МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ КІБЕРБЕЗПЕКИ, КОМП'ЮТЕРНОЇ
ТА ПРОГРАМНОЇ ІНЖЕНЕРІЇ
КАФЕДРА КОМП'ЮТЕРНИХ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач випускової кафедри
_____ Аліна САВЧЕНКО
«___»_____2022 р.

КВАЛІФІКАЦІЙНА РОБОТА
(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ МАГІСТР
ЗА ОСВІТНЬО-ПРОФЕСІЙНОЮ ПРОГРАМОЮ
«ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ПРОЕКТУВАННЯ»

Тема: «Телеграм-бот для прийняття рішень децентралізованою автономною організацією»

Виконавець: Дмитро БЛЕЙЧИК

Керівник: к.пед.н., доцент Юрій СІНЬКО

Нормоконтролер: к.т.н., доцент Олена ТОЛСТІКОВА

КИЇВ 2022

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет *кібербезпеки, комп'ютерної та програмної інженерії*
Кафедра *Комп'ютерних інформаційних технологій*
Спеціальність *122 «Комп'ютерні науки»*
Освітньо-професійна програма *«Інформаційні технології проектування»*

ЗАТВЕРДЖУЮ:
завідувач кафедри КІТ
_____Аліна САВЧЕНКО
(підпис)
«____»_____2022 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи
Блейчика Дмитра Олександровича
(ПІБ випускника)

1. Тема роботи: «Телеграм-бот для прийняття рішень децентралізованою автономною організацією» затверджена наказом ректора № 1774/ст від 28.09.2022р.
2. Термін виконання роботи: з 26 вересня 2022 року по 27 листопада 2022 року.
3. Вихідні дані до роботи: Телеграм-бот верифікації належного користувача.
4. Зміст пояснювальної записки: 1. Децентралізована автономна організація. 2. Проектування Телеграм-бота. 3. Розробка Телеграм-бота.
5. Перелік обов'язкового ілюстративного матеріалу: 1. Актуальність теми. 2. Завдання дослідження. 3. Месенджер Телеграм. 4. TON. 5. NFT. 6. DAO. 7. PyCharm. 8. Розробка Телеграм-Бота. 9. Висновки.

6. Календарний план-графік

№ з/п	Завдання	Термін виконання	Підпис керівника
1.	Аналіз предметної області та огляд аналогів. Написання 1 розділу, представлення керівнику	26.09.2022-16.10.2022	
2.	Вибір та опис використаних технологій. Написання 2 розділу, представлення керівнику	17.10.2022-30.10.2022	
3.	Розробка бота за допомогою обраного стеку технологій. Написання 3 розділу, представлення керівнику	31.10.2022-14.11.2022	
4.	Загальне редагування та друк пояснювальної записки	15.11.2022-20.11.2022	
5.	Проходження нормоконтролю, перепліт пояснювальної записки.	16.11.2022-20.10.2022	
6.	Розробка тексту доповіді. Оформлення графічного матеріалу для презентації	20.11.2022-22.11.2022	

7. Дата видачі завдання 26.09.2022р.

Керівник кваліфіканої роботи _____ Юрій СІНЬКО
(підпис керівника)

Завдання прийняв до виконання _____ Дмитро БЛЕЙЧИК
(підпис випускника)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи на тему: «Телеграм-бот для прийняття рішень децентралізованою автономною організацією» містить: 94 сторінки, 56 рисунків, 1 таблиця, 21 інформаційних джерел.

Об'єкт дослідження – бот для верифікації належних користувачів, розроблений на мові програмування Python для месенджеру Телеграм.

Предмет дослідження – Telegram-бот для підтвердження володіння невзаємозамінним токеном проекту.

Мета кваліфікаційної роботи – отримати готового Telegram-бота для поліпшення процесу авторизації на основі факту володіння токеном колекції проекту, з використанням комплексу сучасних технологій.

Методи дослідження – мова програмування Python, кросплатформове інтегроване середовище розробки PyCharm, месенджер Telegram, фреймворк Pyrogram, асинхронний фреймворк Aiogram.

Результати кваліфікаційної роботи рекомендується використовувати для створення власного проекту на основі колекції невзаємозамінних токенів у блокчейні The Open Network.

TELEGRAM-БОТ, PYROGRAM, AIOGRAM, PYTHON, PYCHARM, THE OPEN NETWORK, TON, БЛОКЧЕЙН.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

DAO *(Decentralized Autonomous Organization)*	—	Автономна система в блокчейні
BTC *(Bitcoin)*	—	Децентралізована цифрова валюта
UTXO *(Unspent Transaction Output)*	—	Вихід невитрачених транзакцій
API *(Application programming interface)*	—	Програмний інтерфейс
ETH *(Ethereum)*	—	Тип цифрової валюти
NFT *(Non-fungible token)*	—	Невзаємозамінний токен
DeFi *(Decentralized Finance)*	—	Аналоги традиційних фінансових інструментів
BAYC *(Bored Ape Yacht Club)*	—	Унікальні цифрові предмети колекціонування
CDP *(*Collateralized Debt Positions*)*	—	Позиція, створена блокуванням застави
IDLE *(Integrated Development and Learning Environment)*	—	Інтегроване середовище розробки та навчання мовою Python

ВСТУП

Telegram — це служба обміну повідомленнями в чаті з додатковою перевагою шифрування даних, безпеки та конфіденційності. Telegram використав те, що якщо новий контакт підписується на бізнес-канал, компанія не отримує доступу до жодної контактної чи особистої інформації користувача.[1] У той же час користувачі Telegram можуть шукати назви компаній і підписуватися на проекти через пошук. Ця домовленість працює таким чином, що клієнти можуть зв'язуватися з компаніями, якщо хочуть, але компанії не можуть налагодити контакт з усіма клієнтами без їхнього дозволу.

Бот у Telegram — це програма, що пропонує функції та засоби автоматизації, які користувачі Telegram можуть інтегрувати у свої чати, канали чи групи. Їх можна налаштувати за допомогою власного конструктора ботів Telegram – Botfather, або API ботів Telegram. Останній пропонує більше можливостей для розробників. Те, як Telegram надихає на використання ботів, лише показує, який у них потенціал. Варіанти використання, особливо для програм обміну повідомленнями, надзвичайно різноманітні. Вони прості у використанні, і приватні користувачі, а також адміністратори та компанії отримують від них користь.[2]

У кваліфікаційній роботі будуть використані сторонні бібліотеки. Бібліотеки Python — це набір додаткових модульних компонентів коду для «зміїної мови», розроблених для певних делікатних завдань. Щоб керувати ними, потрібні спеціальні навички, оволодівши якими, можна зробити програмування на Python значно ефективнішим.

Для розширення функціональності стандартної бібліотеки Python потрібні сторонні бібліотечні модулі. Завдяки набору функцій, які вони пропонують, є можливість працювати над великими проектами поетапно та вирішувати різні складні завдання.

Актуальність теми кваліфікаційної роботи «Телеграм-бот для прийняття рішень децентралізованою автономною організацією» ґрунтується на тому, що Телеграм відносно новий месенджер, створений у 2013 році, а

боти були додані у 2015 році і з тих пір з ростом функціоналу Телеграму збільшували список своїх можливих удосконалень та поліпшень. Під ростом функціоналу мається на увазі розширення екосистеми Telegram.

