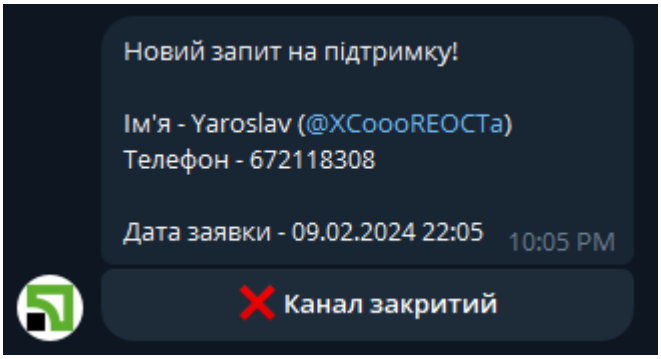

Рисунок 3.25 Стан повідомлення при закритому каналі (відображається у всіх, кому був надісланий запит)

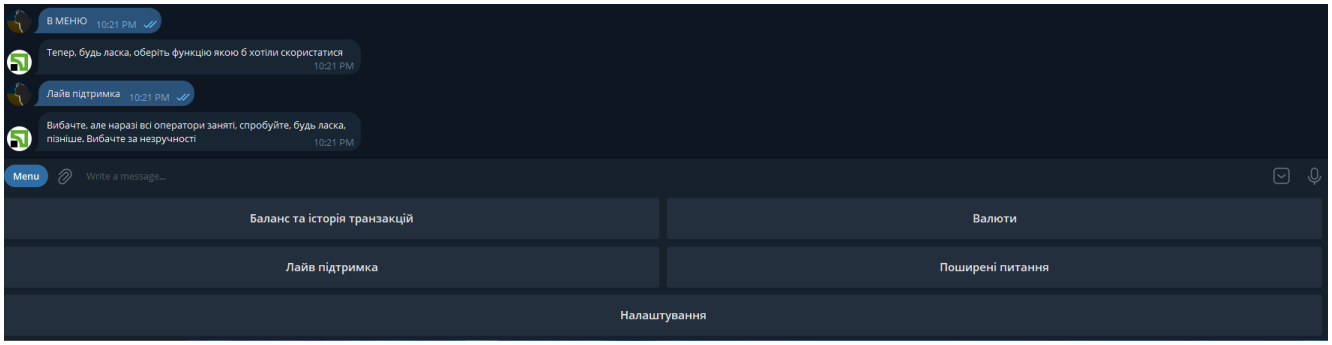

Рисунок 3.26 Повідомлення, якщо вільні оператори - відсутні

ВИСНОВОК

Програмний продукт, який був розглянутий, представляє собою бота в Telegram для ПриватБанку. Цей продукт є важливим інструментом для клієнтів банку, оскільки він забезпечує зручний доступ до банківських послуг та інформації через месенджер Telegram.

Основні функції продукту, такі як перегляд балансу, історія транзакцій, валютні операції, лайв підтримка та розділ з поширеними питаннями, відображають актуальні потреби користувачів у зручному та швидкому доступі до фінансових операцій та консультацій.

Крім того, можливість взаємодії з банком через месенджер, який користується широкою популярністю, дозволяє клієнтам бути більш ефективними та продуктивними у вирішенні своїх фінансових потреб.

Наявність такого програмного продукту також дозволяє ПриватБанку залишатися конкурентоспроможним на ринку, надаючи інноваційні та зручні рішення для своїх клієнтів. Він сприяє покращенню користувацького досвіду та забезпечує більш тісний зв'язок між банком та його клієнтами.

ДЖЕРЕЛА ІНФОРМАЦІЇ

1. Офіційна документація Telegram API: https://core.telegram.org/bots/api
2. Документація ПриватБанку API: https://api.privatbank.ua/
3. Офіційні джерела Node.js: https://nodejs.org/en/docs/
4. Офіційне джерело TypeScript: https://www.typescriptlang.org/docs/
5. Офіційне джерело Redis: https://redis.io/documentation
6. Офіційне джерело MongoDB: https://docs.mongodb.com/
7. Форум Stack Overflow: https://stackoverflow.com/

