ДИПЛОМНА РОБОТА

ДР.ПІс – 09.00.000 ПЗ

Група ПІс-2015

Кріслатий В.В.

2019

ПВНЗ Університет Короля Данила

Кафедра Інформаційних технологій та програмної інженерії

УДК 004

ДИПЛОМНАРОБОТА

Тема ***Розробка програми Telegram-бота для конвертації валют.***

Напрям підготовки ***6.050103 «Програмна інженерія»***

(код і назва спеціальності)

ПОЯСНЮВАЛЬНАЗАПИСКА

ДР.ПІс–09.00.000 ПЗ

(позначення)

Студент

Крісватий В.В..

(підпис) (дата) (розшифрування підпису)

Керівник проекту

к.т.н. Пашкевич О.П.

(посада) (підпис) (дата) (розшифрування підпису)

Нормоконтроль

к.т.н. Мануляк І.З.

(посада) (підпис) (дата) (розшифрування підпису)

Допускається до захисту

Завідувач кафедри

д.т.н.,доц. Мельничук С.І.

(посада) (підпис) (дата) (розшифрування підпису)

2019

ПВНЗ Університет Короля Данила

Факультет *Інформаційних технологій*
Кафедра *Інформаційних технологій та програмної інженерії*
Спеціальність *121 «Інженерія програмного забезпечення»*

ЗАТВЕРДЖУЮ:
Завідувач кафедри ІТПІ
О.П.Пашкевич
“____”_____2019р.

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ)

Студенту *Кріслатому Владиславу Володимировичу*

1.Темапроекту(роботи) *Розробка телеграм бота для виведення поточного курсу валют*

Затверджена наказом ректора ПВНЗ Університету Короля Данила від *15.11.19 16/2-НВ*

2.Терміндачістудентомзакінченогопроекту(роботи)_____
3.Вихідні дані до проекту(роботи) *10.06.2019*

4.Зміст пояснювальної записки (перелік питань, що їх належить розробити)
1. Огляд існуючих існуючих телеграм ботів 2. Технології розроблення телеграм бота 3. Back-end частина проекту 4. Економічне обґрунтування розробки проекту 5. Охорона праці та безпеки в надзвичайних ситуаціях.

5.Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
1.Чат боти 2. Аналізовані боти 3. Privatbank_help_bot 4. Телеграм 5. Сервіси 6. C# 7. Структура 8. Реєстрування чат боту 9. Приклад роботи програми

6. Консультанти з проекту (роботи), із зазначенням розділів проекту, що стосуються

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 15.11.2019

Керівник _____ Пашкевич О.П.

Завдання прийняв до виконання _____ Кріслатий В.В.

КАЛЕНДАРНИЙ ПЛАН

Пор№	Назва етапів дипломного проекту(роботи)	Термін виконання етапів проекту(роботи)	Примітка
1.	Огляд існуючих веб-сайтів наукових організвцій України	25.03.2019	
2.	Технології розробки back-end частини проекту	15.04.2019	
3.	Back-end частина проекту	15.05.2019	
4.	Економічне обгрунтування розробки проекту	23.05.2019	
5.	Охорона праці	29.06.2019	
6.	Оформлення пояснювальної записки	06.06.2019	
7.	Оформлення графічного матеріалу та підготовка до захисту роботи	15.06.2019	

Студент-дипломник _____ *Кріслатий В.В.*
(підпис)(розшифровка підпису)

Керівник проекту _____ *Пашкевич О.П.*
(підпис)(розшифровка підпису)

АНОТАЦІЯ

Метою завдання було розробити телеграм бота, який буде виводити поточний курс валют.

Для досягнення мети було використано технології C#, ASP.NET.

Отриманим результатом є робочий бот який показує курси валют у реальному часі, та повністю виконує поставлені завдання.

Виконано необхідні техніко-економічні розрахунки.

Галузевою сферою застосування, є застосування як інтернет ресурсу.

SUMMARY

The purpose of the task was to develop a telegram of the boat, which will display the current exchange rate.

To achieve the goal, C#, ASP.NET technology was used. The resulting result is a working bot that displays real-time currency rates, and fully performs tasks.

The necessary technical and economic calculations are executed.

The sectoral scope is the application as an online resource.

РЕФЕРАТ

Розрахунково-пояснювальна записка:

- 60 сторінок
- 20 рисунків
- 3 таблиці
- 5 додатків
- 11 посилань.

Об’єктом розробки є використання технології C# .NET для розробки Телеграм бота .

Об’єктом досліджень є 4 телеграм боти, які мають похожий функціонал по перегляду курсів валют

Метою розробки є написанння телеграм бота, який буде показувати поточний курс валют використовуючи 3 різних зовнішніх джерела.

В першому розділі описані чатботи які були вивчені як і їх функціонал і дана об’єктивна оцінка

Другий розділ описує стек технологій і архітектуру данного дипломного проекту

Третій розділ описує функціонал чатботу, відправлення повідомлень в телеграм і зв’язок з сторонніми джерелами, з яких витягуються поточні курси валют

ЗМІСТ

ВСТУП

Інформаційні технології є найбільш важливою складовою процесу використання інформаційних ресурсів суспільства. До сьогодні вони пройшли кілька еволюційних етапів, зміна яких визначалася головним чином розвитком науково-технічного прогресу, появою нових технічних засобів обробки, пошуку та поширення інформації. У сучасному суспільстві основним засобом поширення інформації являється інтернет. В повсякденній мові слово Інтернет найчастіше вживається в значенні Всесвітнього павутиння і доступної в ньому інформації, а не у значенні самої фізичної мережі. Також вживаються терміни «Всесвітня мережа», «Глобальна мережа» чи навіть одне слово «Мережа», «Інéт», «Тенета», «Міжмережжя», «Інтернéтрі» або «Нéтрі». Все частіше Інтернет вживається і з малої літери, що можна пояснити паралелями з термінами «радіо», «телебачення», які пишуть з малої.

INTERNET надає такі основні види послуг:

- e-mail — електронна пошта;
- групи новин;
- списки поштової розсилки;
- доступ до файлів віддалених комп'ютерів;
- сеанси зв'язку з іншими комп'ютерами, під'єднаними до INTERNET;
- пошук інформації в базі даних в оперативному режимі;
- спілкування з іншими користувачами шляхом використання сервісу Internet Relay Chart;
- доступ до інформаційної системи World Wide Web (WWW).

З додаткових послуг можна виділити наступні:

- широка передача MultiMedia;
- RadioInternet;
- розмовний конференційний зв'язок;
- безпечні угоди;

– відеоконференційний зв'язок;
– безпровідне з'єднання.

В даний час WEB використовується у всі сфери діяльності, починаючи від розваг (фільми, відеоігри, музика і т.д.) і закінчуєчи науковою інформацією (десертації, монографії, нові відкриття в галузі науки і т.д.). Голобальна мережева використовується для навчання, спілкування, розваг та розміщення різного роду інформації(наукових статей, реклами, вакансій, новин політики та спорту). Також інтернет дозволив значно полегшити передачу та поширення інформації між людьми, компаніями та країнами за допомогою різних сервісів.

Мета даного дипломного проекту полягає у створенні веб-сайту, який надаватиме можливість розміщення інформації про майбутні конференції, матеріали завершених конференцій, розміщення наукових матеріалів (монографій), новинок та здобутків України в технічній галузі.

1 ОГЛЯД ІСНУЮЧИХ ЧАТБОТІВ

1.1 Аналіз чатботу @moneyexcange

Для розробки використана мова програмування JS. Даний чатбот містить такий функціонал:

– перегляд курсу валют;

– конвертування відповідно до рубля.

JavaScript (JS) — динамічна, об'єктно-орієнтована прототипна мова програмування. Реалізація стандарту ECMAScript. Найчастіше використовується для створення сценаріїв веб-сторінок, що надає можливість на стороні клієнта (пристрої кінцевого користувача) взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером, змінювати структуру та зовнішній вигляд веб-сторінки.

JavaScript містить декілька вбудованих об'єктів: Global, Object, Error, Function, Array, String, Boolean, Number, Math, Date, RegExp. Крім того, JavaScript містить набір вбудованих операцій, які, грубо кажучи, не обов'язково є функціями або методами, а також набір вбудованих операторів, що управляють логікою виконання програм. Синтаксис JavaScript в основному відповідає синтаксису мови Java (тобто, зрештою, успадкований від C), але спрощений порівняно з ним, щоб зробити мову сценаріїв легкою для вивчення. Так, приміром, декларація змінної не містить її типу, властивості також не мають типів, а декларація функції може стояти в тексті програми після неї.

1.2 Аналіз чатботу @Sberometr

Отримувати інформацію Сберометра в Telegram можна так:

– Канал з повідомленнями про зміну курсу валюти;

– Канал з валютними новинами;

– Бот.

Канал з повідомленнями про зміну курсу валюти (Sberometer Currency)

Посилання на канал: https://telegram.me/sberometer_kurs

Що розсилаємо:

– офіційні курси долара США і євро (приблизно в 11-40 МСК);

– повідомлення в разі сильних змін курсу долара (зростання або падіння).

Канал з валютними новинами (Sberometer News)

Посилання на канал: https://telegram.me/sberometer

Що розсилаємо: Найбільш важливі новини, пов'язані з валютою і заощадженнями (5-10 новин в день).

бот @SberometerBot

Наш бот для Telegram**,** вміє (таблиця 1.1– 1.2):

– повідомляти за запитом поточний і офіційний курс долара і євро, а також офіційні курси всіх котируються Центробанком валют;

– видавати поточну (біржову) ціну нафти і золота;

– перераховувати валюту в рублі;

– видавати за запитом останні валютні новини, новини про нафту, золото, політиці.

Таблиця 1.1 – Команди чатбота

Kurskурс валюта (Або просто "до", або латинські "k" або "c")	Отримання офіційного статусу біржового курсу євро і долара	Біржовий курс на 18-45 мск 02.10.2015: Долар США - 66,4 руб. Євро - 74,82 руб. Офіційний курс ЦБ РФ на 03.10.2015: Долар США - 65,9414 руб. Євро - 73,6302 руб.

Продовження таблиці 1.1

USD (або $, долар, долар) EUR (або €, євро) GBP (або £, фунт) AUD AZN BGN ... і т.д. - код валюти згідно	Видає останній офіційний курс валюти, встановлений Центральним банком РФ. Для USD і EUR видає не тільки офіційний, але і біржовий (тобто найактуальніший) курс.	Офіційний USD на 03.10.2015: 65,9414 руб. Біржовий курс USD на 19-15 мск 02.10.2015: 64,16 руб.
10 USD (або "10 $") 30,4 PLN	Перерахунок іноземної валюти в рублі. Для долара і євро повертає перерахунок за офіційним і біржовим курсом, для решта валют - тільки за офіційним.	10 USD = 659,41 руб. (За офіційним курсом) = 641,6 руб. (За біржовим курсом)
advice buy USD (або "рада купити $") advice sell USD (або "рада продати $") advice buy EUR (або "рада купити євро") advice sell EUR (або "рада продати євро")	Видає рада щодо купівлі або продажу валюти в даний момент часу (тобто з урахуванням того, як сильно і давно змінився курс, чи має сенс бігти до обмінника).	Приклад рада см. На головній сторінці Сберометра (кнопка "дайте совет!").
news новини (або просто "н")	Надсилає 5 найпопулярніших сьогодні новин.	15:44 Аналітики Sberbank CIB не виключають нового стрибка курсу долара. На графіку USD / RUB з липня чітко сформувалася фігура «трикутник», є ймовірність різкого прориву - швидше за все, вгору.

Таблиця 1.2 – Список доступних валют

Валюта	**Команди отримання курсу**
Австралійський долар	AUD
Азербайджанський манат	AZN
Вірменський драм	AMD
Фунт стерлінгів GB	GBP
болгарський лев	BGN
Бразильський реал	BRL
угорський форинт	HUF
Он Республіки Корея	KRW
Датська крона	DKK
Долар США	USD
ЄВРО	EUR
Індійський рупій	INR
казахський тенге	KZT
Канадський долар	CAD
киргизький сом	KGS
Молдовський лей	MDL
Новий румунський лей	RON
Новий туркменський манат	TMT
Норвезька крона	NOK
польський злотий	PLN
З ІН	XDR
Сінгапурський долар	SGD
таджицький сомоні	TJS
турецька ліра	TRY
узбецький сум	UZS
Українська гривня	UAH

PHP (англ. PHP: Hypertext Preprocessor — PHP: гіпертекстовий препроцесор), попередня назва: Personal Home Page Tools — скриптова мова програмування, була створена для генерації HTML-сторінок на стороні веб-сервера. PHP є однією з найпоширеніших мов, що використовуються у сфері веб-розробок (разом із Java, .NET, Perl, Python, Ruby). PHP підтримується переважною більшістю хостинг-провайдерів. PHP — проект відкритого програмного забезпечення.

PHP інтерпретується веб-сервером у HTML-код, який передається на сторону клієнта. На відміну від скриптової мови JavaScript, користувач не бачить PHP-коду, бо браузер отримує готовий html-код. Це є перевага з точки зору безпеки, але погіршує інтерактивність сторінок. Але ніхто не забороняє використовувати PHP для генерування JavaScript-кодів, які виконуються вже на стороні клієнта.

Преваги:

- велика різноманітність функцій PHP дають можливість уникнути написання багаторядкових призначених для користувача функцій на C або Pascal;
- наявність інтерфейсів до багатьох баз даних в PHP вбудовані бібліотеки для роботи з MySQL, PostgreSQL, mSQL, Oracle, dbm, Hyperware, Informix, InterBase, Sybase;
- Open Source

Недоліки:

- незручність дизайну мови
- змінні з символом «$»
- складні назви поширених функцій (html_entities_decode, mysql_select_db, nl2br тощо)
- не підтримується Unicode в версіях до 6.0
- непередбачуваність нових версій PHP.