Метою кваліфікаційної роботи є створення бота у Telegram для полегшення керуванням доступу до інформації у інвесторів.

Завдання дослідження:

- проаналізувати та описати різні блокчейн мережі;
- провести аналіз децентрацізованих автономних організацій;
- провести огляд існуючих ботів;
- проаналізувати та обрати середовище розробки;
- розробка Телеграм-бота.

Для досягнення поставленої мети й виконання завдань використано метод системно-структурного аналізу наукової літератури, який дав змогу показати особливості незмінного реєстру та звернень до нього телеграм боту, розроблених на мові програмування Python.

Наукова новизна роботи полягає у створенні нового способу взаємодії користувачів з проектом шляхом звернень до блокчейну за допомогою бота.

Проект має перспективу користуватись великим попитом у зв’язку з широкою доступністю, зручністю, зрозумілістю та простотою налаштування. У майбутньому планується розширення функціоналу бота, завдяки використанню модульної архітектури.

РОЗДІЛ 1
ДЕЦЕНТРАЛІЗОВАНІ АВТОНОМНІ ОРГАНІЗАЦІЇ

1.1. Використання Блокчейну

Блокчейн – це загальний незмінний реєстр, що полегшує процес запису транзакцій та відстеження активів у бізнес-мережі. Актив може бути матеріальним (будинок, машина, гроші, земля) чи нематеріальним (інтелектуальна власність, патенти, авторські права, брендинг). Практично все, що має цінність, можна відстежувати та продавати в мережі блокчейн, що знижує ризики та витрати для всіх учасників.

Чому блокчейн є важливим? Бізнес працює на інформації. Чим швидше вона буде отримана і чим точніше вона буде, тим краще. Блокчейн ідеально підходить для доставки цієї інформації, тому що він надає негайну, загальнодоступну та повністю прозору інформацію, що зберігається у незмінному реєстрі, до якого можуть отримати доступ лише уповноважені члени мережі. Мережа блокчейна може відстежувати замовлення, платежі, рахунки, виробництво та багато іншого. А оскільки учасники мають єдине уявлення про правду, ви можете бачити всі деталі транзакції від початку до кінця, що надає вам більшої впевненості, а також нових можливостей та ефективності.

1.1.1. Історія виникнення сучасного Блокчейну

У 2009 році Сатоші Накамото вперше впровадив на практиці децентралізовану валюту, поєднавши встановлені примітиви для управління власністю через криптографію з відкритим ключем із консенсусним алгоритмом для відстеження того, хто володіє монетами, відомим як «доказ роботи».

Механізм підтвердження роботи став проривом у просторі, оскільки він одночасно вирішив дві проблеми. По-перше, він забезпечив простий і помірно ефективний алгоритм консенсусу, що дозволив вузлам у мережі колективно узгодити набір канонічних оновлень стану реєстру біткойнів. По-друге, це забезпечило механізм вільного входу в процес консенсусу, вирішення політичної проблеми вирішення питання про те, хто має впливати на консенсус, і водночас запобігання атакам сивіл. Це робиться шляхом заміни формальних перешкод для участі, таких як вимога бути зареєстрованим як унікальний суб'єкт у певному списку, на економічні перешкоди – вага одного вузла в процесі консенсусного голосування прямо пропорційна обчислювальній потужності. що приносить вузол. З тих пір був запропонований альтернативний підхід під назвою proof-of-stake, який обчислює вагу вузла пропорційно його валютним запасам, а не обчислювальним ресурсам.

«Стан» у біткоїнах — це сукупність усіх монет (технічно «невитрачені виходи транзакцій» або UTXO), які були викарбувані та ще не витрачені, причому кожен UTXO має номінал і власника (визначається 20-байтовою адресою, яка по суті є криптографічним відкритим ключем fn1)(рис. 1.1). Транзакція містить один або більше вхідних даних, причому кожен вхід містить посилання на існуючий UTXO та криптографічний підпис, створений закритим ключем, пов'язаним з адресою власника, а також один або більше виходів, причому кожен вихід містить новий UTXO, який потрібно додати до стан.

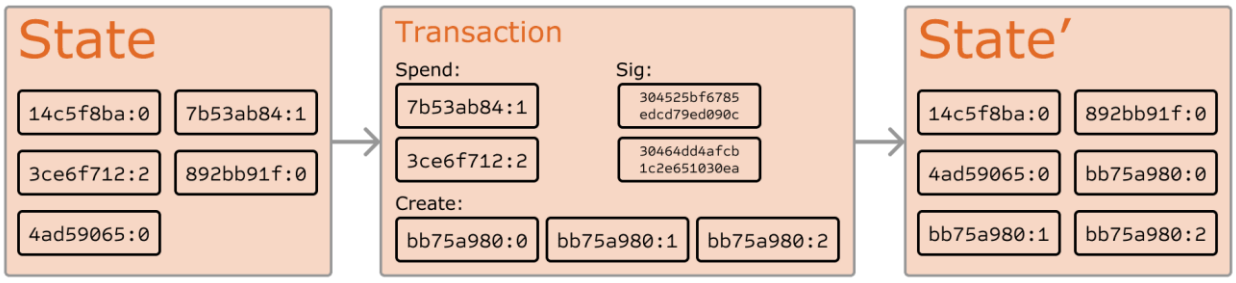

Рис. 1.1. Приклад стану у біткоїнах

Функцію переходу стану APPLY(S,TX) -> S' можна приблизно визначити наступним чином:

- для кожного входу в TX;
- якщо UTXO, на який посилається, не в S, повертає помилку;
- якщо наданий підпис не відповідає власнику UTXO, поверніть помилку;
- якщо сума номіналів усіх вхідних UTXO менша за суму номіналів усіх вихідних UTXO, поверніть помилку;
- повернення S із видаленням усіх вхідних UTXO та додаванням усіх вихідних UTXO.

Перша половина першого кроку не дозволяє відправникам транзакцій витрачати монети, яких не існує, друга половина першого кроку запобігає відправникам транзакцій витрачати монети інших людей, а другий крок забезпечує збереження вартості. Щоб використовувати це для оплати, протокол виглядає наступним чином. Припустімо, що Аліса хоче надіслати Бобу 11,7 BTC. По-перше, Аліса шукатиме набір доступних UTXO, якими вона володіє, на загальну суму щонайменше 11,7 BTC. Реально Аліса не зможе отримати рівно 11,7 BTC; скажіть, що найменше, яке вона може отримати, це 6+4+2=12. Потім вона створює транзакцію з цими трьома входами та двома виходами. Перший вихід становитиме 11,7 BTC з адресою Боба як його власника, а другий вихід буде рештою 0,3 BTC «зміна», причому власником буде сама Аліса.[3]

Якби ми мали доступ до надійної централізованої служби, реалізувати цю систему було б тривіально; його можна просто закодувати точно так, як описано, використовуючи жорсткий диск централізованого сервера для відстеження стану. Проте з біткоїнами ми намагаємося побудувати децентралізовану валютну систему, тому нам потрібно буде поєднати державну систему транзакцій із системою консенсусу, щоб гарантувати, що всі погоджуються щодо порядку транзакцій. Процес децентралізованого

консенсусу біткоїна вимагає, щоб вузли в мережі постійно намагалися створити пакети транзакцій, які називаються «блоками». Мережа створюватиме приблизно один блок кожні десять хвилин, причому кожен блок містить мітку часу, одноразовий номер, посилання (тобто хеш) попереднього блоку та список усіх транзакцій, які відбулися з попереднього блокувати. З часом це створює постійний, постійно зростаючий «блокчейн», який постійно оновлюється, щоб відобразити останній стан реєстру біткоїнів.