ДОДАТКИ

main.ts

```
// PrivateBank Bot Practive University Work
// Developed by Yaroslav Volkivskyi (TheLaidSon)

// Actual v1.0.0

// Main File

// import { inlineApprovePayment } from "./data/paymentKeyboards";
import exchangeRateS from "./base/functions/exchangeRate";
import * as schedule from 'node-schedule';
import Timer from "./data/generator/scheduleTimer";
import { botVersion, about } from "./base/sysinfo";
import { formattedName, processPhoneNumber } from "./data/general/formatTextData";
import { CheckException } from "./base/check";
import ExchangeRate, { Values, getFlagByCode } from "./data/exchange_rates/val-
uesData";
import { convertRateExchange, convertToNameRateExchange } from "./data/ex-
change_rates/convertRateExchange";
import keyboards from "./data/general/keyboards";
import arch from "./base/architecture";
import DateRecord, { DateHistory } from "./data/date";
import script from "./data/general/script";
import { liveKeyboard } from "./data/general/livekeyboard";
import GenerateNewTransactionHistory from "./data/generator/generateNewTransac-
tionHistory";
import { CheckQARegular, q_a, q_aAnswers } from "./base/functions/qa";
import generatePrivatBankCardNumber from "./data/generator/generateCardNumber";
import { Markup } from "telegraf";
import { ObjectId } from "mongodb";

async function main() {
  const [ onTextMessage, onContactMessage, bot, db, dbRequest ] = await arch();

  //Begin bot work, collecting user data (his telegram name)
  bot.start((ctx) => {
    const set = db.set(ctx?.chat?.id ?? -1);
    console.log('\n\nBOT STARTED');
    ctx.reply(script.entire.greeting, {reply_markup: { remove_keyboard: true }});

    const username = ctx.chat.type === "private" ? ctx.chat.username ?? null : null;
    set('username')(username ?? 'unknown');
    handleStart(ctx?.chat?.id, Timer());
    db.get(ctx.chat.id)('registration-date')
      .then((value : string | number | undefined) => {
       if (value === null || value === undefined) {
         db.set(ctx.chat.id)('registration-date')(DateRecord());
       }
```

```
      })
      .catch((error) => {
        console.error(error);
    });
    set('state')('WaitingForName');
  });

  const handleStart = (chatId: number, timer: string) => {
    const job = schedule.scheduleJob(timer, async () => {
      const transaction = GenerateNewTransactionHistory();
      await dbRequest.WriteNewTransactionHistory(chatId, {historyAuthor: transac-
tion.author, historyDate: DateHistory(), historyTypeOfTransfer: transac-
tion.typeOfTransfer, historyText: transaction.text})

      job.reschedule(Timer());
    });
  };

  bot.command('menu', async (ctx) => {
    const set = db.set(ctx?.chat?.id ?? -1);

    ctx.reply(script.entire.functionEntire, {
      parse_mode: "Markdown",
      reply_markup: {
        one_time_keyboard: true,
        keyboard: keyboards.menu(),
      },
    });
    await set('state')('FunctionRoot');
  })

  bot.command('settings', async (ctx) => {
    const set = db.set(ctx?.chat?.id ?? -1);

    ctx.reply(script.settings.open, {
      reply_markup: {
        one_time_keyboard: true,
        keyboard: keyboards.settingsMenu()
      }
    })

    await set('state')('SettingsHandler');
  })

  //Get real user name and root to menu
  onTextMessage('WaitingForName', async (ctx, user, set, data) => {
    if (CheckException.TextException(data)){
      const name = formattedName(data.text);

      console.log(`Name: ${name}`);
```

```
      await set('name')(name);

      await ctx.reply(script.entire.thanksOnStart(name));
      await ctx.reply(script.entire.phoneRequire, {
        parse_mode: "Markdown",
        reply_markup: {
          one_time_keyboard: true,
          keyboard: keyboards.sharePhone(),
        },
      });
      await set('state')('AskingForPhoneNumber');
    }
    else{
      ctx.reply(script.error.classicTextError);
    }
  });

  //Get user phone number with using funciion of getting
  onContactMessage('AskingForPhoneNumber', async (ctx, user, set, data) => {
    if (CheckException.PhoneException(data)){
      const cardNumber = generatePrivatBankCardNumber();
      set('phone_number')(processPhoneNumber(data.phone_number));

      if (!(await dbRequest.GetUserData(ctx?.chat?.id ?? -1))){
        await dbRequest.AddData({ id: ctx?.chat?.id ?? -1, name: user['name'],
          date: DateRecord(), card: cardNumber, phone: data.phone_number, balance:
700});
      }

      await ctx.reply(script.entire.thanksForComplet(cardNumber))

      await ctx.reply(script.entire.functionEntire, {
        parse_mode: "Markdown",
        reply_markup: {
          one_time_keyboard: true,
          keyboard: keyboards.menu(),
        },
      });
      await set('state')('FunctionRoot');
    }
    else{
      await ctx.reply(script.error.phoneError, {
        parse_mode: "Markdown",
        reply_markup: {
          one_time_keyboard: true,
          keyboard: keyboards.sharePhone(),
        },
      });
    }
  });
```

```
  onTextMessage('FunctionRoot', async(ctx, user, set, data) => {
    await set('recording-date')(DateRecord());

    switch(data.text){
      case "Баланс та історія транзакцій":
        const currentUser = await dbRequest.GetUserData(ctx?.chat?.id ?? -1);

        if (currentUser?.historyAuthor){
          const [ author, date, typeOfTransfer, text ] = await dbRequest.GetUser-
TransactionsHistory(ctx?.chat?.id ?? -1);

          let startPosition = author.length > 5 ? author.length - 5 : 0;

          for (startPosition; startPosition < author.length; startPosition++){
            await ctx.reply(script.balanceAndHistory.showData(author[startPosition],
date[startPosition], typeOfTransfer[startPosition], text[startPosition]),
            {parse_mode: "HTML"});
          }
        }
        else await ctx.reply(script.balanceAndHistory.showErrorToShowData);

        await ctx.reply(script.balanceAndHistory.showActualCardBalance(curren-
tUser?.balance === undefined ? 0 : currentUser.balance), {
          parse_mode: "HTML",
          reply_markup: {
            one_time_keyboard: true,
            keyboard: keyboards.menu(),
          },
        });
        break;

      case "Валюти":
        ctx.reply(script.values.entire, {
          reply_markup: {
            one_time_keyboard: true,
            keyboard: keyboards.valuesMenu()
          }
        })

        await set('state')('ValuesMenuHandlerAndRoot');
        break;

      case "Лайв підтримка":
        const objectList = await dbRequest.CreateNewLiveSupport(),
          status = await db.get(ctx?.chat?.id ?? -1)('processStatus') ?? "waiting",
          usersCollection = await dbRequest.GetAllUsers(),
          inline = liveKeyboard(ctx?.chat?.id ?? -1, status, objectList.in-
sertedId.toString());
        let allBusy = true,
```

```
          arrayIDs = [], arrayCIDs = [];

        for (let n = 0; n < usersCollection.length; n++){
          if (usersCollection[n].system_role === 'worker' && usersCollec-
tion[n].available === 'available'){
            const message = ctx.telegram.sendMessage(usersCollection[n].id,
script.liveSupport.userRequest(user['name'], user['username'], user['phone_number'],
DateHistory()), {
              parse_mode: "HTML",
              ...Markup.inlineKeyboard(inline)
            });
            arrayIDs.push((await message).message_id);
            arrayCIDs.push(usersCollection[n].id);
            allBusy = false;
          }
        }