Laravel — безкоштовний, з відкритим кодом PHP-фреймворк, створений Taylor Otwell і призначений для розробки веб-додатків відповідно до шаблону model–view–controller (MVC). Деякі з особливостей Laravel є модульна система упакування з виділеним менеджером залежностей, різні способи для доступу до реляційних баз даних, утиліти, які допомагають в розгортанні додатків і технічного обслуговування, а також його орієнтація на синтаксичний цукор.

Станом на березень 2015 року, Laravel вважається одним з найпопулярніших PHP фреймворком, разом з Symfony2, Nette, CodeIgniter, Yii2 й іншими фреймворками.

Сирцевий код Laravel'a розміщується на GitHub і ліцензований відповідно до умов MIT License.

Yii (вимовляється як «Ї» або [ji:]) — це високопродуктивний веб-фреймворк, написаний на PHP, реалізує парадигму модель-вид-контролер. Yii — скорочення від «Yes It Is!»

За результатами тестів phpmark Yii показав найкращу продуктивність.

Справедливості заради, варто відзначити, що продуктивність фреймворків в цих тестах оцінювалася на штучних прикладах типу Hello world. Тести показують час ініціалізації фреймворків, і на їх підставі можна лише зробити висновок, що Yii має якісну підсистему відкладеної ініціалізації (тобто, код завантажується лише тоді, коли він необхідний). Підтвердженої інформації про те, що Yii в «бойових умовах» працює швидше, ніж інші фреймворки, немає.

Symfony — відкритий PHP-фреймворк, що реалізує концепцію модель-вид-контролер (MVC) та автоматизовує найзагальніші веб-задачі, являє собою широконалаштовну систему пов'язаних класів і призначений для розробки та керування веб-застосунками. Випускається під MIT ліцензією. Symfony є вільним програмним забезпеченням. Symfony спрямований на прискорення створення та підтримки веб-застосунків, а також для уникнення витрат часу для розв'язування тривіальних задач у розробці (наприклад, написання валідаторів форм).

1.3 Аналіз чат боту @minfinbot

Для розробки чат боту @minfinbot використано мову програмування python та невідовий додатковий фреймворк. Бот показує курси валют щодо гривні: НБУ,міжбанк, середній курс в банках і на ринку.(рис 1.2) Особливість @MinfinBot в тому, що він вміє сортувати банки в залежності від вартості долара, євро і рубля.

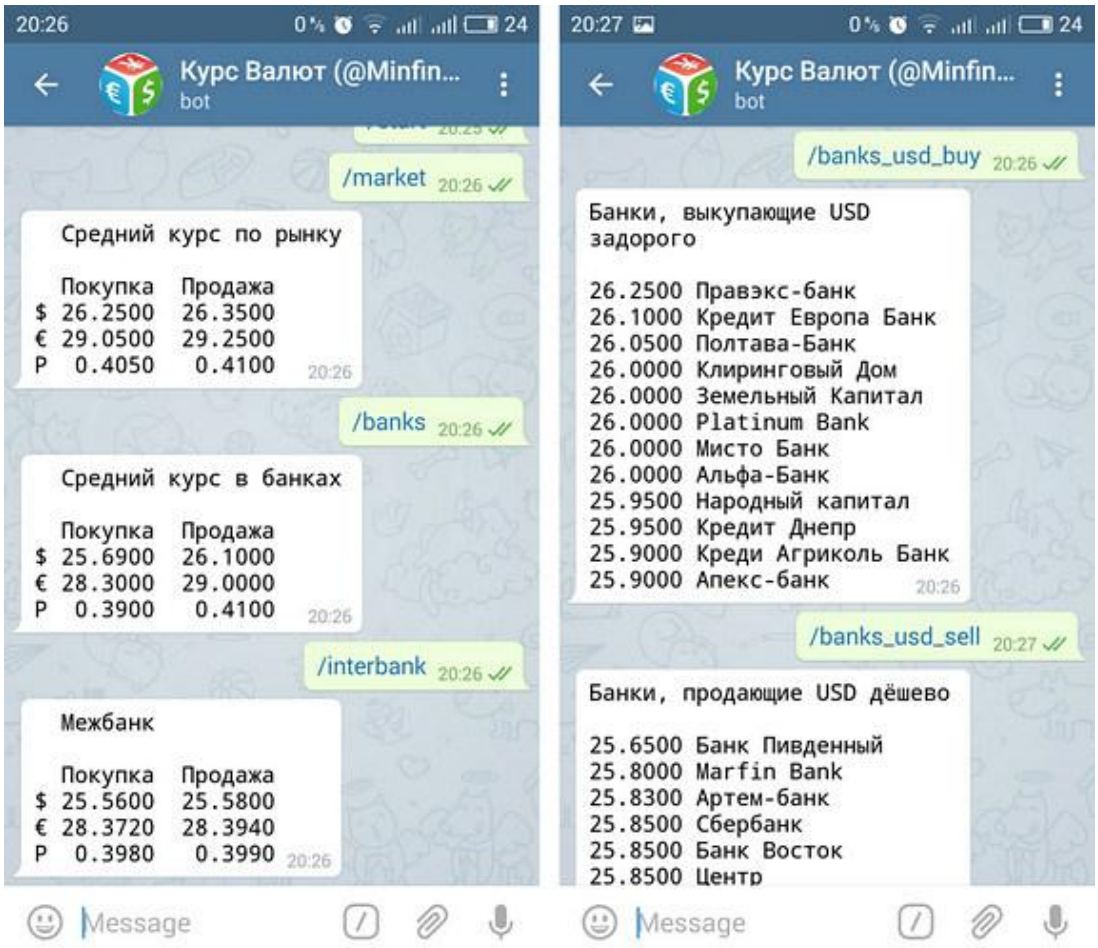

Рисунок 1.2 – Чат бот @minfinbot

Python (найчастіше вживане прочитання — «Па́йтон», запозичено назву з британського шоу Монті Пайтон) — інтерпретована об'єктно-орієнтована мова програмування високого рівня з строгою динамічною типізацією. Розроблена в 1990 році Гвідо ван Россумом. Структури даних високого рівня разом із динамічною семантикою та динамічним зв'язуванням роблять її привабливою для швидкої розробки програм, а також як засіб поєднання існуючих компонентів.

Python підтримує модулі та пакети модулів, що сприяє модульності та повторному використанню коду. Інтерпретатор Python та стандартні бібліотеки доступні як у скомпільованій так і у вихідній формі на всіх основних платформах. В мові програмування Python підтримується кілька парадигм програмування, зокрема: об'єктно-орієнтована, процедурна, функціональна та аспектно-орієнтована.

1.4 Аналіз чатботу Privatbank_help_bot

Данний чатбот розробив український банк Приватбанк. (рис 1.3) Його можна знайти по посиланню, яке вказане на офіційному сайті, або в інших сайтах, які рекламують чатботів. При входженні у телеграм і виклику його із списку контактів, ми побачимо таке меню:

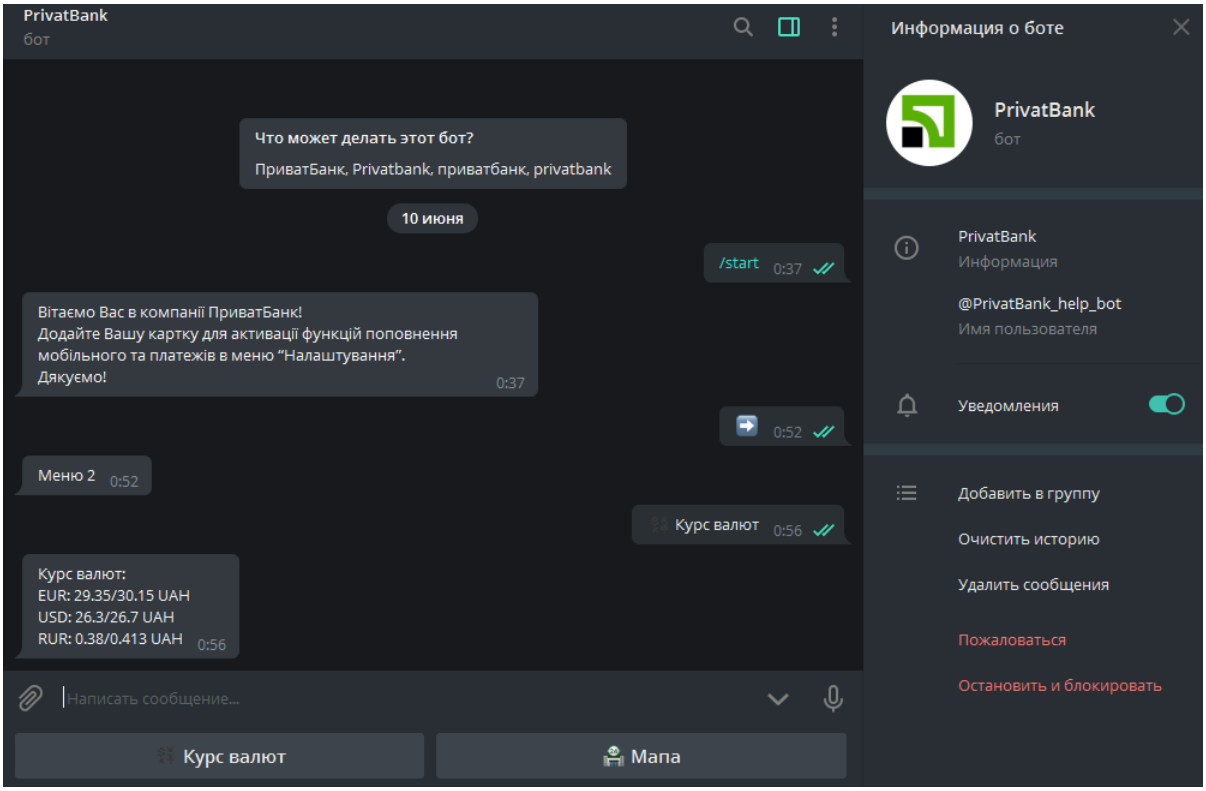

Рисунок 1.3 – Вікно роботи чатбота приватбанку

В данному вікні є доступні багато варіантів меню, але те яке нас цікавить знаходиться в другій частині, тому потрібно натиснути кнопку «стрілочку» що б

перейти до наступного меню. В наступному меню є кнопка курсу валют, яка виводить поточний курс валют по приватбанку.

Чатбот написаний за допомгою сервісу aimylogic який дозволяє створювати чатботів без особливих навичок. Для його створення достатьо володіти потрібним рівнем англійської мови.

Aimylogic: Підтримувані платформи: месенджери, додатки та сайти. Вартість: безкоштовно і від 2900 гривень в місяць Мова інтерфейсу: англійська.

Створеного чат-бота можна вбудувати в месенджери, соціальні мережі, сайти і в голосових помічників - конструктор працює з «Алісою» від «Яндекса», Google Assistant і Alexa.

Чат-бот Aimylogic (рис 1.4) працює з природною мовою. Бот визначають наміри, використовує webhooks для здійснення транзакцій і спілкування.

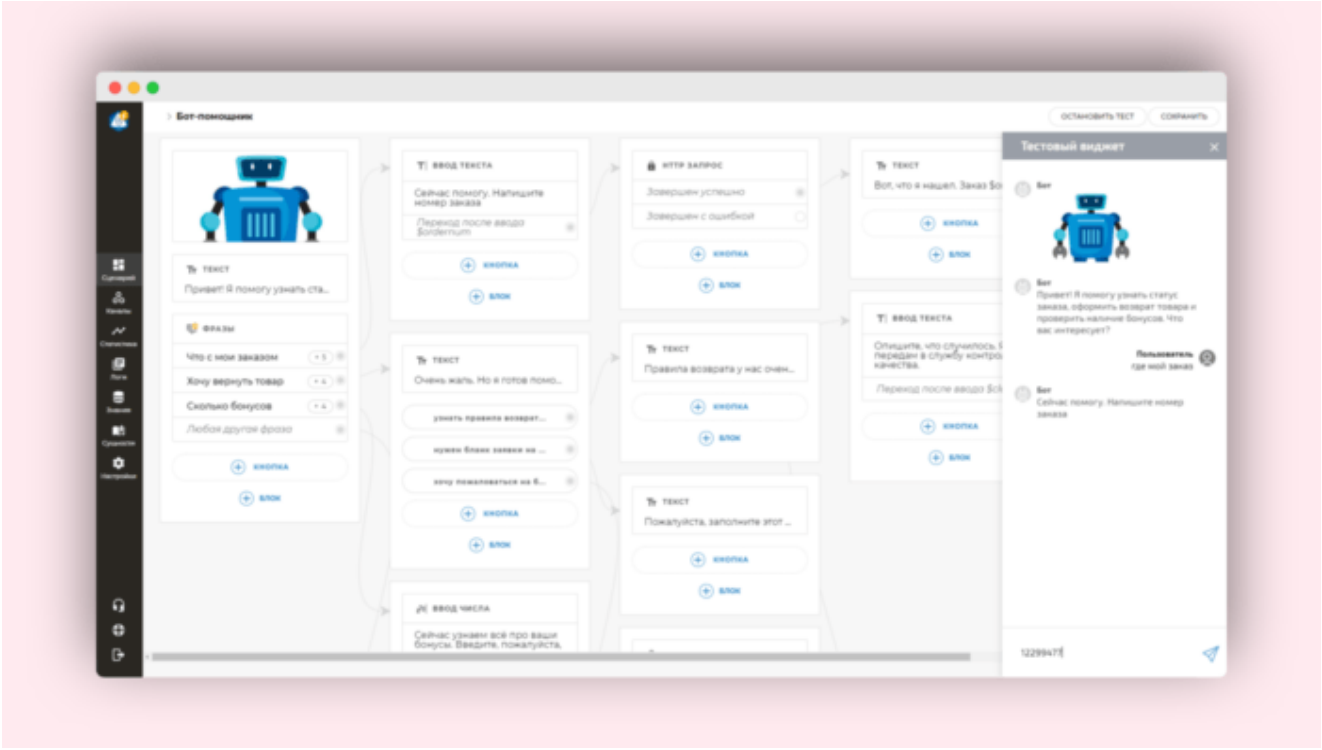

Рисунок 1.4 – технологія розробки чатботу

1.5 Постановка завдання

Було поставлено завдання розробити чатбота для телеграму . Він повинен надавати такі можливості:

- Чатбот повинен бути активним 24/7
- У функціонал бота повинні входити перелік стартових команд для роботи з ним
- Можливість моментально перевіряти данні курсу валют без любих затримок
- Перехід на онлайн ресурси для перевірки правдиності інформації

2 ТЕХНОЛОГІЇ РОЗРОБЛЕННЯ BACK-END ЧАСТИНИ ПРОЕКТУ

2.1 Мова програмування C#

C# (вимовляється Сі-шарп) – об'єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET. Розроблена Андерсом Гейлсбергом, Скотом Вілтамутом та Пітером Гольде під егідою Microsoft Research (при фірмі Microsoft).