Алгоритм перевірки правильності блоку, виражений у цій парадигмі, такий:

- перевірте, чи існує і дійсний попередній блок, на який посилається блок;
- переконайтеся, що позначка часу блоку більша, ніж у попередньому blockfn2, і менша ніж на 2 години в майбутньому;
- перевірте, чи дійсне підтвердження роботи на блоці;
- нехай S[0] буде станом у кінці попереднього блоку;
- припустимо, що TX — це список транзакцій блоку з n транзакцій. Для всіх i в 0...n-1 встановіть S[i+1] = APPLY(S[i],TX[i]) Якщо будь-яка програма повертає помилку, вийдіть і поверніть false;
- поверніть істину та зареєструйте S[n] як стан у кінці цього блоку (рис. 1.2).
-

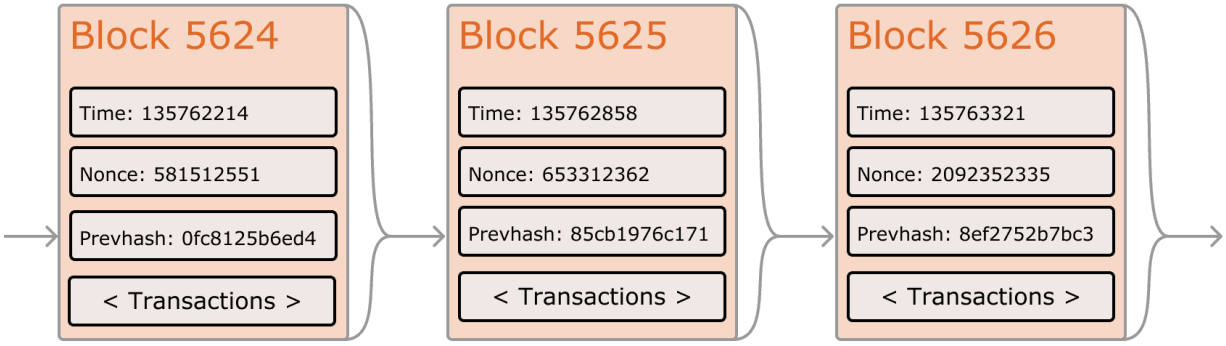

Рис. 1.2. Приклад ланцюга блоків біткоїна

По суті, кожна транзакція в блоці повинна забезпечувати дійсний перехід стану від того, що було канонічним станом до того, як транзакція була виконана, до якогось нового стану. Зверніть увагу, що стан жодним чином не кодується в блоці; це суто абстракція, яку має запам'ятати перевіряючий вузол, і її можна (безпечно) обчислити лише для будь-якого блоку, починаючи зі стану генезису та послідовно застосовуючи кожну транзакцію в кожному блоці. Крім того, зауважте, що порядок, у якому майнер включає транзакції в блок, має значення; якщо в блоці є дві транзакції A і B, так що B витрачає UTXO, створений A, тоді блок буде дійсним, якщо A стоїть перед B, але не інакше.

Єдина умова дійсності, представлена в наведеному вище списку, яка не зустрічається в інших системах, — це вимога «доказу роботи». Точна умова полягає в тому, що подвійний хеш SHA256 кожного блоку, який розглядається як 256-бітне число, має бути меншим за динамічно скориговану ціль, яка на момент написання цієї статті становить приблизно 2187. Мета цього полягає в тому, щоб зробити «жорстко» обчислювальне створення блоку, що запобігає зловмисникам sybil переробити весь блокчейн на свою користь. Оскільки SHA256 створена як абсолютно непередбачувана псевдовипадкова функція, єдиний спосіб створити дійсний блок — це просто метод проб і помилок, багаторазове збільшення nonce і перевірка відповідності нового хешу.

При поточній меті ~2187 мережа повинна зробити в середньому ~269 спроб, перш ніж буде знайдено дійсний блок; загалом ціль повторно калібрується мережею кожні 2016 блоків, так що в середньому новий блок створюється деяким вузлом мережі кожні десять хвилин. Для того, щоб компенсувати майнерам цю обчислювальну роботу, майнер кожного блоку має право включити транзакцію, яка дає собі 25 BTC нізвідки. Крім того, якщо будь-яка транзакція має більший загальний номінал на вході, ніж на виході, різниця також надходить майнеру як «комісія за транзакцію». До речі, це також єдиний механізм, за допомогою якого випускаються BTC; стан генезису взагалі не містив монет.

Щоб краще зрозуміти мету майнінгу, розглянемо, що відбувається у випадку зловмисного зловмисника. Оскільки відомо, що криптографія, що лежить в основі біткойна, безпечна, зловмисник націлиться на ту частину системи біткойн, яка не захищена криптографією безпосередньо: порядок транзакцій. Стратегія зловмисника проста:

- надіслати 100 BTC продавцю в обмін на певний продукт (бажано цифровий товар із швидкою доставкою);
- дочекатись доставки товару;
- провести ще одну транзакцію, відправивши ті самі 100 BTC собі;
- спробувати переконати мережу, що його транзакція з ним самим була першою.

Після виконання кроку (1) через кілька хвилин якийсь майнер включить транзакцію в блок, скажімо, блок номер 270000. Приблизно через годину після цього блоку до ланцюжка буде додано ще п’ять блоків, з кожним з ті блоки, які опосередковано вказують на транзакцію і таким чином «підтверджують» її. На цьому етапі продавець прийме платіж як завершений і доставить товар; оскільки ми припускаємо, що це цифровий товар, доставка здійснюється миттєво. Тепер зловмисник створює ще одну транзакцію, надсилаючи собі 100 BTC. Якщо зловмисник просто випускає його в дику природу, транзакцію не буде оброблено; майнери спробують запустити APPLY(S,TX) і помітять, що TX використовує UTXO, якого більше немає в стані. Тому замість цього зловмисник створює «розгалуження» блокчейну, починаючи з майнінгу іншої версії блоку 270000, яка вказує на той самий блок 269999 як батьківський, але з новою транзакцією замість старої. Оскільки дані блоку відрізняються, це вимагає повторного виконання підтвердження роботи. Крім того, нова версія зловмисника блоку 270000 має інший хеш, тому вихідні блоки 270001-270005 не «вказують» на нього; таким чином, оригінальний ланцюжок і новий ланцюг зловмисника повністю окремі. Правило полягає в тому, що у форку найдовший блокчейн вважається істинним, тому законні майнери працюватимуть на ланцюжку 270005, тоді як зловмисник працює над ланцюжком 270000. Для

того, щоб зловмисник зробив свій блокчейн найдовшим, він повинен мати більше обчислювальної потужності, ніж решта мережі разом узяті, щоб наздогнати (отже, «51% атаки»).