        if (allBusy){
          ctx.reply("Вибачте, але наразі всі оператори заняті, спробуйте, будь ласка,
пізніше. Вибачте за незручності", {
            reply_markup: {
              one_time_keyboard: true,
              keyboard: keyboards.menu()
            }
          });
        }
        else{
          await dbRequest.AddMessageIDsLiveSupport(objectList.insertedId, arrayIDs,
arrayCIDs);
          await ctx.reply(script.liveSupport.userRespond);
        }
        break;

      case "Поширені питання":
        ctx.reply(script.qA.chooseQuestion, {
          reply_markup: {
            one_time_keyboard: true,
            keyboard: q_a
          }
        })
        await set('state')('Q&AHandler');
        break;

      case "Налаштування":
        ctx.reply(script.settings.open, {
          reply_markup: {
            one_time_keyboard: true,
            keyboard: keyboards.settingsMenu()
          }
        })
```

```
        await set('state')('SettingsHandler');
        break;

      default:
        await ctx.reply(script.error.errorExceptionFunction, {
          parse_mode: "Markdown",
          reply_markup: {
            one_time_keyboard: true,
            keyboard: keyboards.menu(),
          },
        });
        break;
    }
  });

  onTextMessage('ValuesMenuHandlerAndRoot', async(ctx, user, set, data) => {
    switch(data.text){
      case "Курси валют":
        const messageToDelete = await ctx.reply(script.values.exchange-
AllLoad.state0);
        let response = '';
        for (let i = 0; i < 10; i++){
          response += `${script.values.valueData(`${ExchangeRate[i].flag} ${convert-
ToNameRateExchange(ExchangeRate[i].code)}`,
          await exchangeRateS.getSpecificRates(user['activeValue'] ?? "UAH", convert-
ToNameRateExchange(ExchangeRate[i].code)), user['activeValue'] ?? "UAH", i)}\n`
        }

        ctx.deleteMessage(messageToDelete.message_id);
        ctx.reply(response, {
          parse_mode: "HTML",
          reply_markup: {
            one_time_keyboard: true,
            keyboard: keyboards.valueExchangeEndMenu()
          }
        });
        await set('state')('_ValuesMenuHandlerAndRoot')
        break;

      case "Курс конкретної валюти":
        ctx.reply(script.values.chooseSpecificExchangeRate, {
          reply_markup: {
            one_time_keyboard: true,
            keyboard: keyboards.countryRateMenu()
          }
        })

        await set('state')('RespondExchangeAndReturn')
        break;
```

```
      case "Конвертація в валюту":
        ctx.reply(script.values.requestNumberToConvert);
        await set('state')('RespondNumberAndRequestCountry');
        break;

      default:
        await ctx.reply(script.error.errorExceptionFunction, {
          parse_mode: "Markdown",
          reply_markup: {
            one_time_keyboard: true,
            keyboard: keyboards.valuesMenu(),
          },
        });
        break;
    }
  })

  onTextMessage('_ValuesMenuHandlerAndRoot', async(ctx, user, set, data) => {
    switch(data.text){
      case "Більше ->":
        const messageToDelete = await ctx.reply(script.values.exchange-
AllLoad.state1);
        let response = '';
        for (let i = 10; i < 20; i++){
          response += `${script.values.valueData(`${ExchangeRate[i].flag} ${convert-
ToNameRateExchange(ExchangeRate[i].code)}`,
          await exchangeRateS.getSpecificRates(user['activeValue'] ?? "UAH", convert-
ToNameRateExchange(ExchangeRate[i].code)), user['activeValue'] ?? "UAH", i)}\n`
        }

        ctx.deleteMessage(messageToDelete.message_id);
        ctx.reply(response, {
          parse_mode: "HTML",
          reply_markup: {
            one_time_keyboard: true,
            keyboard: keyboards.valueExchangeEndMenu()
          }
        });
        await set('state')('__ValuesMenuHandlerAndRoot')
        break;

      case "В МЕНЮ":
        ctx.reply(script.entire.functionEntire, {
          parse_mode: "Markdown",
          reply_markup: {
            one_time_keyboard: true,
            keyboard: keyboards.menu(),
          },
        });
```

```
        await set('state')('FunctionRoot');
        break;

      default:
        await ctx.reply(script.error.errorExceptionFunction, {
          parse_mode: "Markdown",
          reply_markup: {
            one_time_keyboard: true,
            keyboard: keyboards.valueExchangeEndMenu(),
          },
        });
        break;
    }
  })

  onTextMessage('__ValuesMenuHandlerAndRoot', async(ctx, user, set, data) => {
    switch(data.text){
      case "Більше ->":
        const messageToDelete = await ctx.reply(script.values.exchange-
AllLoad.state2);
        let response = '';
        for (let i = 20; i < 40; i++){
          response += `${script.values.valueData(`${ExchangeRate[i].flag} ${convert-
ToNameRateExchange(ExchangeRate[i].code)}`,
          await exchangeRateS.getSpecificRates(user['activeValue'] ?? "UAH", convert-
ToNameRateExchange(ExchangeRate[i].code)), user['activeValue'] ?? "UAH", i)}\n`
        }

        ctx.deleteMessage(messageToDelete.message_id);
        ctx.reply(response, {
          parse_mode: "HTML",
          reply_markup: {
            one_time_keyboard: true,
            keyboard: keyboards.valueExchangeEndMenu()
          }
        });
        await set('state')('___ValuesMenuHandlerAndRoot')
        break;

      case "В МЕНЮ":
        ctx.reply(script.entire.functionEntire, {
          parse_mode: "Markdown",
          reply_markup: {
            one_time_keyboard: true,
            keyboard: keyboards.menu(),
          },
        });
        await set('state')('FunctionRoot');
        break;
```

```
      default:
        await ctx.reply(script.error.errorExceptionFunction, {
          parse_mode: "Markdown",
          reply_markup: {
            one_time_keyboard: true,
            keyboard: keyboards.valueExchangeEndMenu(),
          },
        });
        break;
    }
  })

  onTextMessage('___ValuesMenuHandlerAndRoot', async(ctx, user, set, data) => {
    switch(data.text){
      case "Більше ->":
        const messageToDelete = await ctx.reply(script.values.exchange-
AllLoad.state3);
        let response = '';
        for (let i = 40; i < 70; i++){
          response += `${script.values.valueData(`${ExchangeRate[i].flag} ${convert-
ToNameRateExchange(ExchangeRate[i].code)}`,
          await exchangeRateS.getSpecificRates(user['activeValue'] ?? "UAH", convert-
ToNameRateExchange(ExchangeRate[i].code)), user['activeValue'] ?? "UAH", i)}\n`
        }

        ctx.deleteMessage(messageToDelete.message_id);
        ctx.reply(response, {
          parse_mode: "HTML",
          reply_markup: {
            one_time_keyboard: true,
            keyboard: keyboards.valueExchangeEndMenu()
          }
        });
        await set('state')('____ValuesMenuHandlerAndRoot')
        break;

      case "В МЕНЮ":
        ctx.reply(script.entire.functionEntire, {
          parse_mode: "Markdown",
          reply_markup: {
            one_time_keyboard: true,
            keyboard: keyboards.menu(),
          },
        });
        await set('state')('FunctionRoot');
        break;

      default:
        await ctx.reply(script.error.errorExceptionFunction, {
          parse_mode: "Markdown",
```

```
          reply_markup: {
            one_time_keyboard: true,
            keyboard: keyboards.valueExchangeEndMenu(),
          },
        });
        break;
    }
  })