Синтаксис C# близький до C++ і Java. Мова має строгу статичну типізацію, підтримує поліморфізм, перевантаження операторів, вказівники на функції-члени класів, атрибути, події, властивості, винятки, коментарі у форматі XML. Перейнявши багато що від своїх попередників — мов C++, Delphi, Модула і Smalltalk – C#, спираючись на практику їхнього використання, виключає деякі моделі, що зарекомендували себе як проблематичні при розробці програмних систем, наприклад множинне спадкування класів (на відміну від C++).

Історія виникнення

C# є дуже близьким родичем мови програмування Java. Мова Java була створена компанією Sun Microsystems, коли глобальний розвиток інтернету поставив задачу роззосереджених обчислень. Взявши за основу популярну мову C++, Java виключила з неї потенційно небезпечні речі (типу вказівників без контролю виходу за межі). Для роззосереджених обчислень була створена концепція віртуальної машини та машинно-незалежного байт-коду, свого роду посередника між вихідним текстом програм і апаратними інструкціями комп'ютера чи іншого інтелектуального пристрою.

Java набула чималої популярності, і була ліцензована також і компанією Microsoft. Але з плином часу Sun почала винуватити Microsoft, що та при створенні свого клону Java робить її сумісною виключно з платформою Windows, чим суперечить самій концепції машинно-незалежного середовища виконання і порушує ліцензійну угоду. Microsoft відмовилася піти назустріч вимогам Sun, і

тому з'ясування стосунків набуло статусу судового процесу. Суд визнав позицію Sun справедливою, і зобов'язав Microsoft відмовитися від позаліцензійного використання Java.

У цій ситуації в Microsoft вирішили, користуючись своєю вагою на ринку, створити свій власний аналог Java – мову, в якій корпорація стане повновладним господарем. Ця новостворена мова отримала назву C#. Вона успадкувала від Java концепції віртуальної машини (середовище .NET), байт-коду (MSIL) і більшої безпеки вихідного коду програм, плюс врахувала досвід використання програм на Java. Нововведенням C# стала можливість легшої взаємодії, порівняно з мовами-попередниками, з кодом програм, написаних на інших мовах, що є важливим при створенні великих проектів. Якщо програми на різних мовах виконуються на платформі .NET, .NET бере на себе клопіт щодо сумісності програм (тобто типів даних, за кінцевим рахунком).

Станом на сьогодні C# визначено флагманською мовою корпорації Microsoft, бо вона найповніше використовує нові можливості .NET. Решта мов програмування, хоч і підтримуються, але визнані такими, що мають спадкові прогалини щодо використання .NET.

Особливості мови:

Портативність

C# розроблялась як мова програмування прикладного рівня для CLR і тому вона залежить, перш за все, від можливостей самої CLR. Це стосується, перш за все, системи типів C#. Присутність або відсутність тих або інших виразних особливостей мови диктується тим, чи може конкретна мовна особливість бути трансльована у відповідні конструкції CLR. Так, з розвитком CLR від версії 1.1 до 2.0 значно збагатився і сам C#; подібної взаємодії слід чекати і надалі. (Проте ця закономірність буде порушена з виходом C# 3.0, що є розширеннями мови,що не спираються на розширення платформи .NET.) CLR надає C#, як і всім іншим .NET-орієнтованим мовам, багато можливостей, яких позбавлені «класичні» мови програмування. Наприклад, збірка сміття не реалізована в самому C#, а

проводиться CLR для програм, написаних на C# точно так, як і це робиться для програм на VB.NET, J# тощо.

Типи даних

C# підтримує строго типізовані неявні оголошення змінних з ключовим словом var і неявно типізовані масиви з ключовим словом new [], за яким слідує ініціалізатор колекції.

C# підтримує суворий тип даних Boolean, bool. Вирази, які приймають умови, такі як while та if, вимагають висловлювання, що реалізує оператор true або false. Хоча C++ також має тип Boolean, він може бути вільно перетворений в цілі числа та з них, а вирази, такі як if(a), вимагають тільки того, щоб a був конвертований в bool, що дозволяє бути a int-типу або вказівником. C# забороняє «ціле значення означає справжній або помилковий підхід» на тій підставі, що примус програмістів використовувати вирази, які повертають точно bool, можуть створювати деякі типи помилок програмування, наприклад if (a = b) (використання присвоювання = замість рівності == , які, хоча і не є помилкою на C або C ++, все одно будуть спіймані компілятором).

C# безпечніший в порівнянні з C++. Єдиними неявними перетвореннями за умовчуванням є ті, які вважаються безпечними, наприклад, розширення цілих чисел. Це застосовується під час компіляції, під час JIT і, в деяких випадках, під час виконання. Не відбувається неявних перетворень між булевими і цілими числами, а також між членами перерахування і цілими числами (крім літерала 0, який може бути неявно перетворений в будь-який нумерований тип). Будь-яке призначене для користувача перетворення повинно бути явно позначене як явне або неявне, на відміну від конструкторів копіювання C++ і операторів перетворення, які за умовчанням є неявними.

C# має явну підтримку коваріації та контраваріантності в родових типах, на відміну від C++, яка має певний рівень підтримки контраваріантності просто через семантику типів,що повертаються, на віртуальні методи.

Члени перерахування розміщуються в своєму власному обсязі.

Мова C # не допускає глобальних змінних або функцій. Всі методи і члени повинні бути оголошені всередині класів. Статичні члени відкритих класів можуть замінювати глобальні змінні та функції.

Метапрограмування

Метапрограмування через атрибути C# є частиною мови. Багато з цих атрибутів дублюють функціональні можливості директив препроцесора, орієнтованих на платформу GCC і VisualC ++.

Методи та функції

Методи в мові програмування є членами класу в проекті, деякі методи мають підписи, а деякі не мають підпису. Методи можуть бути недійсними або можуть повертати щось на зразок рядка, цілого, подвійного, десяткового, float і bool. Якщо метод недійсний, це означає, що метод не повертає жодного типу даних.

Подібно C++, і на відміну від Java, програмісти на C # повинні використовувати ключове слово virtual, щоб дозволити перевизначати методи підкласами[1].

Методи розширення в C# дозволяють програмістам використовувати статичні методи, як якщо б вони були методами з таблиці методів класу, дозволяючи програмістам додавати методи до об'єкта, який, на їхню думку, повинен існувати на цьому об'єкті і його похідних.

Динамічний тип dynamic допускає прив'язку методу під час виконання, що дозволяє використовувати JavaScript-подібні виклики методів і склад часу виконання.

У C# є підтримка строго типізованих покажчиків функцій через delegate ключового слова. Подібно псевдо-C ++ - signal і slot фрейма Ct, C# має семантику, спеціально пов'язану з подіями стилю публікації-підписки, хоча C# використовує делегати для цього.

C # пропонує Java-подібні синхронізовані synchronized виклики методів через атрибут [MethodImpl (MethodImplOptions.Synchronized)] і підтримує взаємовиключні блокування за допомогою блокування ключових слів.

Властивості

C# надає властивості як синтаксичного цукру для загального шаблону, в якому пара методів, accessor (getter) і mutator (setter) інкапсулює операції по одному атрибуту класу. Не потрібно писати надлишкові сигнатури методів для реалізацій геттера / сетера і до цієї властивості можна отримати доступ, використовуючи синтаксис атрибутів, а не більш докладні виклики методів.

Синтаксис

Синтаксис мови C# схожий на інші мови C-стилів, такі як C, C ++ і Java. Зокрема:

– крапки з комою використовуються для позначення кінця строки коду;

– фігурні дужки використовуються для угруповання операторів;

– строки коду зазвичай групуються в методи (функції), методи в класи і класи в простір імен;

– змінні присвоюються з використанням знака рівності, а порівнюються з використанням двох послідовних знаків рівності.

Стандартизація

C# стандартизований в ECMA та ISO.

У серпні 2000 Microsoft Corporation, Hewlett-Packard та Intel Corporation виступили спонсорами стандартизації специфікації мови C#, а також Common Language Infrastructure (CLI) в організації зі стандартизації ECMA International. У грудні 2001 ECMA випустила ECMA-334 Специфікація мови C#. C# стала стандартом ISO у 2003 (ISO/IEC 23270:2006 — Information technology— Programming languages—C#). До того ECMA ще встигла адоптувати еквівалентну специфікацію як другу редакцію C# у грудні 2002.

У червні 2005 ECMA схвалила редакцію 3 специфікації C#, і відредагувала ECMA-334. Доповнення включали часткові класи, анонімні методи, тип null, і генерики (аналоги шаблонів C++).

У липні 2005 ECMA подала стандарти і відповідні технічні умови на ISO/IEC JTC 1 через пришвидшену процедуру (Fast-Track). Цей процес звичайно займає 6-9 місяців.

2.2 ASP.NET

ASP.NET — технологія створення веб-застосунків і веб-сервісів від компанії Майкрософт. Вона є складовою частиною платформи Microsoft.NET і розвитком старішої технології Microsoft ASP. На цей час останньою версією цієї технології є ASP.NET Core 2.0

ASP.NET зовні багато в чому зберігає схожість із старішою технологією ASP, що дозволяє розробникам відносно легко перейти на ASP.NET. У той же час внутрішній устрій ASP.NET істотно відрізняється від ASP, оскільки вона заснована на платформі. NET і, отже, використовує всі нові можливості, що надаються цією платформою. Веб-сторінки ASP.NET, офіційно відомі як веб-форми, є основними елементами для розробки додатків в ASP.NET. Є дві основні методології веб-форм — формат веб-додатків та формат веб-сайту.

Веб-додатки потрібно скласти перед розгортанням, тоді як структури веб-сайтів дозволяють користувачеві копіювати файли безпосередньо на сервер без попередньої компіляції. Веб-форми містяться в файлах з розширенням «.aspx»; ці файли зазвичай містять статичну (X) HTML розмітку. Розмітка компонентів може включати веб-елементи керування на стороні сервера та елементи керування на стороні користувача, які були визначені на фреймворці веб-сторінки.

Крім того, динамічний код, який працює на сервері, може бути розміщений на сторінці в блоці <% - динамічний код -%>, подібно до інших технологій веб-розробки, таких як PHP, JSP та ASP. З виходом ASP.NET Framework 2.0, Microsoft представила нову модель, що дозволяє використовувати статичний текст на сторінці .aspx, при цьому динамічний код залишається у файлі .aspx.vb, .aspx.cs або .aspx.fs

Структура проекту в ASP.NET (рис 2.1):

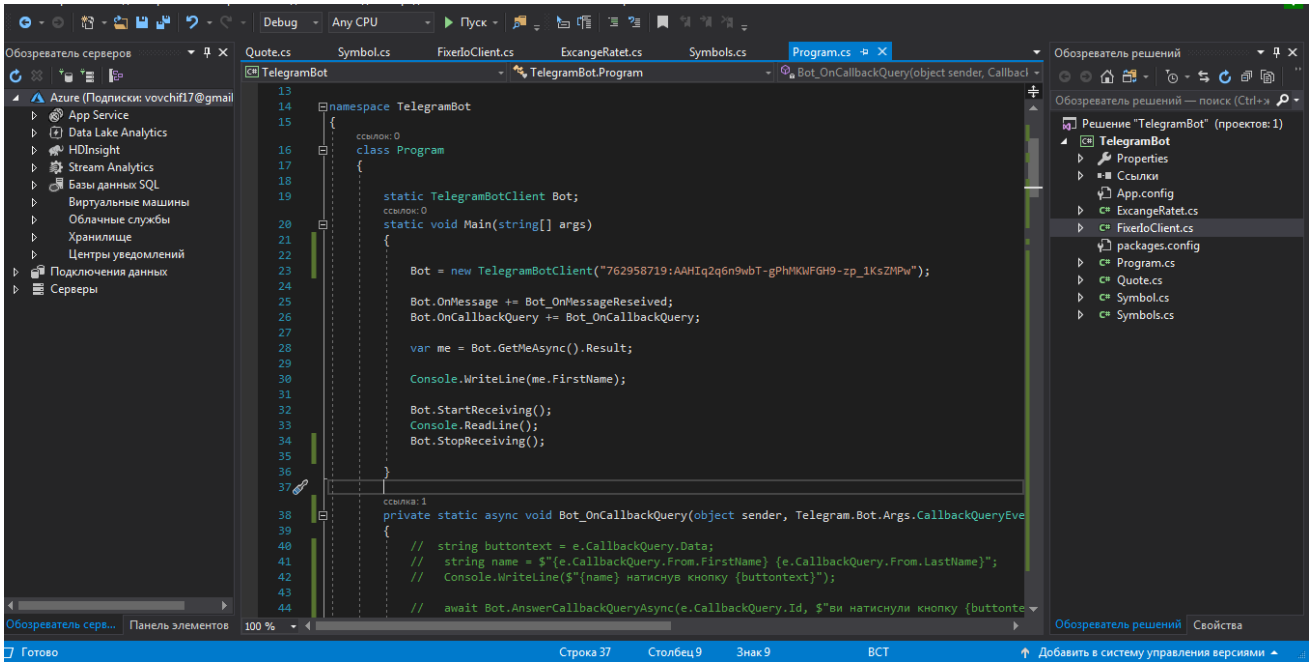

Рисунок 2.1 — Структура проетку ASP.NET

Основними компонентами застосунків C# ASP.NET є модель (model), вид (view) і контролер (controller).