Ключові елементи блокчейна:

1. Технологія розподіленого реєстру

Усі учасники мережі мають доступ до розподіленого реєстру та його незмінного запису транзакцій. У цьому загальному реєстрі транзакції записуються лише один раз, що унеможливлює дублювання зусиль, типове для традиційних бізнес-мереж.

2. Незмінні записи

Жоден учасник не може змінити або підробити транзакцію після того, як її було записано до загального реєстру. Якщо запис транзакції містить помилку, необхідно додати нову транзакцію, щоб усунути помилку, після чого обидві транзакції буде видно.

3. Смарт-контракти

Для прискорення транзакцій набір правил, званий смарт-контрактом, зберігається в блокчейні та виконується автоматично. Смарт-контракт може визначати умови передачі корпоративних облігацій, включати умови оплати страхування подорожей та багато іншого.

Операціоністи часто витрачають зусилля на дублювання записів та сторонні перевірки. Системи обліку можуть бути вразливими для шахрайства та кібератак. Обмежена прозорість може сповільнити перевірку даних. А з появою Інтернету речей обсяги транзакцій різко зросли. Все це уповільнює бізнес, зменшує прибуток і означає, що нам потрібний найкращий вибір — блокчейн.

Більше довіри

Завдяки блокчейну, як члену мережі лише для учасників, ви можете бути впевнені, що отримуєте точні та своєчасні дані та що ваші конфіденційні записи в блокчейні будуть доступні лише тим учасникам мережі, яким ви спеціально надали доступ.

Підвищена безпека

Від усіх учасників мережі потрібен консенсус щодо точності даних, і всі перевірені транзакції незмінні, оскільки записуються постійно. Ніхто навіть системний адміністратор не може видалити транзакцію.

Більше ефективності

Завдяки розподіленому реєстру, що спільно використовується членами мережі, виключається марнування часу на погодження записів. А для прискорення транзакцій набір правил, який називають смарт-контрактом, може зберігатися в блокчейні і виконуватися автоматично.[4]

1.2. Визначення ДАО

ДАО – це децентралізована автономна організація, яка керується за допомогою смарт-контрактів. Останні відповідають за функції голосування, фінансового менеджменту та організації робочих потоків.

За певної кількості співробітників те, що може починатися як монолітний груповий чат, органічно розпадається на робочі групи чи функціональні комітети, у яких члени беруть він відповідальність за цю віху.

Назви та деталі цих груп різняться залежно від DAO, але ми можемо приблизно згрупувати їх за їхніми операційними функціями (рис. 1.3).

Інтерес до подібних структур із боку криптовалютної спільноти зростає. Нові ДАО з'являються практично щодня. Щоб приєднатися до них, люди кидають добре оплачувані роботи, а інвестори вкладають у сектор мільйони доларів.

Децентралізація означає відсутність лідера. Швидше, більше людей мають право виявляти ініціативу для досягнення спільних цілей: дороговказ DAO.

Рис.1.3. Групи розподілені за операційними функціями

Добре підтримувана документація, в якій детально описується структура DAO та над чим працює кожна робоча група, допомагає потенційним учасникам зрозуміти, як їхні навички можуть відповідати потребам DAO. Звичайно, мотивація для участі має бути насамперед. Ось де починається подорож інвестора.[5]

1.3. Історія ДАО

Розберемось хто вигадав концепцію ДАО, які види децентралізованих організацій бувають, і для чого люди об'єднуються у розподілені спільноти.

Концепція ДАО виникла близько дев'яти років тому, коли засновник блокчейн-платформи EOS Ден Ларімер запровадив термін «децентралізована автономна корпорація». Він описав її як структуру, статут якої «визначається вихідним кодом».

Віталік Бутерін розвинув цю думку, припустивши, що кодова база може визначати операційну діяльність організації, її місію та шляхи досягнення останньої.

В оригінальному white paper Ethereum він написав:

«The general concept of a "decentralized autonomous organization" is that of a virtual entity that has a certain set of members or shareholders which, perhaps with a 67% majority, have the right to spend the entity's funds and modify its code. The members would collectively decide on how the organization should allocate its funds. Methods for allocating a DAO's funds could range from bounties, salaries to even more exotic mechanisms such as an internal currency to reward work. This essentially replicates the legal trappings of a traditional company or nonprofit but using only cryptographic blockchain technology for enforcement. So far much of the talk around DAOs has been around the "capitalist" model of a "decentralized autonomous corporation" (DAC) with dividend-receiving shareholders and tradable shares; an alternative, perhaps described as a "decentralized autonomous community", would have all members have an equal share in the decision making and require 67% of existing members to agree to add or remove a member. The requirement that one person can only have one membership would then need to be enforced collectively by the group.».[6]

У перекладі:

«Загальна концепція полягає в тому, що віртуальна організація має певний набір учасників або акціонерів, які, можливо, отримавши більшість у 67% [голосів], мають право витрачати її кошти та змінювати код. Методи розподілу коштів ДАО можуть змінюватись від винагороди та зарплат до більш екзотичних механізмів таких як використання внутрішньої валюти для заохочення роботи. По суті це повторює юридичні атрибути традиційної компанії або НКО, але з використанням технології блокчейн».

Через декілька років з'явилася перша ДАО і називалася The DAO. Вона була децентралізованим аналогом краудфандингової платформи, автономним венчурним фондом, що управлявся колективно всіма інвесторами. За розробкою проекту стояв німецький стартап Slock.it.

The DAO визначила ключові особливості сучасних ДАО. Вона мала чотири важливі характеристики:

- неупередженість у відборі учасників. Приєднатися до управління спільними коштами фонду міг будь-хто. Єдина умова – володіння токеном DAO;

- гнучка структура. Система голосувань дозволяла підтримувати пропозиції будь-якого характеру - від цільових інвестицій та благодійності до створення корисних для організації продуктів;

- можливість отримання прибутку. Організація могла отримувати прибуток від своїх ініціатив. Її направляли на розвиток проекту або просто конвертували в DAO і розподіляли між учасниками;

- справедлива модель управління. Утримувачі DAO мали право голосу під час прийняття важливих рішень, пов'язаних із управлінням організацією чи внутрішніми проектами.

The DAO залучила 11,5 млн. ETH (~$150 млн. за курсом на той момент), що становило приблизно 14% загальної пропозиції Ethereum. Її історія закінчилася крахом — через помилку в коді проект втратив понад $50 млн. Однак саме ця ДАО заклала основу для подальшого розвитку концепції.

Загальний обсяг коштів під управлінням 1076 децентралізованих організацій, що відстежуються DeepDAO, становить $10 млрд, а сукупна кількість учасників перевищує 1,7 млн.

Прихильники моделі ДАО вважають, що самоорганізовані співтовариства можуть забезпечити новий рівень соціальної координації без централізованого керівництва.