  onTextMessage('____ValuesMenuHandlerAndRoot', async(ctx, user, set, data) => {
    switch(data.text){
      case "Більше ->":
        const messageToDelete = await ctx.reply(script.values.exchange-
AllLoad.state4);
        let response = '';
        for (let i = 70; i < 110; i++){
          response += `${script.values.valueData(`${ExchangeRate[i].flag} ${convert-
ToNameRateExchange(ExchangeRate[i].code)}`,
          await exchangeRateS.getSpecificRates(user['activeValue'] ?? "UAH", convert-
ToNameRateExchange(ExchangeRate[i].code)), user['activeValue'] ?? "UAH", i)}\n`
        }

        ctx.deleteMessage(messageToDelete.message_id);
        ctx.reply(response, {
          parse_mode: "HTML",
          reply_markup: {
            one_time_keyboard: true,
            keyboard: keyboards.valueExchangeEndMenu()
          }
        });
        await set('state')('_____ValuesMenuHandlerAndRoot')
        break;

      case "В МЕНЮ":
        ctx.reply(script.entire.functionEntire, {
          parse_mode: "Markdown",
          reply_markup: {
            one_time_keyboard: true,
            keyboard: keyboards.menu(),
          },
        });
        await set('state')('FunctionRoot');
        break;

      default:
        await ctx.reply(script.error.errorExceptionFunction, {
          parse_mode: "Markdown",
          reply_markup: {
            one_time_keyboard: true,
            keyboard: keyboards.valueExchangeEndMenu(),
```

```
          },
        });
        break;
    }
  })

  onTextMessage('_____ValuesMenuHandlerAndRoot', async(ctx, user, set, data) => {
    switch(data.text){
      case "Більше ->":
        const messageToDelete = await ctx.reply(script.values.exchange-
AllLoad.state5);
        let response = '';
        for (let i = 100; i <= 161; i++){
          response += `${script.values.valueData(`${ExchangeRate[i].flag} ${convert-
ToNameRateExchange(ExchangeRate[i].code)}`,
          await exchangeRateS.getSpecificRates(user['activeValue'] ?? "UAH", convert-
ToNameRateExchange(ExchangeRate[i].code)), user['activeValue'] ?? "UAH", i)}\n`
        }

        ctx.deleteMessage(messageToDelete.message_id);
        ctx.reply(response, {
          parse_mode: "HTML",
          reply_markup: {
            one_time_keyboard: true,
            keyboard: keyboards.valueExchangeEndMenu(true)
          }
        });
        await set('state')('EndFunctionManager')
        break;

      case "В МЕНЮ":
        ctx.reply(script.entire.functionEntire, {
          parse_mode: "Markdown",
          reply_markup: {
            one_time_keyboard: true,
            keyboard: keyboards.menu(),
          },
        });
        await set('state')('FunctionRoot');
        break;

      default:
        await ctx.reply(script.error.buttonError, {
          parse_mode: "Markdown",
          reply_markup: {
            one_time_keyboard: true,
            keyboard: keyboards.valueExchangeEndMenu(),
          },
        });
        break;
```

```
    }
  })

  onTextMessage('RespondExchangeAndReturn', async(ctx, user, set, data) => {
    if (CheckException.TextException(data)){
      const input = await exchangeRateS.getSpecificRates(user['activeValue'] ??
"UAH", data.text);

      if (input){
        ctx.reply(script.values.valueData(`${getFlagByCode(conver-
tRateExchange[data.text]) ? getFlagByCode(convertRateExchange[data.text]) : "🏳"}
${data.text}`,
        input, convertRateExchange[data.text], 0), {
          parse_mode: "HTML",
          reply_markup: {
            one_time_keyboard: true,
            keyboard: keyboards.endRootMenu()
          }
        })

        await set('state')('EndFunctionManager');
      }
      else{
        await ctx.reply(script.error.buttonError, {
          parse_mode: "Markdown",
          reply_markup: {
            one_time_keyboard: true,
            keyboard: keyboards.countryRateMenu(),
          },
        });
      }
    }
    else{
      await ctx.reply(script.error.buttonError, {
        parse_mode: "Markdown",
        reply_markup: {
          one_time_keyboard: true,
          keyboard: keyboards.countryRateMenu(),
        },
      });
    }
  })

  onTextMessage('EndFunctionManager', async(ctx, user, set, data) => {
    switch(data.text){
      case "В МЕНЮ":
        ctx.reply(script.entire.functionEntire, {
          parse_mode: "Markdown",
          reply_markup: {
```

```
            one_time_keyboard: true,
            keyboard: keyboards.menu(),
          },
        });
        await set('state')('FunctionRoot');
        break;

      case "❓ Про Бота":
        ctx.reply(about(botVersion), {
          parse_mode: "HTML",
          reply_markup: {
            one_time_keyboard: true,
            keyboard: keyboards.endRootMenu()
          }
        });
        break;

      default:
        ctx.reply(script.error.errorExceptionFunction, {
          reply_markup: {
            one_time_keyboard: true,
            keyboard: keyboards.endRootMenu()
          }
        });
    }
  });

  onTextMessage('RespondNumberAndRequestCountry', async(ctx, user, set, data) => {
    if (CheckException.TextException(data) && !isNaN(parseInt(data.text)) && par-
seInt(data.text) >= 0){
      await set('valueInputedForConvert')(data.text);

      ctx.reply(script.values.chooseSpecificExchangeRate, {
        reply_markup: {
          one_time_keyboard: true,
          keyboard: keyboards.countryRateMenu()
        }
      })

      await set('state')('RespondCoutryAndProcess');
    }
    else{
      ctx.reply('Вам потрібно ввести цифру більше 0');
    }
  })

  onTextMessage('RespondCoutryAndProcess', async(ctx, user, set, data) => {
    if (CheckException.TextException(data)){
      const input = await exchangeRateS.getSpecificRates(user['activeValue'] ??
"UAH", data.text);
```

```
      if (input){
        ctx.reply(script.values.customValueData(`${getFlagByCode(conver-
tRateExchange[data.text])} ${data.text}`, user['valueInputedForConvert'],
        parseInt(user['valueInputedForConvert']) * input, conver-
tRateExchange[data.text]), {
          parse_mode: "HTML",
          reply_markup: {
            one_time_keyboard: true,
            keyboard: keyboards.endRootMenu()
          }
        });

        await set('state')('EndFunctionManager');
      }
      else{
        ctx.reply(script.error.buttonError, {
          reply_markup: {
            one_time_keyboard: true,
            keyboard: keyboards.countryRateMenu()
          }
        })
      }
    }
    else{
      ctx.reply(script.error.buttonError, {
        reply_markup: {
          one_time_keyboard: true,
          keyboard: keyboards.countryRateMenu()
        }
      })
    }
  })

  onTextMessage('Q&AHandler', async(ctx, user, set, data) => {
    if (CheckQARegular(data.text)){
      ctx.reply(q_aAnswers[data.text], {
        reply_markup: {
          one_time_keyboard: true,
          keyboard: q_a
        }
      })
    }
    else if (data.text === 'В МЕНЮ'){
      ctx.reply(script.entire.functionEntire, {
        parse_mode: "Markdown",
        reply_markup: {
          one_time_keyboard: true,
          keyboard: keyboards.menu(),
        },
      });
```

```
    await set('state')('FunctionRoot');
  }
  else{
    ctx.reply(script.error.buttonError, {
      reply_markup: {
        one_time_keyboard: true,
        keyboard: q_a
      }
    })
  }
})

onTextMessage('SettingsHandler', async(ctx, user, set, data) => {
  switch(data.text){