Модель надає решті компонентів програми об'єктно-орієнтоване представлення даних (таких як каталог продуктів або список замовлень). Об'єкти моделі здійснюють завантаження і збереження даних в реляційній базі даних.

Завдяки можливостям динамічної типізації в мові C# розробникові досить успадкувати свій клас моделі від базового класу ActiveRecord::Base. C# автоматично пов'язує класи моделі з таблицями в базі даних і створює атрибути об'єктів для відповідних полів таблиці.

2.3 Шаблон проектування MVC

Моде́ль–вигляд–контро́лер (або Модель–представлення–контролер, англ. Model-view-controller, MVC) — архітектурний шаблон, який використовується під час проектування та розробки програмного забезпечення (рис 2.2).

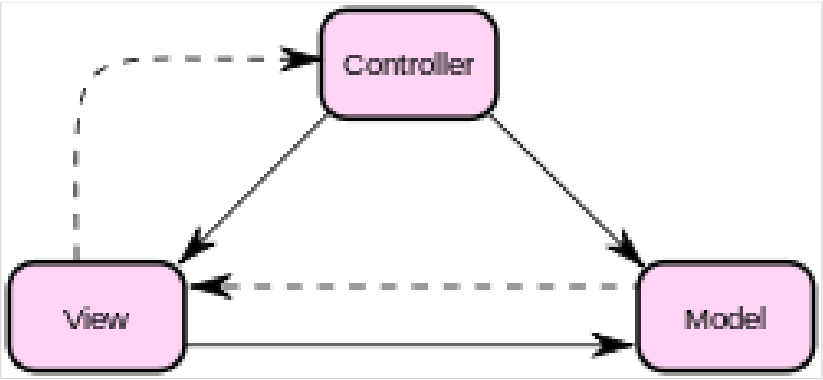

Рисунок 2.2 – Діагама взаємодії між компонентами шаблону

Цей шаблон передбачає поділ системи на три взаємопов'язані частини: модель даних, вигляд (інтерфейс користувача) та модуль керування. Застосовується для відокремлення даних (моделі) від інтерфейсу користувача (вигляду) так, щоб зміни інтерфейсу користувача мінімально впливали на роботу з даними, а зміни в моделі даних могли здійснюватися без змін інтерфейсу користувача.

Мета шаблону — гнучкий дизайн програмного забезпечення, який повинен полегшувати подальші зміни чи розширення програм, а також надавати можливість повторного використання окремих компонентів програми. Крім того використання цього шаблону у великих системах сприяє впорядкованості їхньої структури і робить їх більш зрозумілими за рахунок зменшення складності.

У рамках архітектурного шаблону модель–вигляд–контролер (MVC) програма поділяється на три окремі, але взаємопов'язані частини з розподілом функцій між компонентами. Модель (Model) відповідає за зберігання даних і забезпечення інтерфейсу до них. Вигляд (View) відповідальний за представлення цих даних користувачеві. Контролер (Controller) керує компонентами, отримує сигнали у вигляді реакції на дії користувача (зміна положення курсора миші, натискання кнопки, ввід даних в текстове поле) і передає дані у модель.

– модель є центральним компонентом шаблону MVC і відображає поведінку застосунку, незалежну від інтерфейсу користувача. Модель стосується прямого керування даними, логікою та правилами застосунку;

– вигляд може являти собою будь-яке представлення інформації, одержуване на виході, наприклад графік чи діаграму. Одночасно можуть співіснувати кілька виглядів (представлень) однієї і тієї ж інформації, наприклад гістограма для керівництва компанії й таблиці для бухгалтерії;

– контролер одержує вхідні дані й перетворює їх на команди для моделі чи вигляду.

Модель інкапсулює ядро даних і основний функціонал їхньої обробки і не залежить від процесу вводу чи виводу даних.

Вигляд може мати декілька взаємопов'язаних областей, наприклад різні таблиці і поля форм, в яких відображаються дані.

У функції контролера входить відстеження визначених подій, що виникають в результаті дій користувача. Контролер дозволяє структурувати код шляхом групування пов'язаних дій в окремий клас. Наприклад у типовому MVC-проекті може бути користувацький контролер, що містить групу методів, пов'язаних з управлінням обліковим записом користувача, таких як реєстрація, авторизація, редагування профілю та зміна пароля.

Зареєстровані події транслюються в різні запити, що спрямовуються компонентам моделі або об'єктам, відповідальним за відображення даних. Відокремлення моделі від вигляду даних дозволяє незалежно використовувати різні компоненти для відображення інформації. Таким чином, якщо користувач через контролер внесе зміни до моделі даних, то інформація, подана одним або декількома візуальними компонентами, буде автоматично відкоригована відповідно до змін, що відбулися.

2.4 API

Прикладни́й програ́мний інтерфе́йс (інтерфейс програмування застосунків, інтерфейс прикладного програмування) (англ. Application Programming Interface, API) — набір визначень підпрограм, протоколів взаємодії та засобів для створення

програмного забезпечення. Спрощено - це набір чітко визначених методів для взаємодії різних компонентів. API надає розробнику засоби для швидкої розробки програмного забезпечення. API може бути для веб-базованих систем, операційних систем, баз даних, апаратного забезпечення, програмних бібліотек.

2.5 Платформа для тестування роботи проекту Telegram

Telegram — месенджер, програмне забезпечення для смартфонів, планшетів та ПК, яке дозволяє обмінюватися текстовими повідомленнями та різноманітними файлами, зокрема графічними файлами та відеофайлами, а також безкоштовно телефонувати іншим користувачам програми.

Обліковий запис користувача прив'язується до номеру мобільного телефону: щоб авторизуватися, потрібно ввести код авторизації з СМС. Такі коди мають обмежені терміни придатності. Таким чином, користувач позбавляється необхідності запам'ятовувати чи зберігати десь свій пароль.

Боти

За допомогою спеціального API сторонні розробники можуть створювати «ботів», спеціальні акаунти, керовані програмами. Типові боти відповідають на спеціальні команди в персональних і групових чатах, також вони можуть здійснювати пошук в інтернеті або виконувати інші завдання, застосовуються задля розваг або в бізнесі. Також чат-ботів використовують для досягнення якої-небудь мети (наприклад, надання потрібної інформації) або задля розваги. Чат-боти часто використовуються в системах діалогу для різних практичних цілей, включаючи обслуговування клієнтів або отримання інформації. Деякі чат-боти використовують складні системи обробки людської мови, але більшість використовує простіші системи.

2.6 Архітектура

Программа була розроблена на мові програмування C# і основне середовище. Программа має наступну архітектуру (рис 2.3).

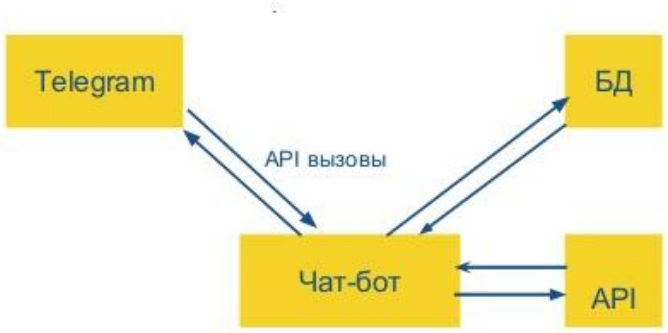

Рисунок 2.3 – Архітектура проекту

Оскільки наш чат бот є посередником, який передає данні за допомогою API, а данні бере з БД, для нього підходить стандартна архітектура веб додатку, який зв’язується з API.

Для роботи программи необхідно наступні компоненти:

– telegram;

– программа TelegramBot;

– робочі API від онлайн сервісів.

При запуску программи ми переходимо в чат бот, який є зв’язаний з нашою програмою за допомогою зовнішнього ключа, який ми отримуємо при створенні чат бота. Після запуску программи ми маємо можливість виконати Перевірку курсу валют по декількох сервісах, які є пов’язані з нашим чатботом за допомогою своїх API, ключ яких теж потрібно записувати в нашу программу. В результаті наша программа виступає посередником між користувачем та іншими онлайн сервісами курсу валют

3 BACK-END ЧАСТИНА ПРОЕКТУ

3.1 Класи проекту

Після вивчення предметної області було розроблено початковий функціонал чатбота який виводить курси валют і взаємодію з ним (рис. 3.1). Після чого була отримання API від телеграму нашого ботя для програмування його на C# використовуючи Back-end розробку (серверну архітектуру):

Для підключення до телеграму необхідно звернутись спочатку до боту який називається BotFather з якого починається створення бота:

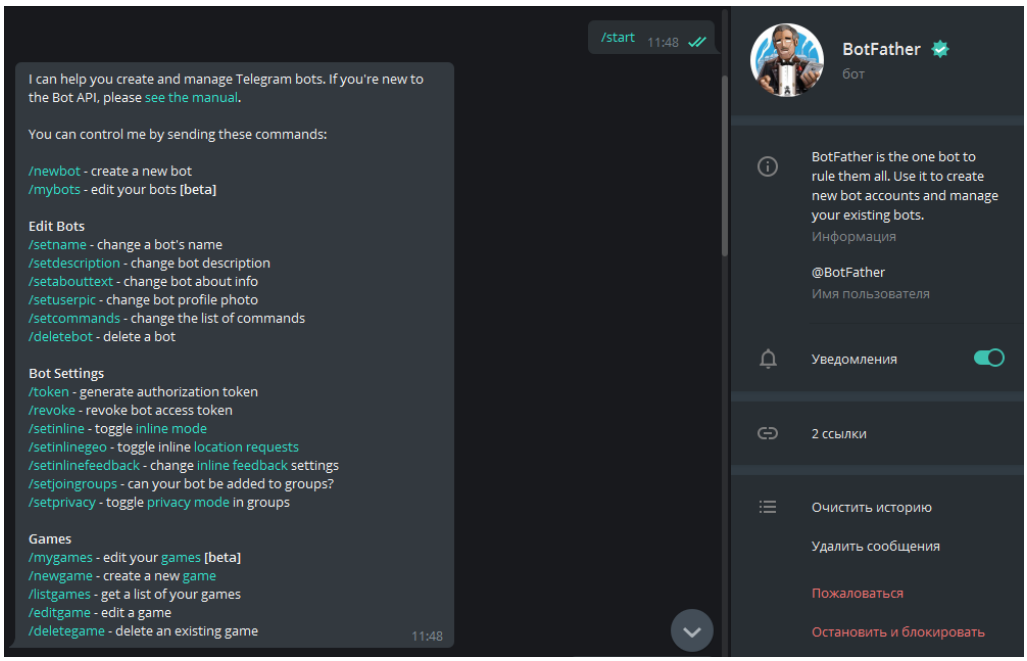

Рисунок 3.1 — BotFather

Після введення команди „/start” бот виводить нам доступні команди з яких використовуються команда „/newbot” для створення нового бота, якому потрібно дати назву в пошуку а також назву в діалозі (рис 3.2):

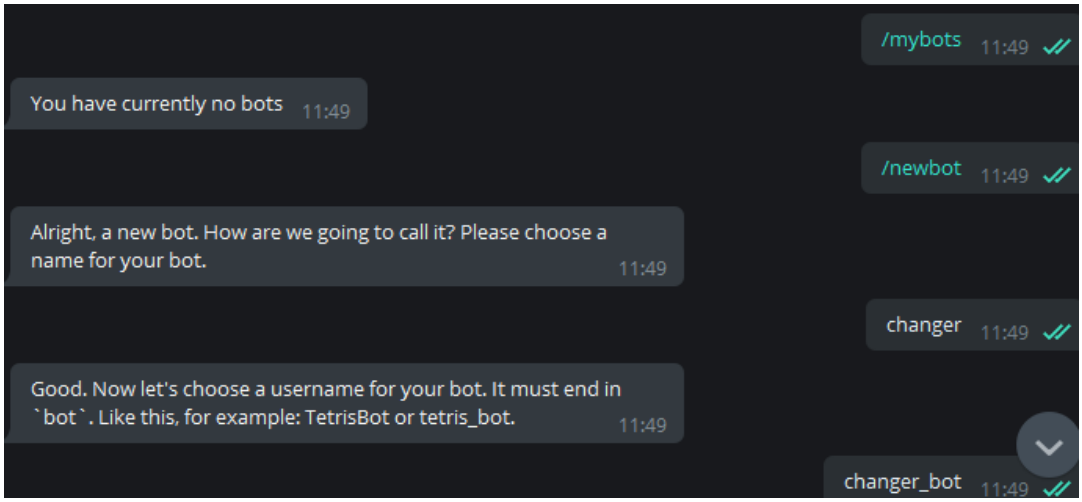

Рисунок 3.2 — Створення бота

Після створення бота BotFather надає нам доступи до чатбота (як для діалогу так і для розробника – у вигляді api для коннекту з ним) (рис 3.3)