Люди групуються в ДАО, щоб спільно створювати, розподіляти та отримувати ресурси в рамках спільної місії. Остання може бути обумовлена широким спектром цілей: від ефективнішого управління DeFi-протоколом до колекціонування NFT.[7]

1.4. Різновиди ДАО. Аналіз кожного виду

Умовно всі децентралізовані організації поділяються на великі категорії: технічно орієнтовані і суспільно орієнтовані.

Перші зосереджені на побудові цифрової інфраструктури, управлінні протоколами та розвитку галузевих сервісів. Вони схильні використати блокчейн для координації дій учасників.

Головною метою других є об'єднання людей та пошук нових способів взаємодії між ними. Управління такими організаціями може здійснюватися і поза децентралізованою мережею.

За зонами інтересів ДАО можна поділити на вісім груп:

- інфраструктурні;
- протокольні;
- сервісні;
- соціальні;
- інвестиційні;
- грантові;
- спільноти колекціонерів;
- медіа.

Інфраструктурні ДАО

Ці організації дозволяють людям створювати власні ДАО. Вони пропонують шаблони, фреймворки, готові смарт-контракти та API, які спрощують операційний процес у блокчейні.

Іншими словами, завдяки таким проектам організувати децентралізовану спільноту можуть навіть користувачі з обмеженими технічними навичками.

На ринку є понад 20 інфраструктурних ДАО, що пропонують різні рішення для організації спільнот у різних мережах. Серед яскравих представників сегменту:

Aragon - проект з відкритим вихідним кодом, що пропонує лінійку рішень для створення та управління ДАО на базі блокчейнів Ethereum, Polygon, Metis Andromeda чи Harmony,

Colony - блокчейн-платформа, що працює виключно у мережі Ethereum. Набір смарт-контрактів, що пропонується проектом, заточений під спільну

роботу груп людей — він забезпечує дотримання певних правил, не передбачаючи ієрархічну структуру.

DAOhaus - своєрідний оглядач для децентралізованих організацій, що дозволяє приєднуватися до існуючих спільнот та створювати власні на базі фреймворку Moloch. Проект працює в різних мережах – від Polygon до Ethereum та відповідних L2-рішень (рішення другого рівню блокчейну, своєрідні надбудови над основною мережею).

Протокольні ДАО

Основне завдання протокольних ДАО – передати управління проектом до рук спільноти. Організації випускають цифрові токени, власники яких можуть пропонувати, голосувати та вносити зміни до механіки роботи протоколу.

Подібні структури зазвичай вводять різні моделі майнінгу ліквідності та прибуткового фермерства, щоб залучити нових користувачів та підвищити децентралізацію.

До протокольних ДАО у тому числі належать:

Uniswap – найбільша децентралізована біржа на базі механізму автоматизованого маркетмейкера. Вектор розвитку проекту задають власники токена управління UNI.

MakerDAO - DeFi-платформа на базі блокчейну Ethereum, що дозволяє випускати стейблкоїн DAI під заставу цифрових активів. Токеном управління тут є MKR, який також бере участь у механізмі забезпечення стійкості DAI.

Compound – децентралізований лендинговий протокол, у якому відсоткові ставки формуються алгоритмічно на основі попиту та пропозиції. Також базується у мережі Ethereum.

Сервісні ДАО

Як було зазначено вище, децентралізовані організації управляють активами сукупної вартістю $10 млрд і представляють значний сектор криптовалютної промисловості. Сервісні ДАО – B2B-сегмент ринку. Вони надають іншим розподіленим спільнотам послуги у межах своїх компетенцій.

У деяких юрисдикціях ДАО можуть набути статусу юридичної особи, в інших — їм необхідна афілійована класична структура для взаємодії з традиційними компаніями та партнерами. Це створює попит на послуги фахівців у галузі права.

Найчастіше ДАО приймають пропозиції, у відповідність до якими ці питання вирішують юридичні консультанти. Однак у деяких випадках правовий аспект може забезпечувати смарт-контракт.

Розподілені організації, наприклад, LexDAO пропонують інструменти, які ДАО можуть використовувати замість дорогих і найчастіше недоступних юридичних послуг.

Крім цього існує сегмент «децентралізованих судів», що розвивається. Якщо між двома ДАО виникла суперечка, вони можуть звернутися до платформи на кшталт Kleros, яка є свого роду арбітражним сервісом. Подібним чином можуть вирішуватись і конфлікти всередині організації.

Існує ще безліч різновидів сервісних ДАО: від послуг аутсорсу (VectorDAO) до делегування управління проектом (Governor DAO).

Соціальні ДАО

Соціальні ДАО можна вважати продуктом еволюції форумів та групових чатів. Насамперед такі спільноти орієнтовані на соціальну взаємодію людей, а тому можуть бути створені навколо їхнього спільного хобі.

Наприклад, проект Bright Moments – це галерея NFT-мистецтва, організована як ДАО. Щоб приєднатися до спільноти, користувачеві необхідно придбати токен з колекції Crypto Citizens.

FiatLuxDAO - співтовариство випускників Каліфорнійського університету в Берклі. ДАО зосереджена на фінансуванні наукових та технологічних інновацій.

KrausHaus - об'єднання фанатів баскетболу, які вирішили створити ДАО для збору коштів, покупки та управління командою ліги НБА.

Інвестиційні ДАО

Першим представником цього гурту була вищезгадана The DAO. Подібні організації зосереджені на отриманні прибутку — їх учасники об'єднують капітал, знання та досвід для формування інвестиційного портфеля відповідно до спільних цілей.

Такі ДАО можуть фінансувати децентралізовані проекти на ранніх стадіях розвитку або вкладати кошти у цифрові токени.

CSP DAO — команда розробників, які використовують технологію блокчейн. Управління структурою здійснюється через токени NEBO, які також відповідають за розподіл інвестиційних часток.

TheLAO є духовною спадкоємицею The DAO. Це венчурний фонд, орієнтований на екосистему Ethereum. У портфелі запущеної у квітні 2020 року організації понад 30 проектів.

Усі учасники спільноти підписують операційну угоду. Взаємодія з традиційними інститутами забезпечує компанія OpenLaw - вона відповідає за підготовку податкової звітності та виконує різні адміністративні функції. Як і традиційні фонди, LAO стягує комісію за управління.

Деякі інвестиційні ДАО спрямовані на конкретний сегмент промисловості. Наприклад, стратегії Flamingo обмежені сектором незамінних токенів, включаючи фрагментовані NFT.

Грантові ДАО

Як і в попередньому випадку, учасники подібних організацій об'єднують капітал для досягнення спільних цілей, але не ставлять собі завдання максимізувати прибуток.

Найчастіше це «розподілені» грантові програми, орієнтовані підтримку екосистеми того чи іншого протокола:

Audius Grants забезпечує фінансування проектів та ініціатив екосистеми Audis;

MolochDAO є спільнотою, учасники якої фінансують розробку EIP.

Проте є й інші приклади — грантоотримувачами Mint Fund виступають цифрові художники, зацікавлені у створенні NFT. ДАО субсидує витрати на

випуск токенів (комісія за газ), а також надає консультаційну та іншу підтримку.