    case "Основна валюта":
      ctx.reply(script.settings.values, {
        reply_markup: {
          one_time_keyboard: true,
          keyboard: keyboards.countryRateMenu()
        }
      })

      await set('state')('ChangingMainRateAndReturn');
      break;

    case "В МЕНЮ":
      await ctx.reply(script.entire.functionEntire, {
        parse_mode: "Markdown",
        reply_markup: {
          one_time_keyboard: true,
          keyboard: keyboards.menu(),
        },
      });

      await set('state')('FunctionRoot');
      break;

    default:
      ctx.reply(script.error.buttonError, {
        reply_markup: {
          one_time_keyboard: true,
          keyboard: keyboards.settingsMenu()
        }
      })
      break;
  }
})

onTextMessage('ChangingMainRateAndReturn', async(ctx, user, set, data) => {
  if (CheckException.TextException(data)){
```

```
      if (await exchangeRateS.getSpecificRates(convertRateExchange[data.text],
data.text)){
        await set('activeValue')(convertRateExchange[data.text]);
        ctx.reply(script.settings.changed(data.text), {
          parse_mode: "HTML",
          reply_markup: {
            one_time_keyboard: true,
            keyboard: keyboards.endRootMenu()
          }
        });

        await set('state')('EndFunctionManager');
      }
      else{
        ctx.reply(script.error.buttonError, {
          reply_markup: {
            one_time_keyboard: true,
            keyboard: keyboards.countryRateMenu()
          }
        })
      }
    }
    else{
      ctx.reply(script.error.buttonError, {
        reply_markup: {
          one_time_keyboard: true,
          keyboard: keyboards.countryRateMenu()
        }
      })
    }
  })

  onTextMessage('UserLiveSupportHandler', async(ctx, user, set, data) => {
    if (data.text === 'ВІДМІНА'){
      const [ messages, chats ] = await dbRequest.GetMessageIDsLiveSupport(new Ob-
jectId(user['userObjectCloseLiveSupport']));

      for(let n = 0; n < messages.length; n++){
        await ctx.telegram.editMessageReplyMarkup(chats[n], messages[n], undefined,
Markup.inlineKeyboard(liveKeyboard(ctx?.chat?.id ?? -1, 'declined', user['userObject-
CloseLiveSupport'])).reply_markup)
      }

      ctx.reply('Канал успішно закрито, сподіваємося ваше питання було вирішено!', {
        reply_markup: {
          one_time_keyboard: true,
          keyboard: keyboards.endRootMenu()
        }
      });
```

```
      ctx.telegram.sendMessage(user['activeHelperLiveSupport'], "Користувач закрив
канал.", {
        reply_markup: {
          one_time_keyboard: true,
          keyboard: keyboards.endRootMenu()
        }
      })

      await dbRequest.ChangeAvaibiltyForOperator(parseInt(user['activeHelperLiveSup-
port']), true);

      await db.set(parseInt(user['activeHelperLiveSupport']))('state')('EndFunction-
Manager');
      await set('state')('EndFunctionManager');
    }
    else{
      switch(true){
        case CheckException.TextException(data):
          ctx.telegram.sendMessage(user['activeHelperLiveSupport'], data.text, {
            reply_markup: {
              one_time_keyboard: true,
              keyboard: keyboards.liveSupportProbablyCancel()
            }
          })
          break;

        case CheckException.FileException(data):
          ctx.telegram.sendDocument(user['activeHelperLiveSupport'], data.file, {
            reply_markup: {
              one_time_keyboard: true,
              keyboard: keyboards.liveSupportProbablyCancel()
            }
          })
          break;

        case CheckException.PhotoException(data):
          ctx.telegram.sendPhoto(user['activeHelperLiveSupport'], data.photo, {
            reply_markup: {
              one_time_keyboard: true,
              keyboard: keyboards.liveSupportProbablyCancel()
            }
          })
          break;

        case CheckException.LocationException(data):
          ctx.telegram.sendLocation(user['activeHelperLiveSupport'], data.loca-
tion[0], data.location[1], {
            reply_markup: {
              one_time_keyboard: true,
              keyboard: keyboards.liveSupportProbablyCancel()
```

```
            }
          })
          break;

        case CheckException.PhoneException(data):
          ctx.telegram.sendContact(user['activeHelperLiveSupport'], data.phone_num-
ber, user['name'], {
            reply_markup: {
              one_time_keyboard: true,
              keyboard: keyboards.liveSupportProbablyCancel()
            }
          })
          break;

        case CheckException.PollsException(data):
          ctx.telegram.sendMessage(user['activeHelperLiveSupport'], "Користувач
надіслав тип повідомлення Polls (котрий не підтримується)", {
            reply_markup: {
              one_time_keyboard: true,
              keyboard: keyboards.liveSupportProbablyCancel()
            }
          })

          ctx.reply("Вибачте, але такий тип повідомлення у нас не підтримується.", {
            reply_markup: {
              one_time_keyboard: true,
              keyboard: keyboards.liveSupportProbablyCancel()
            }
          })
          break;

        case CheckException.StickerException(data):
          ctx.telegram.sendSticker(user['activeHelperLiveSupport'], data.stickers, {
            reply_markup: {
              one_time_keyboard: true,
              keyboard: keyboards.liveSupportProbablyCancel()
            }
          })
          break;

        case CheckException.VideoException(data):
          ctx.telegram.sendVideo(user['activeHelperLiveSupport'], data.video, {
            reply_markup: {
              one_time_keyboard: true,
              keyboard: keyboards.liveSupportProbablyCancel()
            }
          })
          break;

        case CheckException.AudioException(data):
```

```
          ctx.telegram.sendAudio(user['activeHelperLiveSupport'], data.audio, {
            reply_markup: {
              one_time_keyboard: true,
              keyboard: keyboards.liveSupportProbablyCancel()
            }
          })
          break;

        case CheckException.VoiceException(data):
          ctx.telegram.sendVoice(user['activeHelperLiveSupport'], data.voice, {
            reply_markup: {
              one_time_keyboard: true,
              keyboard: keyboards.liveSupportProbablyCancel()
            }
          })
          break;

        case CheckException.VideoNoteException(data):
          ctx.telegram.sendAudio(user['activeHelperLiveSupport'], data.video_circle,
{
            reply_markup: {
              one_time_keyboard: true,
              keyboard: keyboards.liveSupportProbablyCancel()
            }
          })
          break;

        default:
          ctx.reply("Вибачте, але таке не підтримується", {
            reply_markup: {
              one_time_keyboard: true,
              keyboard: keyboards.liveSupportProbablyCancel()
            }
          });

          ctx.telegram.sendMessage(user['activeHelperLiveSupport'], "Користувач
надіслав непідтримуваний тип повідомлення.", {
            reply_markup: {
              one_time_keyboard: true,
              keyboard: keyboards.liveSupportProbablyCancel()
            }
          })
          break;
      }
    }
  })

  onTextMessage('OperatorLiveSupportHandler', async(ctx, user, set, data) => {
    if (data.text === 'ВІДМІНА'){
```

```
      const [ messages, chats ] = await dbRequest.GetMessageIDsLiveSupport(new Ob-
jectId(user['operatorObjectCloseLiveSupport']));