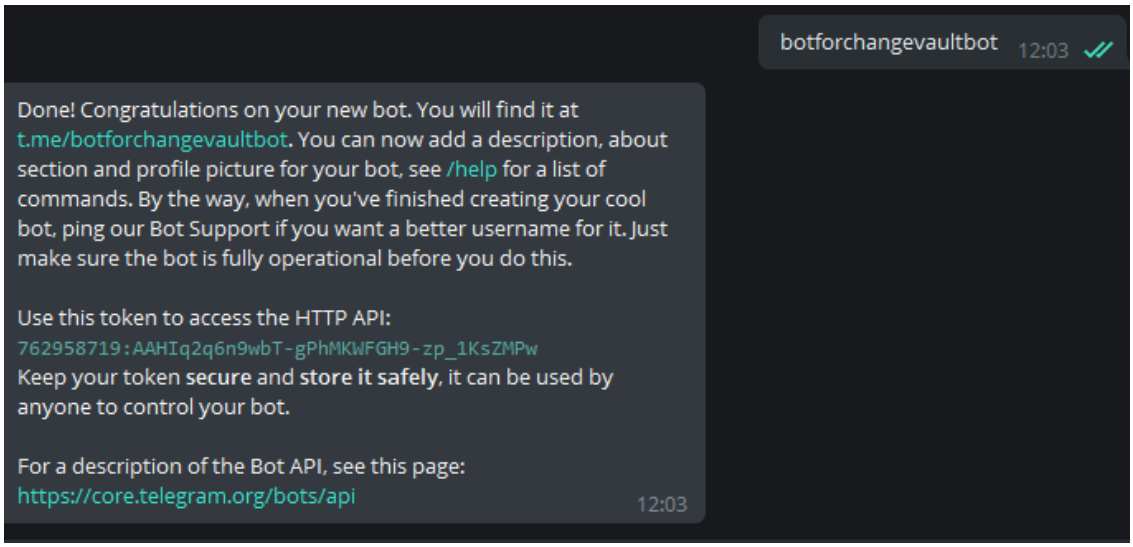

Рисунок 3.3 — Контакт

3.2 Опис функціональної частини

Для того щоб підключитись до чатбота у C# необхідно включити в t клас ідентифікатор бот - API

Частина коду Program.cs:

```
        static void Main(string[] args)
         {

             Bot = new TelegramBotClient("762958719:AAHIq2q6n9wbT-
gPhMKWFGH9-zp_1KsZMPw");

             Bot.OnMessage += Bot_OnMessageReseived;
             Bot.OnCallbackQuery += Bot_OnCallbackQuery;

             var me = Bot.GetMeAsync().Result;

             Console.WriteLine(me.FirstName);

             Bot.StartReceiving();
             Console.ReadLine();
             Bot.StopReceiving();
```

Для кофігурації роботи з сервісом fixer.io, API які ми використовуєм використовується API данного джерела

Приклад коду:

```
public void Configure(IApplicationBuilder app)
         {
         // to SetApiKey
         Fixer.SetApiKey("3e8d219b9ada7db2dfcb4cf8390d8f69");
         }
    }
    public class Fixer
    {
         private const string BaseUri = "http://data.fixer.io/api/";
```

```
        private static string _apiKey;

        private static string ApiKey
        {
            get => !string.IsNullOrWhiteSpace(_apiKey) ? _apiKey :
throw new InvalidOperationException("Fixer.io now requires an API
key! Call .SetApiKey(\"key\") first");
            set => _apiKey = value;
        }
        public static double Convert(string from, string to, double
amount, DateTime? date = null)
        {
            return GetRate(from, to, date).Convert(amount);
        }

        public static async Task<double> ConvertAsync(string from,
string to, double amount, DateTime? date = null)
        {
            return (await GetRateAsync(from, to,
date)).Convert(amount);
        }

        public static ExchangeRate Rate(string from, string to,
DateTime? date = null)
        {
            return GetRate(from, to, date);
        }

        public static async Task<ExchangeRate> RateAsync(string
from, string to, DateTime? date = null)
        {
            return await GetRateAsync(from, to, date);
        }

        public static void SetApiKey(string apiKey)
```

```
        {
            ApiKey = apiKey;
        }
```

Весь код файлу наведено в додатку А.

Нижче наведений код файлу, який повертає оброблені данних в реальному часі

Код файлу Quote.cs:

```
using System;
using System.Collections.Generic;

namespace TelegramBot
{
    public class Quote
    {
        public String Base { get; set; }

        public DateTime Date { get; set; }

        public IDictionary<string, decimal> Rates { get; set; }
    }
}
```

В данному Stringbase повертає значення ім’я користувача, який проводить дану операцію, і яке є невидимим для користувача

DateTime повертає поточний час

Idictionari повертає валюту яку нам потрібно

Rates повертає значення яке було взято поточної валюти з сервісу

Щоб запустити бота в Telegram спочатку треба його знайти за ім’ям,яке ми йому дали при створенні (рис 3.4).

Рисунок 3.4 — Результат пошуку чатбота.

Для нашого бота була створено меню, яка зроблена для облегшення користування ботом (рис 3.5).

Рисунок 3.5 — Результат пошуку чатбота.

Код першого і другого меню:

```
private static async void Bot_OnMessageReseived(object sender,
Telegram.Bot.Args.MessageEventArgs e)
        {
            var message = e.Message;
            String name = $"{message.From.FirstName} {
message.From.LastName}";
            Console.WriteLine($"{name} відправив повідомленнк
'{message.Text}'");

            switch (message.Text)
            {
                case "/start":
                    string text =
@"Список команд:
/start  - запуск бота
/menu - вивід меню
/keyboard - вивід клавіатури";
                    await Bot.SendTextMessageAsync(message.From.Id,
text);
```

```
                    break;
                case "/menu":
                    var inlineKeyBoard = new
InlineKeyboardMarkup(new[]
                    {
                        new [] {
InlineKeyboardButton.WithUrl("finance.ua","https://finance.i.ua/ban
k/10/"),

InlineKeyboardButton.WithUrl("Приватбанк","https://minfin.com.ua/ua
/company/privatbank/currency/"),
                        },
                        new []
                        {

InlineKeyboardButton.WithCallbackData("USD"),


InlineKeyboardButton.WithCallbackData("EUR"),


InlineKeyboardButton.WithCallbackData("RUB"),


InlineKeyboardButton.WithCallbackData("PLN"),


                        }
                    });

                    await Bot.SendTextMessageAsync(
                        message.Chat.Id,
                        "Курс",
                        replyMarkup: inlineKeyBoard);
                    break;
```

Після написання вигляд основного меню має такий вигляд (рис 3.6):

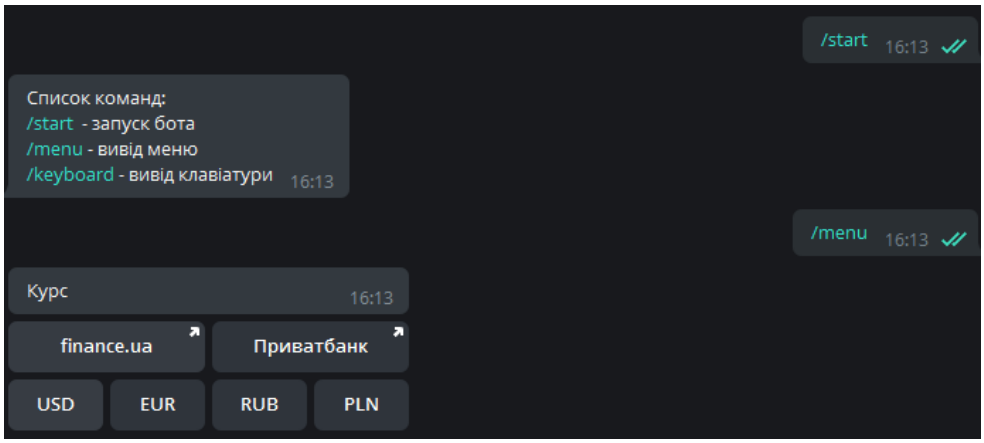

Рисунок 3.6 – Результат створення меню.

В данному меню чатбот пропонує відвідати два сайти, які відображають поточний курс валют, або вивести данні по поточному курсу Міжнародного національного банку.

Перша частина меню відкриває сайт Міністерства фінансів, який відображає курс, а в другий – курс по приватбанку (рис 3.7).

Наступна частина меню відображає значення по міжнародному банку.

Данні по міжнародному банку показуються як сповіщення, що б не «замусурювати чат»

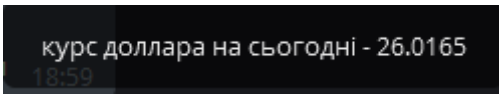

Рисунок 3.7 – Результат створення меню.

Частина коду яка відповідає за сповіщення

```
   private static ExchangeRate GetRate(string from, string to,
DateTime? date = null)
        {
            from = from.ToUpper();
            to = to.ToUpper();

            if (!Symbols.IsValid(from))
```

```
                throw new ArgumentException("Symbol not found for
provided currency", "from");

            if (!Symbols.IsValid(to))
                throw new ArgumentException("Symbol not found for
provided currency", "to");

            var url = GetFixerUrl(date);

            using (var client = new HttpClient())
            {
                var response = client.GetAsync(url).Result;
                response.EnsureSuccessStatusCode();

                return
ParseData(response.Content.ReadAsStringAsync().Result, from, to);
            }
        }

        private static async Task<ExchangeRate> GetRateAsync(string
from, string to, DateTime? date = null)
        {
            from = from.ToUpper();
            to = to.ToUpper();

            if (!Symbols.IsValid(from))
                throw new ArgumentException("Symbol not found for
provided currency", "from");

            if (!Symbols.IsValid(to))
                throw new ArgumentException("Symbol not found for
provided currency", "to");

            var url = GetFixerUrl(date);
```

```
            using (var client = new HttpClient())
            {
                var response = await client.GetAsync(url);
                response.EnsureSuccessStatusCode();

                return ParseData(await
response.Content.ReadAsStringAsync(), from, to);
            }
        }

        private static ExchangeRate ParseData(string data, string
from, string to)
        {
            var root = JObject.Parse(data);

            var rates = root.Value<JObject>("rates");
            var fromRate = rates.Value<double>(from);
            var toRate = rates.Value<double>(to);

            var rate = toRate / fromRate;
            var returnedDate =
DateTime.ParseExact(root.Value<string>("date"), "yyyy-MM-dd",
                System.Globalization.CultureInfo.InvariantCulture);

            return new ExchangeRate(from, to, rate, returnedDate);
        }

        private static string GetFixerUrl(DateTime? date = null)
        {
            var dateString = date.HasValue ?
date.Value.ToString("yyyy-MM-dd") : "latest";

            return $"{BaseUri}{dateString}?access_key={ApiKey}";
```

Для підключення до сайту Fixer.io і його API необхідно перейти на головну сторінку, в якій ми зможемо отримати доступ до API (рис 3.8).

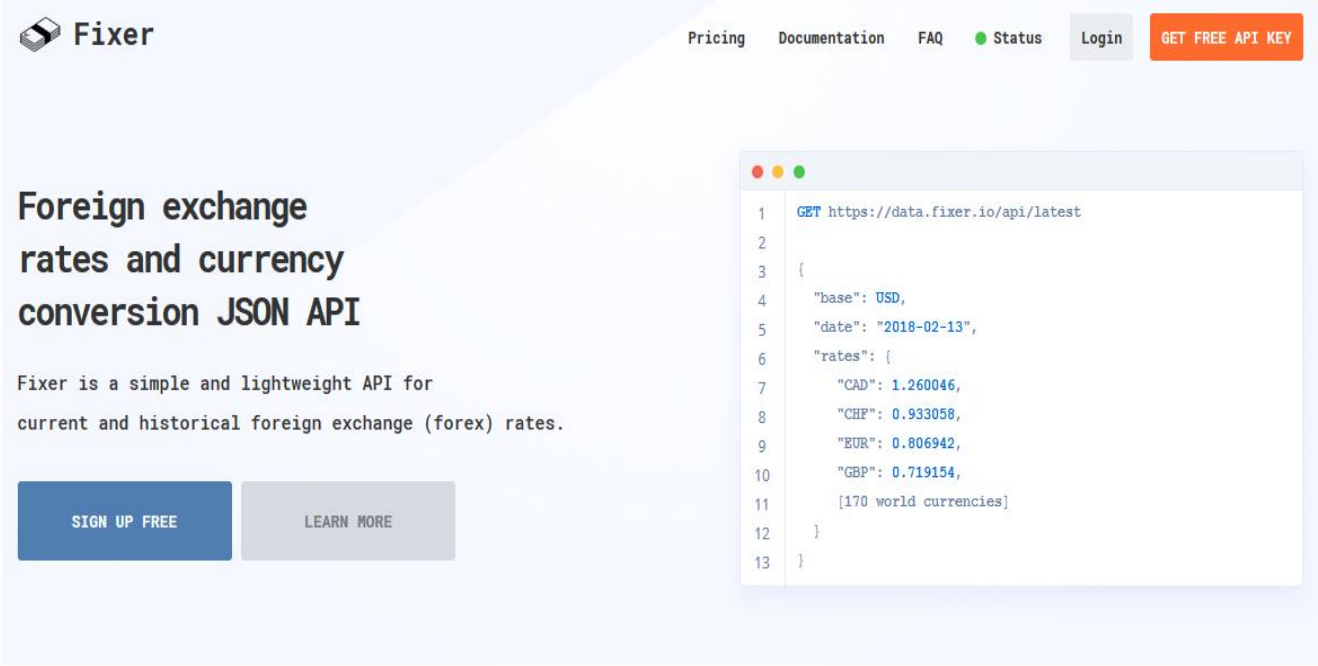

Рисунок 3.8 – Головна сторінка Fixer.io.