До цієї групи можна віднести благодійні організації, наприклад, Ukraine DAO.

ДАО колекціонерів

Бурхливий розвиток сегменту NFT та зростання вартості цифрових творів мистецтва викликало появу децентралізованих організацій, метою яких є колекціонування незамінних токенів. Учасники таких ДАО зазвичай володіють часткою у скарбниці спільноти.

Деякі такі ДАО орієнтовані на конкретні колекції. Наприклад, APE DAO підтримує екосистему Bored Ape Yacht Club. Інші вибирають предмети колекціонування, керуючись іншими принципами - herstoryDAO займається колекціонуванням цифрових робіт чорношкірих жінок.

Також існують організації, які купують NFT з метою їхнього шардингу (сегментування) та подальшого фракційного володіння. Прикладом таких структур є PartyDAO.

Окремим видом є ДАО, які орієнтовані на придбання фізичних предметів чи фінансування будь-яких громадських ініціатив. Так, у лютому 2022 року AssangeDAO зібрала понад $32 млн для сприяння звільненню із в'язниці засновника WikiLeaks Джуліана Ассанжа.

Медіа-ДАО

Метою децентралізованих організацій цієї групи є поширення новин та іншої інформації.

Деякі медіа-ДАО нагадують керовані спільнотою традиційні ЗМІ. Наприклад, власники токенів BANK організації Bankless мають право виносити на голосування різні пропозиції та ініціативи, а також бути авторами контенту за умови підтвердження відповідних навичок.

Mirror є блогерською платформою, що належить спільноті. Вона дозволяють авторам публікувати матеріали та монетизувати контент.

Ще один представник групи – Global Coin Research – є гібридною спільнотою авторів та інвесторів, орієнтованих на індустрію Web 3.0. Утримувачі токенів управління GCR отримують доступ до платформи спільних інвестицій та унікального аналітичного контенту. Автори останнього заробляють токени за написання матеріалів.

Ландшафт ДАО змінюється та розвивається, тому в майбутньому ми побачимо ще більше варіантів децентралізованої координації людей. Суспільно орієнтовані структури в цьому плані динамічніші, оскільки роблять ставку на соціальну взаємодію користувачів.

У технічно орієнтованих ДАО часто на кону стоять величезні суми грошей, тому ці організації вкрай «неповоротливі» — відповідальність за капітал потребує ончейн-управління з обов'язковою системою внесення та ратифікації пропозицій.

Децентралізована структура пропонує іншу модель управління та розвитку, яка, за певних умов, може бути набагато ефективнішою за традиційні механізми. Водночас через юридичні та інші проблеми створення ДАО не завжди виправдане — часом класична структура управління є більш життєздатною [8].

1.5. Недоліки та переваги ДАО

Децентралізована організація має кілька переваг:

- Більше вогневої могутності - будь-яка людина у світі може зробити свій внесок і покращити код.
- Дешевше - багато розробників роблять свій внесок у криптовалюти у вільний час, а це означає, що монети можуть бути створені без великих сум фінансування.
- Спільна робота - надання кожній людині права голосу веде до безлічі дискусій та передбачення кожної пропозиції, допомагаючи розробити монету.

Недоліки:

- Плоска структура – через відсутність чіткої авторитетної особи чи команди децентралізовані організації працюють повільніше, оскільки прийняття рішень займає більше часу.

- Розбіжності - коли співтовариство не погоджується в чомусь, воно часто може розколоти організацію на дві частини. Це сталося, коли Bitcoin Cash форкнувся від Bitcoina.

Прихильники Bitcoin Cash хотіли збільшити розмір блоку BTC вище 1 МБ. Ця проблема була настільки суперечливою, що поділила спільноту на дві частини.

1.6. Приклади ДАО

1.6.1. The DAO

DAO була цифровою децентралізованою автономною організацією та формою венчурного фонду, керованого інвесторами. Після запуску у квітні 2016 року через продаж токенів вона стала однією з найбільших краудфандингових кампаній в історії.

Перед DAO стояло завдання надати нову децентралізовану бізнес-модель для організації як комерційних, так і некомерційних підприємств. Він був створений у блокчейні Ethereum і не мав звичайної структури управління або ради директорів. Код DAO є відкритим кодом.

У червні 2016 року користувачі скористалися помилкою у коді The DAO, яка дозволила їм перевести третину коштів The DAO на дочірній рахунок. Співтовариство Ethereum неоднозначно вирішило провести хард-форк блокчейна Ethereum, щоб відновити приблизно всі кошти у вихідному контракті. Це розділило блокчейн Ethereum на дві гілки, кожна зі своєю власною криптовалютою, де початковий нерозгалужений ланцюжок блоків продовжився як Ethereum Classic.

До вересня 2016 року токен The DAO, відомий під прізвиськом DAO, був виключений зі списку основних криптовалютних бірж (таких як Poloniex та Kraken) і, по суті, припинив своє існування.

Хронологія подій

Комп'ютерний код з відкритим кодом, що лежить в основі організації, був написаний в основному Крістофом Джентшем і опублікований на GitHub , де інші учасники додавали і модифікували код. Саймон Дженч, брат Крістофа Дженча, також був залучений до цього підприємства.

DAO було запущено 30 квітня 2016 року о 01:42:58 +UTC на блоці Ethereum 1428757 з веб-сайтом та 28-денним краудсейлом для фінансування організації. Продаж токенів залучив понад 34 мільйони доларів США до 10 травня 2016 року та понад 50 мільйонів доларів США в ефірі (ETH) — токен цифрової вартості мережі Ethereum — до 12 травня та понад 100 мільйонів доларів США. до 15 травня 2016 р. 17 травня 2016 р. найбільший інвестор DAO володів менше ніж 4% усіх токенів DAO, а 100 найбільших власників володіли трохи більше 46% усіх токенів DAO. Вартість ефіру фонду на 21 травня 2016 р. склала понад 150 мільйонів доларів США від більш ніж 11 000 інвесторів.

Станом на травень 2016 року DAO залучила майже 14% усіх токенів ефіру, випущених на сьогоднішній день.

28 травня 2016 року токени DAO стали продаватися на різних криптовалютних біржах.

У документі, опублікованому в травні 2016 року, наголошується низка уразливостей безпеки, пов'язаних з The DAO, і рекомендується, щоб інвестори в The DAO утримувалися від вказівки DAO інвестувати в проекти доти, доки проблеми не будуть вирішені. Розробник Ethereum на GitHub вказав на недолік, пов'язаний з "рекурсивними викликами". 9 червня про це написав у своєму блозі Пітер Вессенес, засновник Blockchain Foundation. До 14 червня виправлення були запропоновані та очікували схвалення членами The DAO.

16 червня блогери, пов'язані з Ініціативою щодо криптовалютів та контрактів, привернули додаткову увагу до вразливостей рекурсивних викликів.

17 червня 2016 року DAO зазнала атак з використанням комбінації вразливостей, у тому числі пов'язаної з рекурсивними викликами, в результаті якої було переведено 3,6 мільйона ефірів — близько третини з 11,5 мільйонів ефірів, які були передані The DAO - оцінюється в той час приблизно 50 мільйонів доларів. Кошти було переведено на рахунок з 28-денним періодом зберігання відповідно до умов смарт-контракту Ethereum, тому насправді вони не зникли.