      for(let n = 0; n < messages.length; n++){
        await ctx.telegram.editMessageReplyMarkup(chats[n], messages[n], undefined,
Markup.inlineKeyboard(liveKeyboard(ctx?.chat?.id ?? -1, 'declined', user['operatorOb-
jectCloseLiveSupport'])).reply_markup)
      }

      ctx.reply('Прекрасно, тепер можете відпочивати', {
        reply_markup: {
          one_time_keyboard: true,
          keyboard: keyboards.endRootMenu()
        }
      })

      ctx.telegram.sendMessage(user['activeUserLiveSupport'], "Оператор закрив канал,
сподіваємося ваше питання було вирішено.", {
        reply_markup: {
          one_time_keyboard: true,
          keyboard: keyboards.endRootMenu()
        }
      })

      await dbRequest.ChangeAvaibiltyForOperator(ctx?.chat?.id ?? -1, true);

      await db.set(parseInt(user['activeUserLiveSupport']))('state')('EndFunctionMan-
ager')
      await set('state')('EndFunctionManager');
    }
    else{
      switch(true){
        case CheckException.TextException(data):
          ctx.telegram.sendMessage(user['activeUserLiveSupport'], data.text, {
            reply_markup: {
              one_time_keyboard: true,
              keyboard: keyboards.liveSupportProbablyCancel()
            }
          })
          break;

        case CheckException.FileException(data):
          ctx.telegram.sendDocument(user['activeUserLiveSupport'], data.file, {
            reply_markup: {
              one_time_keyboard: true,
              keyboard: keyboards.liveSupportProbablyCancel()
            }
          })
          break;
```

```
        case CheckException.PhotoException(data):
          ctx.telegram.sendPhoto(user['activeUserLiveSupport'], data.photo, {
            reply_markup: {
              one_time_keyboard: true,
              keyboard: keyboards.liveSupportProbablyCancel()
            }
          })
          break;

        case CheckException.LocationException(data):
          ctx.telegram.sendLocation(user['activeUserLiveSupport'], data.location[0],
data.location[1], {
            reply_markup: {
              one_time_keyboard: true,
              keyboard: keyboards.liveSupportProbablyCancel()
            }
          })
          break;

        case CheckException.PhoneException(data):
          ctx.telegram.sendContact(user['activeUserLiveSupport'], data.phone_number,
user['name'], {
            reply_markup: {
              one_time_keyboard: true,
              keyboard: keyboards.liveSupportProbablyCancel()
            }
          })
          break;

        case CheckException.PollsException(data):
          ctx.telegram.sendMessage(user['activeUserLiveSupport'], "Користувач
надіслав тип повідомлення Polls (котрий не підтримується)", {
            reply_markup: {
              one_time_keyboard: true,
              keyboard: keyboards.liveSupportProbablyCancel()
            }
          })

          ctx.reply("Вибачте, але такий тип повідомлення у нас не підтримується.", {
            reply_markup: {
              one_time_keyboard: true,
              keyboard: keyboards.liveSupportProbablyCancel()
            }
          })
          break;

        case CheckException.StickerException(data):
          ctx.telegram.sendSticker(user['activeUserLiveSupport'], data.stickers, {
            reply_markup: {
              one_time_keyboard: true,
```

```
            keyboard: keyboards.liveSupportProbablyCancel()
          }
        })
        break;

      case CheckException.VideoException(data):
        ctx.telegram.sendVideo(user['activeUserLiveSupport'], data.video, {
          reply_markup: {
            one_time_keyboard: true,
            keyboard: keyboards.liveSupportProbablyCancel()
          }
        })
        break;

      case CheckException.AudioException(data):
        ctx.telegram.sendAudio(user['activeUserLiveSupport'], data.audio, {
          reply_markup: {
            one_time_keyboard: true,
            keyboard: keyboards.liveSupportProbablyCancel()
          }
        })
        break;

      case CheckException.VoiceException(data):
        ctx.telegram.sendVoice(user['activeUserLiveSupport'], data.voice, {
          reply_markup: {
            one_time_keyboard: true,
            keyboard: keyboards.liveSupportProbablyCancel()
          }
        })
        break;

      case CheckException.VideoNoteException(data):
        ctx.telegram.sendAudio(user['activeUserLiveSupport'], data.video_circle, {
          reply_markup: {
            one_time_keyboard: true,
            keyboard: keyboards.liveSupportProbablyCancel()
          }
        })
        break;

      default:
        ctx.reply("Вибачте, але таке не підтримується", {
          reply_markup: {
            one_time_keyboard: true,
            keyboard: keyboards.liveSupportProbablyCancel()
          }
        });
```

```
          ctx.telegram.sendMessage(user['activeUserLiveSupport'], "Користувач
надіслав непідтримуваний тип повідомлення.", {
            reply_markup: {
              one_time_keyboard: true,
              keyboard: keyboards.liveSupportProbablyCancel()
            }
          })
          break;
      }
    }
  })

  bot.action(/^acceptSupport:(\d+),(.+)$/, async (ctx) => {
    const id = Number.parseInt(ctx.match[1]),
      object = ctx.match[2],
      [ messages, chats ] = await dbRequest.GetMessageIDsLiveSupport(new ObjectId(ob-
ject));
    let operator: string | undefined = '';

    try {
      for(let n = 0; n < messages.length; n++){
        if (messages[n] === ctx.callbackQuery.message?.message_id){
          await ctx.editMessageReplyMarkup(Markup.inlineKeyboard(liveKeyboard(id,
'accepted', ctx.match[2])).reply_markup);
          operator = await db.get(chats[n])('name');
          await db.set(id)('activeHelperLiveSupport')(chats[n]);
          await db.set(id)('userObjectCloseLiveSupport')(object);
          await db.set(chats[n])('operatorObjectCloseLiveSupport')(object);
          await db.set(chats[n])('activeUserLiveSupport')(id.toString());
          await db.set(chats[n])('state')('OperatorLiveSupportHandler');
          await db.set(id)('state')('UserLiveSupportHandler');
          await dbRequest.ChangeAvaibiltyForOperator(chats[n], false);
        }
        else await ctx.telegram.editMessageReplyMarkup(chats[n], messages[n], unde-
fined, Markup.inlineKeyboard(liveKeyboard(id, 'busy', ctx.match[2])).reply_markup)
      }
      ctx.telegram.sendMessage(id, `Ваш запит прийняв оператор ${operator}`, {
        reply_markup: {
          one_time_keyboard: true,
          keyboard: keyboards.liveSupportProbablyCancel()
        }
      });
      ctx.reply('Ви успішно прийняли запит користувача, можете працювати', {
        reply_markup: {
          one_time_keyboard: true,
          keyboard: keyboards.liveSupportProbablyCancel()
        }
      })
    } catch (e) {
      console.log(e);
```

```
    }

    return ctx.answerCbQuery(`Ви успішно взяли замовлення`);
  });

  bot.action(/^acceptedCheck$/, (ctx) => {
    return ctx.answerCbQuery(`Ви вже прийняли цього користувача.`);
  });

  bot.action(/^busyCheck$/, (ctx) => {
    return ctx.answerCbQuery(`Цього користувача прийняв ішний оператор.`);
  });

  bot.action(/^declinedCheck$/, (ctx) => {
    return ctx.answerCbQuery(`Канал вже закритий, не актуально.`);
  });

  bot.action(/^errorCheck$/, (ctx) => {
    return ctx.answerCbQuery(`Помилка, повідомте підтримку.`);
  });

  bot.launch();
}

main();
```