Після чого необхідно натиснути на кнопку Get free api key, що б вибрати один із варіантів API як будемо використовувати. Сайт Fixer.io пропонує на вибір декілька варіантів API:

- Free;
- Basic;
- Professional;
- Professional plus;
- Enterprice.

Free дозволяє безплатно використовувати API для зв’язку із программою, проте накладає ряд умов (рис 3.9):

- 1000 викликів до API на місяць;
- погодинне оновлення данних;
- обмеженне обслуговування службою підтримки;
- архів данних.

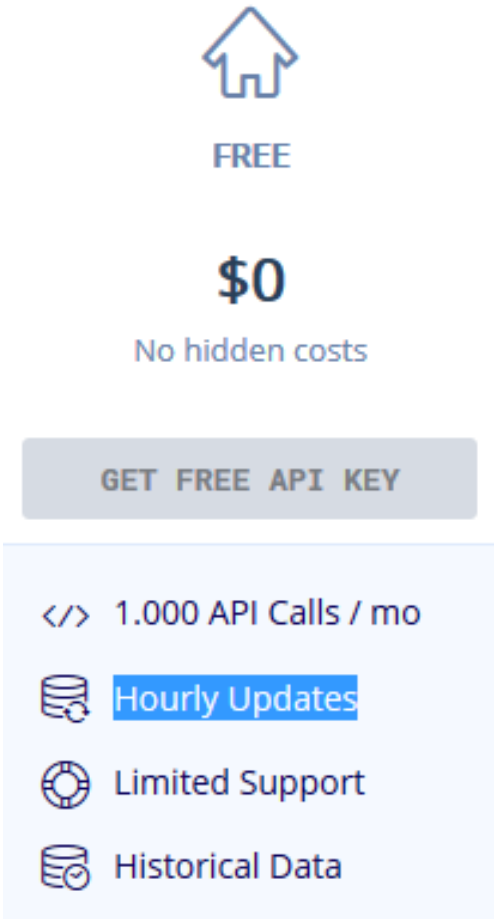

Рисунок 3.9 – Безплатний варіант API

Інші пропозиції, які надає данний сервіс є платними. Їхня ціна коливається від 10 до 80 доларів у місяць. Відповідно послуги, які надаються є якіснішими і більш розширеними (рис 3.10).

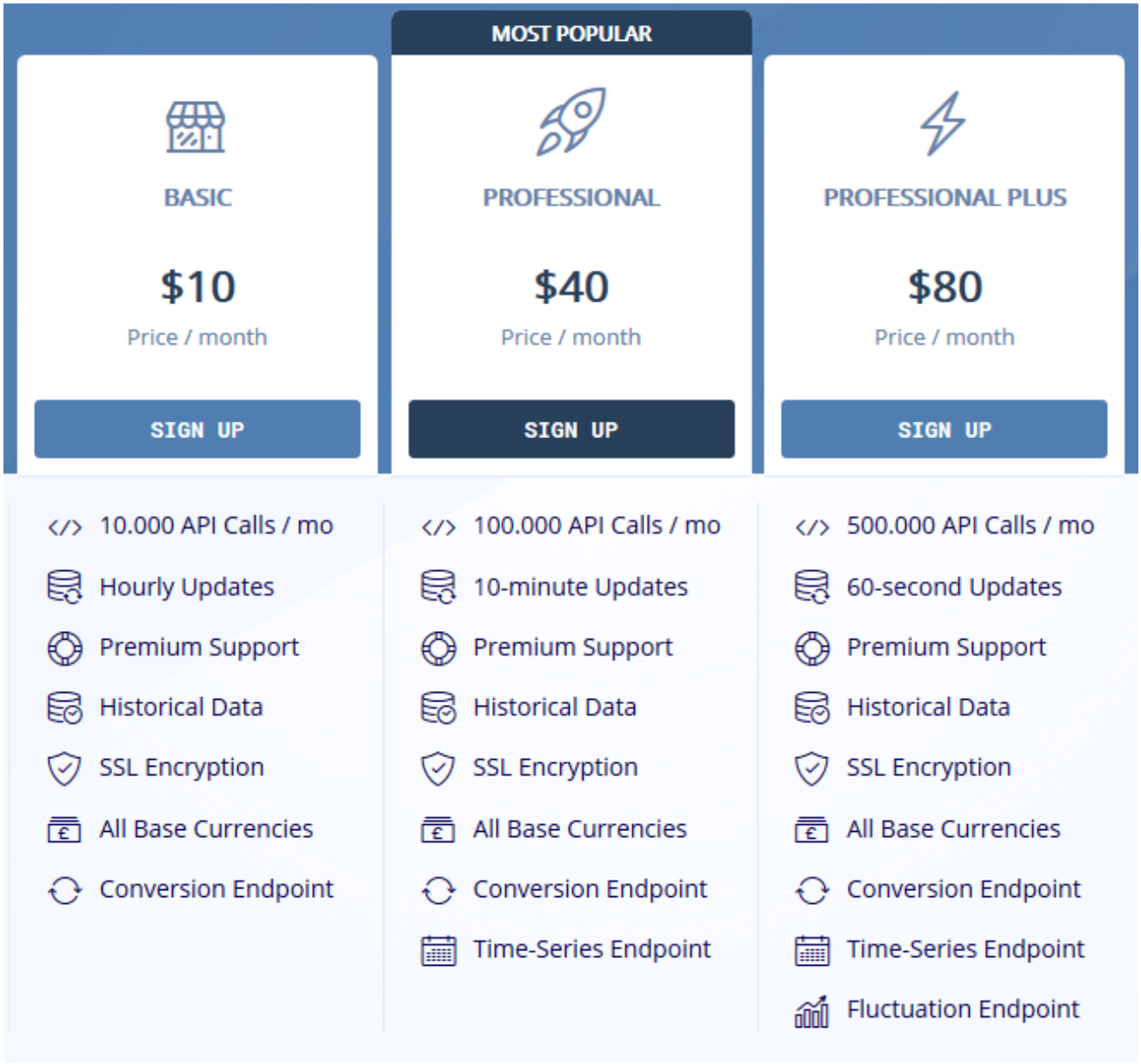

Рисунок 3.10 – Платні варіант API

Третій вид, який надає данний сервіс є API по типу Enterprice призначені для великих компаній і розробки великого програмного продукту з великим функціоналом. Данне API не має своєї ціни і про його обслуговування необхідно зв’язуватись з командою сервісу Fixer io, яка надає такі послуги. Переваги цього API є набагато більші, ніж усіх попередніх (рис 3.11).

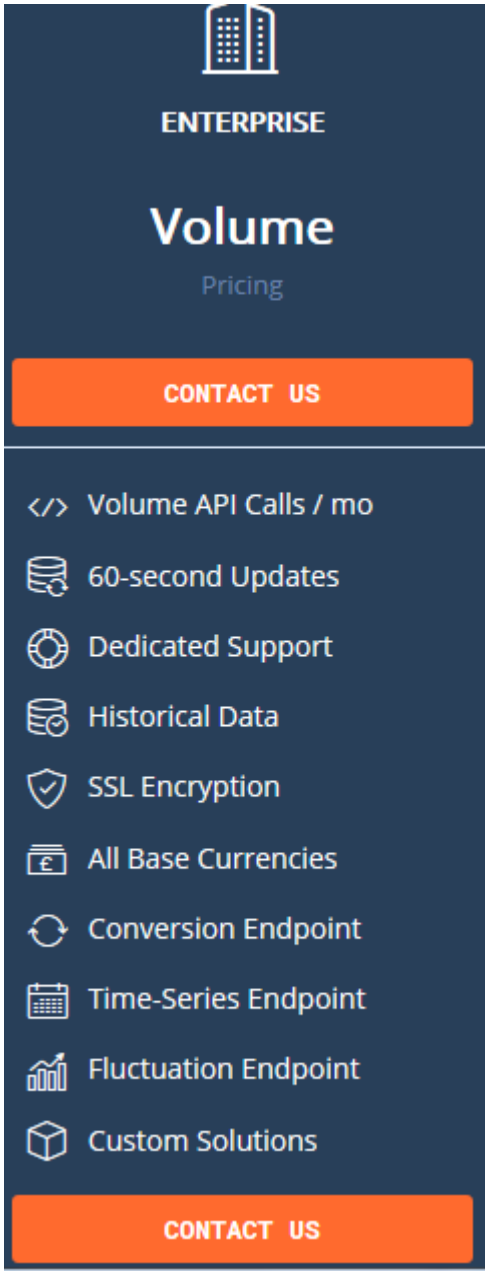

Рисунок 3.11 – Enterprice API

Оскільки наш проект не має необхідності на данний період часу використовувати платний варіант API або API Enterprice, буде використовувати API безплатної версії для демонстрування можливостей проекту.

Після натиснення кнопки Get free api key данний сервер переправляє на сторінку реєстрації (рис 3.12)

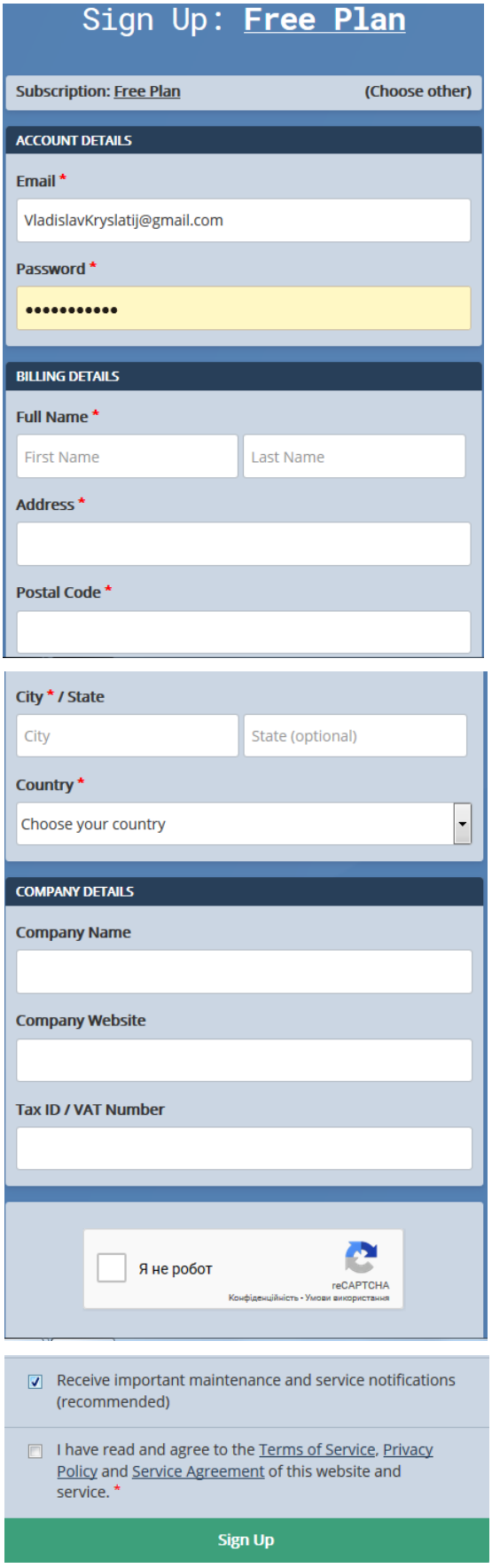

Рисунок 3.12 – Форма реєстрації API

Обов’язковою умовою реєстрації в данній формі є запис даних в полях відзначених зірочкою. Без введених даних зареєструватись неможливо. Після проходження реєстрації данний сервіс надає нам ключ до API, який ми будемо використовувати і надає короткий опис про його використання. Дана сторінка поділена на декілька кроків. У першому кроці показують ключ API згенероване для чатбота, і ссилку на перехід до Account Dashboard де знаходиться можливість його змінити, або змінити логін і пароль користувача (рис 3.13).