Члени DAO та спільноти Ethereum обговорювали, що робити далі: одні називали атаку допустимим, але неетичним маневром, інші закликали до повторного присвоєння ефіру, а деякі закликали закрити DAO. Зрештою, 20 липня 2016 року мережу Ethereum була піддана хард-форку для переміщення коштів у The DAO на адресу відновлення, де вони могли бути обмінені назад на Ethereum їхніми початковими власниками. Проте деякі продовжували використовувати оригінальний нерозгалужений блокчейн Ethereum, який тепер називається Ethereum Classic. [9]

1.6.2. MakerDAO

Проект MakerDAO було запущено у 2015 році. Над першими фрагментами коду, архітектурою та документацією працювали розробники з усього світу. У грудні 2017 року було опубліковано перший офіційний нормативний документ MakerDAO з описом вихідної системи стейблкойну Dai (тепер Sai).

Документ описував, як за допомогою цієї системи будь-який користувач може генерувати Dai, використовуючи як забезпечення Ethereum (ETH). Система базується на унікальних смарт-контрактах, які називаються заставними борговими зобов'язаннями (Collateralized Debt Positions, CDP). Як забезпечення приймався тільки ETH, тому створена у системі валюта Dai

називалася Dai з однотипною заставою (Single-Collateral Dai, SCD), або Sai. У документі також було представлено план удосконалення системи та підтримки інших видів забезпечення, крім ETH. У життя ця ідея втілилася у листопаді 2019 року.

Тепер у системі стейблкойну Dai, яка зараз називається протоколом Maker, як забезпечення можна використовувати будь-які активи на базі Ethereum, схвалені власниками токенів MKR. Власники токенів голосуванням також встановлюють параметри ризику для кожного заставного активу. Голосування – це найважливіший процес у схемі децентралізованого управління Maker.

Протокол Maker

Протокол Maker - одна з найбільших цифрових програм блокчейна Ethereum. Це перший децентралізований фінансовий додаток, що набув такого широкого поширення. У його проектуванні брали участь різні групи фахівців, включаючи розробників із фонду Maker, зовнішніх партнерів та інших фізичних та юридичних осіб.

Роботу протоколу Maker організують власники токенів MKR із різних країн світу. Вони управляють протоколом Maker та фінансовими ризиками Dai, забезпечуючи стабільність, прозорість та ефективність роботи. Для цього збудовано науково обґрунтовану систему управління, яка передбачає опитування та адміністративні голосування. Один токен MKR, надісланий на адресу контракту голосування, дорівнює одному голосу.

1.6.3. Стейблкойн Dai

Стейблкойн Dai - це справедлива децентралізована багата криптовалюта з м'якою прив'язкою до долара США. Dai зберігають у криптовалютних гаманцях або інших платформах, а також ця валюта використовується в Ethereum та інших популярних блокчейнах.

Dai відрізняється зручністю створення, доступу та використання. Користувачі генерують Dai, вносячи заставні активи до сховищ Maker в

рамках протоколу Maker. Так Dai потрапляє в обіг, а користувачі мають доступ до ліквідності. Валюту Dai також можна придбати у брокерів і на біржах або отримати як платіжний засіб.

Згенеровану, куплену або отриману валюту Dai можна використовувати як будь-які інші криптовалюти: відправляти, оплачувати товари та послуги і навіть віддавати під відсоток, отримуючи передбачену протоколом Maker ставку за заощадженнями в Dai (Dai Savings Rate, DSR).

Вся валюта Dai, що знаходиться в обігу, має надлишкове забезпечення. Це означає, що вартість забезпечення перевищує обсяг боргу у Dai. Усі операції з Dai публічно доступні у блокчейні Ethereum.

Чим валюта Dai схожа на звичайні гроші

У грошей чотири основні функції:

- засіб накопичення;
- засіб обігу;
- міра вартості;
- засіб відстроченого платежу.

Dai має властивості та сценарії використання, які відповідають цим функціям.

Dai як засіб накопичення

Засіб накопичення - це актив, який слабо схильний до знецінення з часом. Валюта Dai відноситься до розряду стейблкойнів, тому її можна використовувати для накопичення навіть в умовах ринкової нестабільності.

Dai як засіб звернення

Засіб обігу — це актив, який є стандартом вартості та відіграє роль посередника при продажу, купівлі або обміні товарів та послуг. Стейблкойн Dai використовується у всьому світі для проведення будь-яких видів комерційних операцій.

Dai як міра вартості

Міра вартості - це еталонна одиниця, в якій виражена вартість товарів та послуг (наприклад, долар США, євро, єна). Наразі цільова вартість Dai

дорівнює 1 долару США (1 Dai = 1 USD). Поза блокчейном Dai не використовується як стандартна міра вартості, проте виконує цю функцію в рамках протоколу Maker і в деяких децентралізованих блокчейн-додатках. Таким чином, розрахунки в протоколі Maker або встановлення цін у децентралізованих додатках ведуться у Dai, а не у державній валюті на кшталт долара США.

Dai як засіб відстроченого платежу

Dai використовується для погашення заборгованості в рамках протоколу Maker (наприклад, у Dai користувачі оплачують комісію за стабільність та закривають сховища). Це перевага Dai перед іншими стейблкойнами.

Заставні активи

Створення, забезпечення та підтримання стабільності Dai відбувається за рахунок заставних активів, що вносяться до сховищ Maker в рамках протоколу Maker. Забезпеченням є цифровий актив, який власники токенів MKR домовилися приймати при роботі з протоколом Maker.

При створенні Dai забезпеченням у протоколі Maker може бути будь-який актив на базі Ethereum, затверджений власниками токенів MKR. Крім того, власники токенів повинні узгодити конкретні параметри ризику для кожного прийнятного виду застави (наприклад, стабільніші активи мають нижчі показники ризику, а волатильніші — суворіші). Докладніше про параметри ризику викладено далі. Ці та інші рішення ухвалюються в рамках децентралізованого процесу управління Maker.[10]

Майбутнє протоколу Maker: подальше поширення та повна децентралізація

Цільовий ринок

Стабільна криптовалюта - це важливий засіб для багатьох децентралізованих додатків. У цьому сенсі потенційний ринок Dai як мінімум можна порівняти з розміром усієї децентралізованої індустрії блокчейн. Насправді перспективи Dai набагато ширші, оскільки ця валюта може стати в нагоді у багатьох інших галузях.

Нижче наведено неповний список поточних та найближчих можливих ринків для стейблкойну Dai.

Оборотний капітал, хеджування та забезпечені кредити. Сховища Maker створюють умови для загальнодоступної торгівлі. Валюту Dai, згенеровану за рахунок поміщеного у сховище забезпечення, користувачі можуть спрямувати до оборотного капіталу. Було неодноразово помічено, що власники сховищ використовують Dai для купівлі додаткових одиниць ETH (того ж типу активу, що й їхнє заставне забезпечення), тим самим формуючи позикову, але повністю забезпечену торгову позицію.