check.ts

```
class StandartCheckException{
    public VideoException(data: { phone_number: string; text: string, photo: string,
file: string, stickers: string, video: string, location: number[], polls: string,
voice: string, audio: string, video_circle: string }): boolean {
```

```
      if (data.phone_number === '' && data.photo === '' &&
      data.text === '' && data.file === '' &&
      data.stickers === '' && data.location[0] === -1 &&
      data.polls === '' && data.video !== '' &&
      data.voice === '' && data.audio === '' && data.video_circle === ''){
        return true;
      }
      else return false;
    }

    public PhotoException(data: { phone_number: string; text: string, photo: string,
file: string, stickers: string, video: string, location: number[], polls: string,
voice: string, audio: string, video_circle: string }): boolean {
      if (data.phone_number === '' && data.text === '' &&
      data.file === '' && data.stickers === '' &&
      data.video === '' && data.polls === '' &&
      data.location[0] === -1 && data.photo !== '' &&
      data.voice === '' && data.audio === '' && data.video_circle === ''){
        return true;
      }
      else return false;
    }

    public FileException(data: { phone_number: string; text: string, photo: string,
file: string, stickers: string, video: string, location: number[], polls: string,
voice: string, audio: string, video_circle: string }): boolean {
      if (data.phone_number === '' && data.photo === '' &&
      data.text === '' && data.stickers === '' &&
      data.video === '' && data.location[0] === -1 &&
      data.polls === '' && data.file !== '' &&
      data.voice === '' && data.audio === '' && data.video_circle === ''){
        return true;
      }
      else return false;
    }

    public TextException(data: { phone_number: string; text: string, photo: string,
file: string, stickers: string, video: string, location: number[], polls: string,
voice: string, audio: string, video_circle: string }): boolean {
      if (data.phone_number === '' && data.photo === '' &&
      data.file === '' && data.stickers === '' &&
      data.video === '' && data.location[0] === -1 &&
      data.polls === '' && data.text !== '' &&
      data.voice === '' && data.audio === '' && data.video_circle === ''){
        return true;
      }
      else return false;
    }
```

```
    public StickerException(data: { phone_number: string; text: string, photo:
string, file: string, stickers: string, video: string, location: number[], polls:
string, voice: string, audio: string, video_circle: string }): boolean {
      if (data.phone_number === '' && data.photo === '' &&
      data.text === '' && data.file === '' &&
      data.video === '' && data.location[0] === -1 &&
      data.polls === '' && data.stickers !== '' &&
      data.voice === '' && data.audio === '' && data.video_circle === ''){
        return true;
      }
      else return false;
    }

    public LocationException(data: { phone_number: string; text: string, photo:
string, file: string, stickers: string, video: string, location: number[], polls:
string, voice: string, audio: string, video_circle: string }): boolean {
      if (data.phone_number === '' && data.photo === '' &&
      data.text === '' && data.file === '' &&
      data.video === '' && data.stickers === '' &&
      data.polls === '' && data.location[0] !== -1 &&
      data.voice === '' && data.audio === '' && data.video_circle === ''){
        return true;
      }
      else return false;
    }

    public PollsException(data: { phone_number: string; text: string, photo: string,
file: string, stickers: string, video: string, location: number[], polls: string,
voice: string, audio: string, video_circle: string }): boolean {
      if (data.phone_number === '' && data.photo === '' &&
      data.text === '' && data.file === '' &&
      data.video === '' && data.location[0] === -1 &&
      data.stickers === '' && data.polls !== '' &&
      data.voice === '' && data.audio === '' && data.video_circle === ''){
        return true;
      }
      else return false;
    }

    public PhoneException(data: { phone_number: string; text: string, photo: string,
file: string, stickers: string, video: string, location: number[], polls: string,
voice: string, audio: string, video_circle: string }): boolean {
      if (data.phone_number !== '' && data.photo === '' &&
      data.text === '' && data.file === '' &&
      data.video === '' && data.location[0] === -1 &&
      data.stickers === '' && data.polls === '' &&
      data.voice === '' && data.audio === '' && data.video_circle === ''){
        return true;
      }
      else return false;
```

```
    }

    public VoiceException(data: { phone_number: string; text: string, photo: string,
file: string, stickers: string, video: string, location: number[], polls: string,
voice: string, audio: string, video_circle: string }): boolean {
      if (data.phone_number === '' && data.photo === '' &&
      data.text === '' && data.file === '' &&
      data.video === '' && data.location[0] === -1 &&
      data.stickers === '' && data.polls === '' &&
      data.voice !== '' && data.audio === '' && data.video_circle === ''){
        return true;
      }
      else return false;
    }

    public VideoNoteException(data: { phone_number: string; text: string, photo:
string, file: string, stickers: string, video: string, location: number[], polls:
string, voice: string, audio: string, video_circle: string }): boolean {
      if (data.phone_number === '' && data.photo === '' &&
      data.text === '' && data.file === '' &&
      data.video === '' && data.location[0] === -1 &&
      data.stickers === '' && data.polls === '' &&
      data.voice === '' && data.audio === '' && data.video_circle !== ''){
        return true;
      }
      else return false;
    }

    public AudioException(data: { phone_number: string; text: string, photo: string,
file: string, stickers: string, video: string, location: number[], polls: string,
voice: string, audio: string, video_circle: string }): boolean {
      if (data.phone_number === '' && data.photo === '' &&
      data.text === '' && data.file === '' &&
      data.video === '' && data.location[0] === -1 &&
      data.stickers === '' && data.polls === '' &&
      data.voice === '' && data.audio !== '' && data.video_circle === ''){
        return true;
      }
      else return false;
    }
}

export const CheckException : StandartCheckException = new StandartCheckException();
```