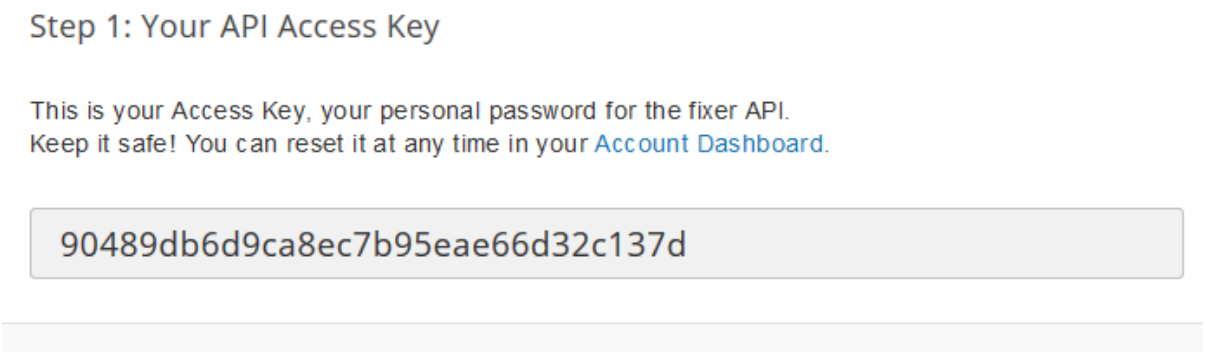

Рисунок 3.13 – Форма реєстрації API

При переході в кабінет можна переглянути ключ API та кількість використаних запитів, що є досить корисним при перевірці використаних запитів а також зміні ключа API (рис 3.14)

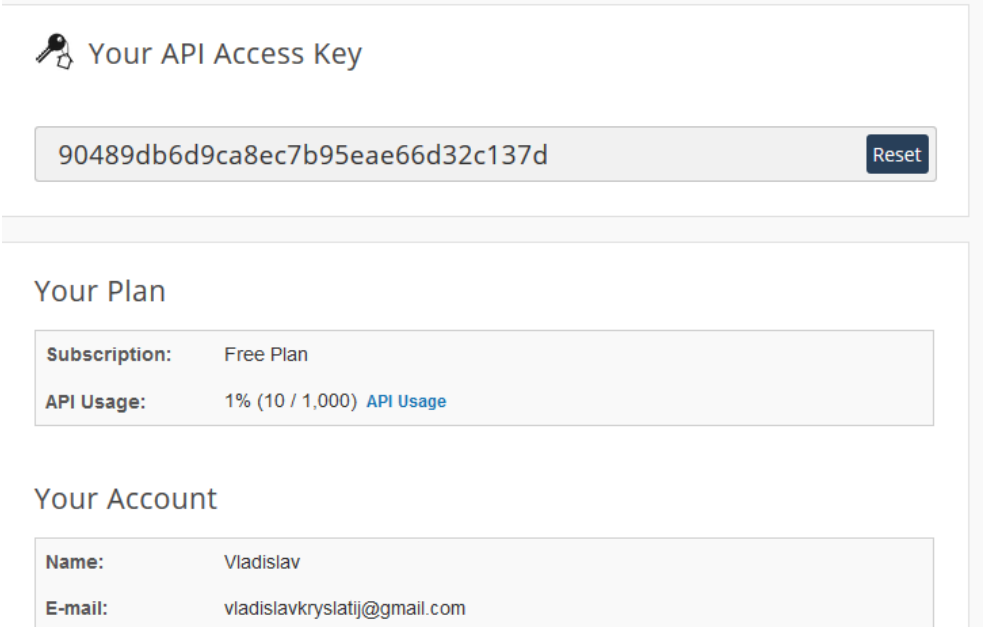

Рисунок 3.14 – Форма реєстрації API

Крок 2 передбачає наглядний приклад використання API в коді в декількох варіаціях (рис 3.15)

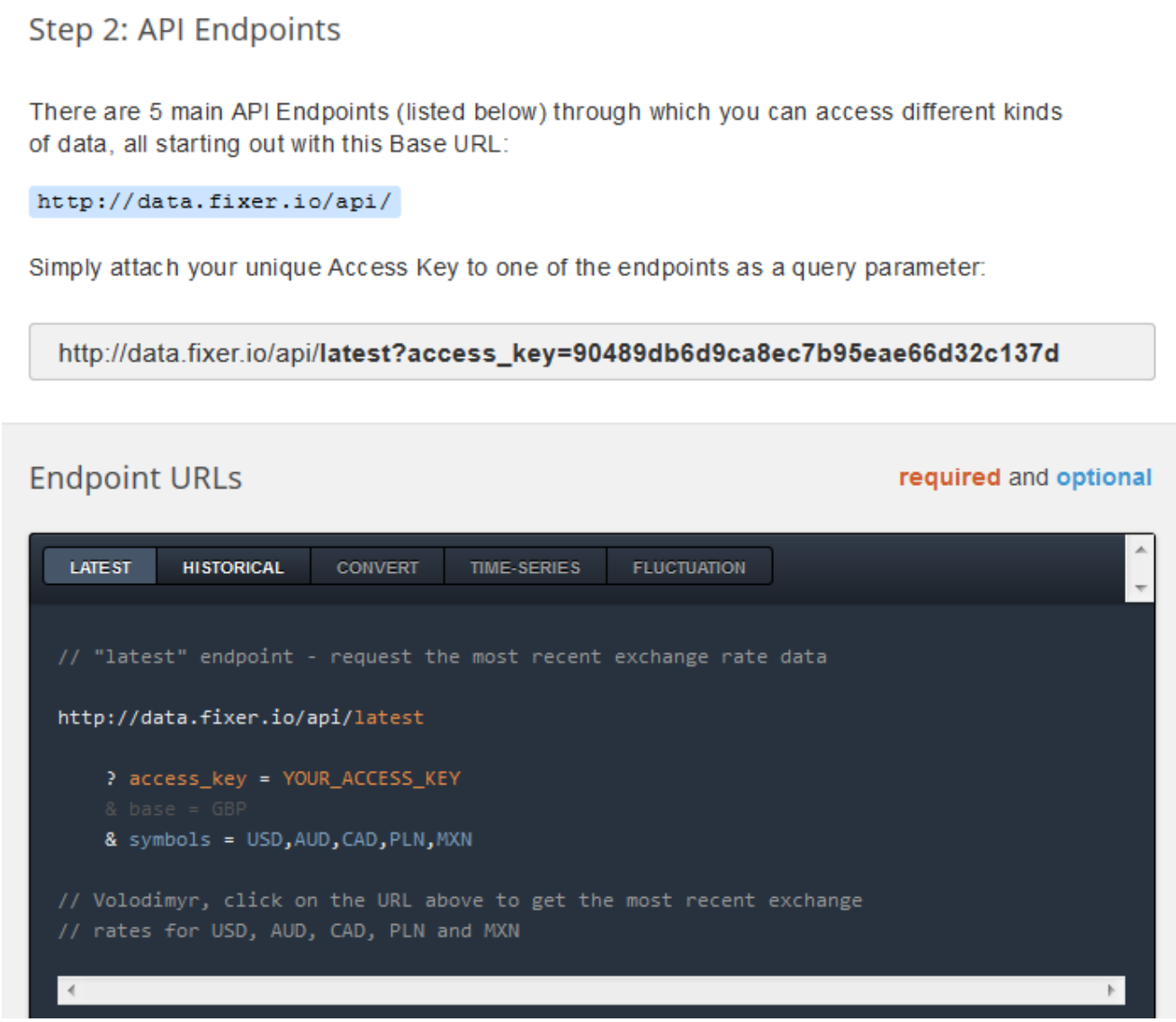

Рисунок 3.15 –Приклад використання API

Третій крок представляє документацію по інтеграції данного API до в аплікації, а також електронний адрес служби підтримки, для зв’язку у випадку помилок або некоректної роботи. Служба підтримки відкликається до 10 хвилин, що свідчить про швидку підтримку (рис 3.16)

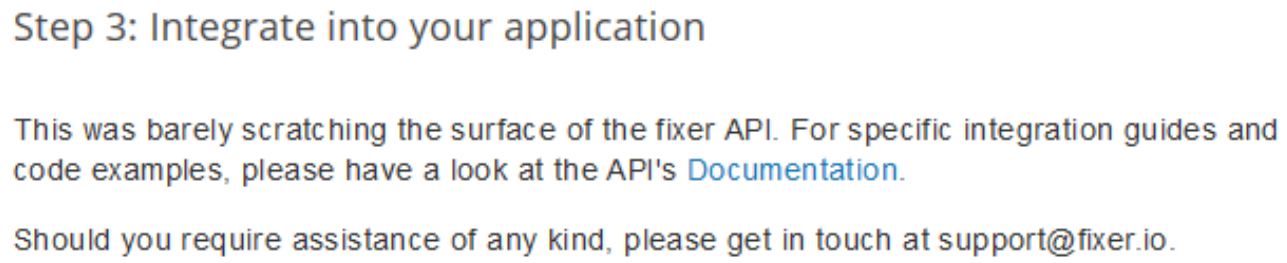

Рисунок 3.16 – Приклад використання API.

FixerAPI

Працюючи на 15 + джерелах даних обмінного курсу, API Fixer здатний надавати дані обмінного курсу реального часу для 170 світових валют. API поставляється з кількома кінцевими точками, кожна з яких обслуговує різні випадки використання.

Функціональні можливості кінцевої точки включають отримання останніх даних обмінного курсу для всіх або певного набору валют, перетворення суми з однієї валюти в іншу, отримання даних часових рядків для однієї або декількох валют і запит API для щоденних даних про коливання.

Наступне API, яке викоистовується у данній роботі є API від приватбанку, і для його використання необхідно зайти на api.privatbank.ua і виконати реєстрацію для його надання. Особливістю його використання є обов’язкова потреба бути зареєстрованим у приватбанку і мати доступ до інтернет-банкінгу. Без аккаунта у приватбанку використовувати API буде неможливо, що може створити деякі проблеми з його використанням, оскільки розробник або група розробників можуть не хотіти розголошувати свою персональну інформацію.

Також данне API має роздільність на фізичне та юридичне лице. Оскільки даний проект не є розробкою для продажу а використовується тільки для демонстрації його можливостей вхід виконується через фізичне лице (рис 3.17).

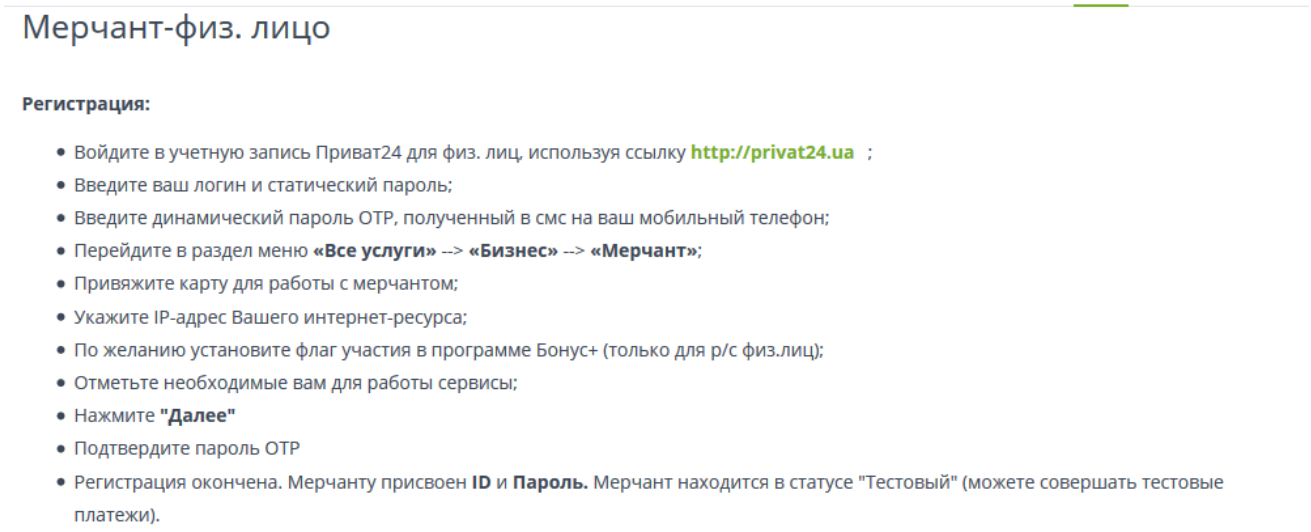

Рисунок 3.17 – Інструкція використанн API приватбанку

Після виконування усіх інструкцій вище ми отримуємо ключ до API, і вносимо його в нашу программу, як це зроблено з попереднім API.

Третє API яке використовується є із сервісу Finance.ua. Данний сервіс як і попередні працює з використанням технологій JSON, який дозволяє отримувати дані без перезагрузки чатбота.

Після переходу на сторінку використання API нас запрошує логін і пароль. У випадку із цим API немає потребності отримування ключа, а використання логіну і паролю користувача. Данне варіанти API є дуже розширеним і має безліч необхідних настройок, опису помилок і роботи з ним. Також наведені приклади роботи (рис 3.18).

```
{
   "id":ID_REQUEST,
   "service":SERVICE_NAME,
   "method":METHOD_NAME,
   "params":[]
}
```

Рисунок 3.18 – Стандартний запит API Finance

Для авторизації в системі використовується логін і пароль, який прописується в аплікації (рис 3.19)

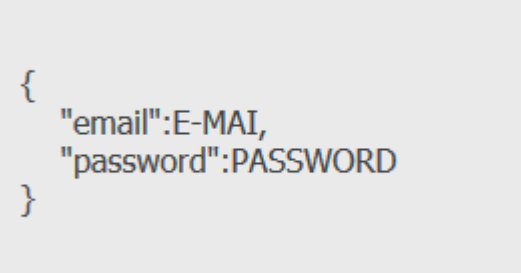

Рисунок 3.19 – Авторизація API Finance

В данному проекті данні уже є занесені і немає потребності в авторизації, при роботі чатбота:

```
{
"id":1,
"service":"sauth",
"method":"auth",
"params":[
    {
        "email":"vladislavkryslatij@finanse.ua",
        "password":"123123"
    }
]
}
```

В данному фрагменті коду виконується авторизація користувача з введеним паролем в параметрах.

Головною перевагою також є робота з токенами, що дозволяє безпечно отримувати данні та відправляти запити на данний ресурс

```
{
    "id" : 11,
```

```
    "service" : finance,
    "method" : METHOD_NAME,
    "params" :
    [        {
            "token" : ak1s3sj4fb6k7s9s9d89s,
            "idAccount" : 11,
            "idUser" : Vladislav
        },
```

Після перевірки коректної роботи наступним кроком є відправлення запиту на валюту яка нас цікавить а також дату запиту з поточним станом, в коді це записано як:

```
{
          "id" : 11,
    "service" : finance,
    "method" : METHOD_NAME,
    "params" :
    [        {
            "token" : ak1s3sj4fb6k7s9s9d89s,
            "idAccount" : 11,
            "idUser" : Vladislav
        },

  "timestamp": 1436284516,
  "source": "USD",
  "quotes": {
      "USDAUD": 1.345352401,
      "USDCAD": 1.27373397,
      "USDCHF": 0.947845302,
      "USDEUR": 0.91313905,
      "USDGBP": 0.647603397,
```

```
    } }
```

ВИСНОВКИ

В даній дипломній роботі спроектовано та реалізовано чатбот курсу валют.

В процесі дослідження предметної області було систематизовано вхідні дані та створено специфікацію системи..

Розроблений чатбот дозволяє переглядати данні по курсах валют у реальному часі .

При проектуванні виконано всі необхідні економічні розрахунки і досліджено, що розробка та впровадження системи є доцільним та економічно вигідним.

При опрацюванні питань охорони праці проаналізовано потенційно небезпечні та шкідливі виробничі фактори, проведено розрахунок штучної освітленості приміщення, де використовується комп'ютерна техніка.

Отже в даній дипломній роботі спроектовано повністю функціональний, логічно побудований чатбот, з використанням C# та ASP.NET, який повністю задовільняє поставлені вимоги.

ПЕРЕЛІК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Руководство по ASP.NET Core 2.0 / Metanit -Дата поновлення: 15.11.2017. URL: https://metanit.com/sharp/aspnet5.

2. Руководство по JQuery / Metanit -Дата поновлення: 15.11.2017. URL: https://metanit.com/web/jquery.

3. Руководство по HTML (дата звернення: 21.04.2018) https://metanit.com/web/html5

4. Джеффрі Ріхтер, CLRviaC # -М.: Російська редакція 2008р -636С

5. Бот для біржі криптовалюта / Робимо успішний бізнес-Дата поновлення: 04.10.2017. URL: https://delen.ru/onlayn-biznes/bot-dlya-birzhi-kriptovalyut.html.

6. Топ6 арбітражнихботовbitcoin / Коінмайнінг-Датаобновленія: 04.03.2017. URL: https: //coinsmining.ru/топ-6-арбітражних-ботов/

7. Правила охорони праці під час експлуатації електроннообчислювальних машин : ДНАОП 0.00.-1.31-99. [Чинний від 1999-02-10] – К. , 1999. – 28 с. (Нормативні акти про охорону праці)

8. Правила безпечної експлуатації електроустановок споживачів : ДНАОП 0.00-1.21-98. [Чинний від 1998-02-20] – К. , 1999. – 54 с. (Нормативні акти про охорону праці)

9. Правила охорони праці під час експлуатації електронно-обчислювальних машин : НПАОП 0.00-1.28-10. [Чинний від 2010-03-26] – К. , 2011. – 93 с. (Нормативні акти про охорону праці)

10. Санітарні норми мікроклімату виробничих приміщень : ДСН 3.3.6.042-99. [Чинний від 1999-12-01] – К. , 2001. – 86 с. (Санітарні норми і правила)

11. Зубенко В.В. Програмування : навчальний посібник (гриф МОН України) / В.В. Зубенко, Л.Л. Омельчук. - К. : ВПЦ "Київський університет", 2011. - 623 с.

12. Йенсен К., Вирт Н. Руководство для пользователя и описание языка. – М., 1982. 151 с.

13. Керниган Б., Ритчи Д., Фьюэр А. Язык программирования Си. – М.: Финансы и статистика, 1985.

14. Ковалюк Т.В.Основи програмування. – К.:Вид. група BHV, 2005. – 384 с.

15. Кристиан Нейгел, Билл Ивьен, Джей Глинн, Карли Уотсон, Морган Скиннер - C# 2008 и платформа .Net 3.5 для профессионалов.

16. Лавров С.С. Програмирование. Математические основи, средства,теория / С.С. Лавров. - СПб. : БХВ-Петербург,2001. - 251с.

17. Непейвода Н.Н. Основания програмирования : учеб. пособие / Н.Н. Непейвода, И.Н. Скопин. - Ижевск, 2003.

18. Нікітченко М.С. Теоретичні основи програмування : навчальний посібник / М.С Нікітченко - Ніжин : Видавництво НДУ імені Миколи Гоголя, 2010. - 121с.

19. Пратт Т., Зелкович М., Языки программирования: разработка и реализация. – Спб.: Питер, 2002.-688 с.

20. Проценко В.С., Чаленко П.Й., Сорока Р.А. Техника программирования. – К.: Вища школа. 1990.- 183с.

21. Симон Робинсон, Олли Корнес, Джей Глинн, Бартон Харвей, Крейг Макквин, Джерод Моемека, Кристиан Нагель, Морган Скиннер, Карли Ватсон - C# для профессионалов Том I.

22. Симон Робинсон, Олли Корнес, Джей Глинн, Бартон Харвей, Крейг Макквин, Джерод Моемека, Кристиан Нагель, Морган Скиннер, Карли Ватсон - C# для профессионалов Том II.

23. Стефенс Д. Р. C++. Сборник рецептов. — КУДИЦ-ПРЕСС, 2007. — 624 с.

24. Страуструп Б. Программирование: принципы и практика использования C++, исправленное издание = Programming: Principles and Practice Using C++. — М.: Вильямс, 2011.

25. Страуструп Б. Язык программирования C++. Специальное издание = The C++ programming language. Special edition. — М.: Бином-Пресс, 2007. —

1104 с.

26. Уинер Р. Язык Turbo Си: Пер. с англ. – М.: Мир, 1991. – 384 с.: ил.

27. Уэйт М., Прата С., Мартин Д. Язык Си. Руководство для начинающих: Пер. с англ. – М.: Мир, 1988. – 512 с.: ил.

28. Фокс Дж. Программное обеспечение и его разработка: Пер. с англ. – М.: Мир, 1985. – 368 с.: ил.

29. Фролов А.В., Фролов Г.В. - Язык C#. Самоучитель. – М.: Диалог-Мифи, 2003.-560с.

30. Язык компьютера/ Под ред. и предисл. В.М. курочкина. Пер. с англ. – М.: 1989. – 240 с.: ил.

БІБЛІОГРАФІЧНА ДОВІДКА

Тема дипломної роботи: Розробка чат-бота для перевірки курсу валют. Back-end частина.

Обсяг пояснювальної записки: 60 аркушів.

Перелік графічних матеріалів:

- рисунків 13
- додатків 5 на 8 аркушах.

Дата закінчення дипломного проекту: “__”_____2019р.

Студент–дипломник_____

(підпис)(розшифровка підпису)

Додаток А

Код файлу Program.cs

```
   private static ExchangeRate GetRate(string from, string to, DateTime? date = null)
        {
            from = from.ToUpper();
            to = to.ToUpper();

            if (!Symbols.IsValid(from))
                throw new ArgumentException("Symbol not found for provided currency",
"from");

            if (!Symbols.IsValid(to))
                throw new ArgumentException("Symbol not found for provided currency",
"to");

            var url = GetFixerUrl(date);

            using (var client = new HttpClient())
            {
                var response = client.GetAsync(url).Result;
                response.EnsureSuccessStatusCode();

                return ParseData(response.Content.ReadAsStringAsync().Result, from, to);
            }
        }

        private static async Task<ExchangeRate> GetRateAsync(string from, string to,
DateTime? date = null)
        {
            from = from.ToUpper();
            to = to.ToUpper();

            if (!Symbols.IsValid(from))
                throw new ArgumentException("Symbol not found for provided currency",
"from");

            if (!Symbols.IsValid(to))
                throw new ArgumentException("Symbol not found for provided currency",
"to");

            var url = GetFixerUrl(date);
```

```
            using (var client = new HttpClient())
            {
                var response = await client.GetAsync(url);
                response.EnsureSuccessStatusCode();

                return ParseData(await response.Content.ReadAsStringAsync(), from, to);
            }
        }

        private static ExchangeRate ParseData(string data, string from, string to)
        {
            // Parse JSON
            var root = JObject.Parse(data);

            var rates = root.Value<JObject>("rates");
            var fromRate = rates.Value<double>(from);
            var toRate = rates.Value<double>(to);

            var rate = toRate / fromRate;

            // Parse returned date
            // Note: This may be different to the requested date as Fixer will return the
closest available
            var returnedDate = DateTime.ParseExact(root.Value<string>("date"), "yyyy-MM-
dd",
                System.Globalization.CultureInfo.InvariantCulture);

            return new ExchangeRate(from, to, rate, returnedDate);
        }

        private static string GetFixerUrl(DateTime? date = null)
        {
            var dateString = date.HasValue ? date.Value.ToString("yyyy-MM-dd") :
"latest";

            return $"{BaseUri}{dateString}?access_key={ApiKey}";
        }


                                                }
```

Додаток Б

FixerIoClien

```
using System;
using System.Collections.Generic;
using System.Net.Http;
using System.Threading.Tasks;
using Newtonsoft.Json;

namespace TelegramBot
{
    public class FixerIoClient
    {
        private readonly string _baseCurrency;
        private readonly bool _https;
        private ICollection<string> _symbols;

        public FixerIoClient()
        {
            _https = false;
            _baseCurrency = Symbol.EUR.ToString();
        }

        public FixerIoClient(Symbol baseCurrency)
        {
            _baseCurrency = baseCurrency.ToString();
        }

        public FixerIoClient(Symbol baseCurrency, IEnumerable<Symbol>
symbols)
        {
            _baseCurrency = baseCurrency.ToString();
            SetSymbols(symbols);
```

```
        }

        public FixerIoClient(Symbol baseCurrency, IEnumerable<Symbol>
symbols, bool https)
        {
            _baseCurrency = baseCurrency.ToString();
            SetSymbols(symbols);
            _https = https;
        }

        private void SetSymbols(IEnumerable<Symbol> symbols)
        {
            _symbols = new List<string>();

            foreach (var symbol in symbols)
            {
                _symbols.Add(symbol.ToString());
            }
        }

        public async Task<Quote> GetLatestAsync()
        {
            return await Request();
        }

        public Quote GetLatest()
        {
            return Request().Result;
        }

        public async Task<Quote> GetForDateAsync(DateTime date)
        {
            return await Request(date, null);
        }
```

```
        public Quote GetForDate(DateTime date)
        {
            return Request(date, null).Result;
        }

        public async Task<decimal> ConvertAsync(Symbol from, Symbol
to)
        {
            return (await Request(null, from)).Rates[to.ToString()];
        }

        public decimal Convert(Symbol from, Symbol to)
        {
            return Request(null, from).Result.Rates[to.ToString()];
        }

        public async Task<decimal> ConvertAsync(Symbol from, Symbol
to, DateTime date)
        {
            return (await Request(date, from)).Rates[to.ToString()];
        }

        public decimal Convert(Symbol from, Symbol to, DateTime date)
        {
            return Request(date, from).Result.Rates[to.ToString()];
        }

        private async Task<Quote> Request()
        {
            return await Request(null, null);
        }

        private string PrepareRequest(DateTime? date, Symbol? from)
```

```
        {
            var currency = (from.HasValue ? from.ToString() :
_baseCurrency);
            var dateString = date == null ? "latest" :
date.Value.ToString("yyyy-MM-dd");
            var queryString = $"{dateString}?base={currency}";

            if (_symbols != null && from == null)
            {
                queryString += $"&symbols={string.Join(",",
_symbols)}";
            }

            return queryString;
        }

        private async Task<Quote> Request(DateTime? date, Symbol?
from)
        {
            using (var httpClient = new HttpClient())
            {
                httpClient.BaseAddress = new Uri($"{(_https ? "https"
: "http")}://api.fixer.io/");

                var queryString = PrepareRequest(date, from);
                var response = await httpClient.GetAsync(queryString);

                if (response.IsSuccessStatusCode)
                {
                    var stringResponse = await
response.Content.ReadAsStringAsync();
                    return
JsonConvert.DeserializeObject<Quote>(stringResponse);}
```

Додаток В

Код моделей

```
using System;

namespace TelegramBot
{
    public class ExchangeRate
    {
        public ExchangeRate(string from, string to, double rate,
DateTime date)
        {
            From = from;
            To = to;
            Rate = rate;
            Date = date;
        }

        public string From { get; }

        public string To { get; }

        public double Rate { get; }

        public DateTime Date { get; set; }

        public double Convert(double amount)
        {
            return amount * Rate;
        }
    }
```

Додаток Г

Код Quote

```
using System;
using System.Collections.Generic;

namespace TelegramBot
{
    public class Quote
    {
        public String Base { get; set; }
        public DateTime Date { get; set; }
        public IDictionary<string, decimal> Rates { get; set; }
    }
}
```

Додаток Д

Код конфігурацій валют

```
using System.Runtime.Serialization;
namespace TelegramBot {
    public enum Symbol
    {
        AUD,
        BGN,
        BRL,
        CAD,
        CHF,
        CNY,
        CZK,
        DKK,
        EUR,
        GBP,
        HKD,
        HRK,
        HUF,
        IDR,
        ILS,
        INR,
        JPY,
        KRW,
        MXN,
        MYR,
        NOK,
        NZD,
        PHP,
        PLN,
        RON,
        RUB,
        SEK,
        SGD,
        THB,
        TRY,
        USD,
        ZAR
    }
}
```