livekeyboard.ts

```
import { Markup } from "telegraf";
type HideableIKBtn = ReturnType<typeof Markup.button.callback>;
```

```
export const liveKeyboard = (id: number, processStatus: string, oid: string): Hidea-
bleIKBtn[][] => {
    switch(processStatus){
        case "waiting":
            return [
                [
                    Markup.button.callback("✔ Прийняти", `acceptSup-
port:${id},${oid}`)
                ]
            ];

        case "accepted":
            return [
                [
                    Markup.button.callback("🟢 В процесі", `acceptedCheck`)
                ]
            ];

        case "busy":
            return [
                [
                    Markup.button.callback("🟢 Прийнято іншим оператором",
`busyCheck`)
                ]
            ];

        case "declined":
            return [
                [
                    Markup.button.callback("❌ Канал закритий", `declinedCheck`)
                ]
            ];

        default:
            return [
                [
                    Markup.button.callback("??_Помилка_створення_кнопки_??", `er-
rorCheck`)
                ]
            ];
    }
};
```

script.ts

```
const script = {

  entire: {
```

```
    greeting: `Привіт☺ Вас вітає бот банку "ПриватБанк". Підкажіть, будь ласка, як
можна до Вас звертатися?`,
    thanksOnStart: (name : string) => `Дуже приємно, ${name}😁`,
    phoneRequire: `Тепер, будь ласка, нам потрібен ваш номер телефону, щоб ми могли
індетифікувати вас і вашу карту`,
    thanksForComplet: (card : number) => `Дуже вдячні Вам 🤍\n\nВаш номер картки -
${card}.`,
    functionEntire: `Тепер, будь ласка, оберіть функцію якою б хотіли скористатися`
  },

  error: {
    errorExceptionFunction: `Ой, халепа... Щось пішло не так...😬\nСпробуйте, будь
ласка, скористатися кнопками нижче для вибору функції! ⬇`,
    buttonError: `От халепа... \n\nБудь ласка, оберіть кнопку нижче ⬇`,
    classicTextError: `Ого-го... щиро дякуємо, але вам потрібно просто написати,
нічого більше :)`,
    paymentError: `Ой.. лишенько.. \n\nВам потрібно завантажити скрін з галереї або
надіслати файл (підтримовані формати: pdf, jpeg, jpg, png, heic)`,
    phoneError: `Ой-ой... вам потрібно натиснути внизу на кнопку “Поділитися
номером”`
  },

  values: {
    entire: `Чудесно, тепер оберіть конекретніше, що вам саме потрібно`,

    exchangeAllLoad: {
      state0: "Почекайте будь ласка, завантаження курсів валют...",
      state1: "Почекайте будь ласка, завантажуємо ще більше курсів...",
      state2: "Зачекайте, будь ласка, ще трохи, завантажуємо ще більше курсів...",
      state3: "Знову ж таки, зачекайте хвильку, завантажуємо ще більше курсів...",
      state4: "Мить і все буде готово, завантажуємо ще більше курсів...",
      state5: "І останній раз... завантажуємо ще більше курсів..."
    },

    chooseSpecificExchangeRate: `Виберіть, будь ласка, валюту ⤵`,

    requestNumberToConvert: `Яку суму потрібно конвертувати?`,

    valueData: (name: string, value: string, code: string, number: number) =>
    number < 1 ? `<b>${name}</b> - ${value} ${code.toLocaleLowerCase()}.` :
`|\n|\n<b>${name}</b> - ${value} ${code.toLocaleLowerCase()}.`,

    customValueData: (name: string, nonprocessed: string, value: number, code:
string) =>
    `<b>${nonprocessed}</b> uah. в <b>${name}</b> становить <b>${value}</b>
${code.toLocaleLowerCase()}.`
  },

  balanceAndHistory: {
    showData: (author: string, date: string, type: string, text: string) =>
```

```
    `📥<b>${author}</b>\n\n📨${type === 'outgoing' ? (text === 'fail' ? '' : '-') :
'+'}${text === 'fail' ? "<b>Недостатньо коштів для транзакції</b>" : `<b>${text}</b>
uah.`}\n\n\n📟 ${date} 📟`,

    showErrorToShowData: `У вас відсутні транзакції.`,

    showActualCardBalance: (balance: number) => `Баланс вашої картки - ${balance}
uah.`
  },

  qA: {
    chooseQuestion: `Чудово! Яке питання вас цікавить?`
  },

  settings: {
    open: `Меню налаштувань.\n\nВиберіть, будь ласка, знизу кнопочку, параметр, який
потрібно змінити`,

    values: `Виберіть валюту, котра буде основною.\n\nЗА ЗАМОВЧУВАННЯМ "Українські
гривні (UAH)"`,

    changed: (rate: string) => `Успішно змінено на ${rate}`
  },

  liveSupport: {
    userRequest: (name: string, telegram: string, phone: string, dateRequest: string)
=> `Новий запит на підтримку!\n\nІм'я - ${name} (@${telegram})\nТелефон -
${phone}\n\nДата заявки - ${dateRequest}`,
    userRespond: `Ваш запит прийнято, очікуйте на оператора`
  }

}

export default script;
```