Торгові платежі, транскордонні угоди та грошові перекази. Використання Dai значно скорочує операційні витрати у міжнародній торгівлі завдяки пом'якшенню ефекту валютних коливань та відсутності посередників.

Благодійні та недержавні організації, які використовують прозору технологію розподіленого реєстру.

Ігрова індустрія. Для розробників ігор на базі блокчейну Dai – оптимальна валюта. Впроваджуючи Dai, вони вносять у гру не просто нову валюту, а цілу економічну модель. Поєднання Dai з різними ігровими функціями дозволяє створювати нові поведінкові ігрові сценарії, що базуються на децентралізованій фінансовій системі.

Ринки передбачень. При створенні незалежного прогнозу мало хто захоче збільшувати свій ризик та робити ставки у криптовалюті з нестабільною ціною. Довгострокові ставки стають особливо невигідними, якщо користувачеві також доводиться робити припущення про майбутню ціну волатильного активу, що використовується для розміщення ставки. Тому стейблкойни Dai будуть явно затребувані на ринках пророкувань.

Розширення списку активів

Якщо власники токенів MKR погодяться приймати як забезпечення нові види активів, на них поширюватимуться ті ж вимоги до ризику, параметри оцінки ризику та заходи безпеки, що й на Dai (наприклад, коефіцієнти

ліквідації, комісії за стабільність, процентні ставки за заощадженнями, верхні межі боргу тощо. буд.).

1.7. Telegram та TON

Telegram — це безкоштовна хмарна програма для обміну миттєвими повідомленнями, доступна для багатьох мобільних і настільних платформ, включаючи Android, iOS, Windows, macOS і Linux.

Популярність цього додатка різко зросла у 2021 році після того, як WhatsApp оголосив про зміни у своїй політиці конфіденційності, які дозволять йому обмінюватися даними з материнською компанією Meta.

Telegram пропонує своїм користувачам низку функцій, включаючи відсутність обмежень на розмір медіа, наскрізне шифрування в «секретних чатах» і величезну місткість 200 000 осіб для групових чатів.

Існує також Bot API, який заохочує розробників створювати власних ботів для Telegram.

Окрім Signal, програма має репутацію однієї з найбільш конфіденційних програм для обміну повідомленнями. Код Telegram також є відкритим кодом, і додаток підтримує відтворювані збірки.

Чи зашифрований Telegram?

Усі чати Telegram зашифровані, але рівень шифрування залежить від типу чату, який ви ведете. Це може спонукати користувачів вважати, що їхні чати захищеніші, ніж вони є насправді.

Приватні та групові чати захищені шифруванням сервер-клієнт, що дозволяє їм жити в хмарі, тоді як секретні чати користуються перевагами надійнішого шифрування клієнт-клієнт або наскрізного шифрування. Це означає, що лише відправник і одержувач можуть прочитати ваші повідомлення, і навіть Telegram не може їх розшифрувати. Якщо вам потрібен найкращий рівень конфіденційності, вам слід спілкуватися через секретні чати.

На жаль, це також означає, що наскрізне шифрування обмежено чатами один на один і недоступне в групових чатах.

The Open Network

Гнучка мультиблокчейн-платформа, здатна обробляти мільйони транзакцій за секунду, зі смарт-контрактами повними за Тюрінгом, формальними специфікаціями блокчейну, які можна оновлювати, мультикриптовалютним переказом вартості, підтримкою каналів мікроплатежів і платіжних мереж у ланцюжку. TON Blockchain представляє деякі нові та унікальні функції, такі як самовідновлюваний вертикальний блокчейн механізм і Instant Hypercube Routing, які дозволяють йому бути швидким, надійним, масштабованим і самоузгодженим одночасно.[11]

Однорангова мережа, яка використовується для доступу до блокчейну TON, надсилання транзакцій-кандидатів і отримання оновлень лише про ті частини блокчейну, які цікавлять клієнта (наприклад, пов'язані з клієнтом облікові записи та смарт-контракти), але також здатні підтримувати довільні розподілені служби, пов'язані чи ні з блокчейном.

Технологія розподіленого зберігання файлів, доступна через TON Network, яка використовується блокчейном TON для зберігання архівні копії блоків і даних про стан (миттєві знімки), але також доступні для зберігання довільних файлів для користувачів або інших служб, що працюють на платформі, з технологією доступу, схожою на торрент.

Рівень мережевого проксі/анонімайзера, подібний до I2P (Invisible Internet Project), який використовується для приховування ідентифікаційних даних та IP-адрес вузлів мережі TON, якщо це необхідно (наприклад, вузли, які здійснюють транзакції з облікових записів із великою сумою криптовалюти або блокчейну з високими ставками вузли валідатора, які бажають приховати свою точну IP-адресу та географічне розташування як засіб проти DDoS-атаки).

TON Services це платформа для довільних служб, які знаходяться в TON Network і TON Proxy і доступні через них, з формалізовані інтерфейси, що

дають змогу подібно до браузера чи смартфону взаємодія програми. Ці формальні інтерфейси та постійне обслуговування точки входу можуть бути опубліковані в TON Blockchain; можна знайти фактичні вузли, що надають послуги в будь-який момент через TON DHT, починаючи з інформації, опублікованої в Блокчейні TON. Сервіси можуть створювати смарт-контракти в TON Blockchain запропонувати певні гарантії своїм клієнтам.

TON DNS, служба для призначення зрозумілих імен до облікових записів, смарт-контрактів, сервісів і вузлів мережі.

TON Payments, платформа для мікроплатежів, каналів мікроплатежів і мережі каналів мікроплатежів. Його можна використовувати для швидкого переказу цінностей через о-ланцюг і для оплати послуг від TON Services.

TON забезпечить легку інтеграцію зі сторонніми додатками для обміну повідомленнями та соціальними мережами, завдяки чому технології блокчейну та розподілені сервіси остаточно доступні і доступні звичайним користувачам, а не лише до кількох ранніх криптовалют усиновлювачів.

ВИСНОВКИ ДО РОЗДІЛУ 1

Використання принципу ДАО дозволяє обходитися без традиційного контролю з боку керуючих інститутів. Автоматизація виконання правил, спільні цілі та конкретні стимули управляють сисктмою анітрохи не гірше, а багато в чому – навіть краще.

Однак без ретельного опрацювання правил взаємодії, алгоритмів консенсусу і смарт – контрактів, нічого реально ефективного не вийде.

Крім того, ДАО дуже часто зіштовхуються з проблемами, що носять не технічний, а виключно соціальний характер. А в цілому – це відмінна альтернатива традиційним методам управління.

Протокол Maker дозволяє генерувати стейблкойни Dai, які є засобом накопичення і існують виключно на базі блокчейна. Dai – це децентралізована стабільна валюта. Її емісія та звернення не залежать від жодних центральних регуляторів, довірених посередників чи контрагентів. Це справедлива валюта

з необмеженим поводженням, доступна будь-якій людині в будь-якій точці світу.

У всіх монет Dai є надлишкове забезпечення, внесене кожним учасником на адресу публічних та смарт-контрактів Ethereum, що перевіряються. Слідкувати за станом системи може будь-який учасник з доступом до Інтернету. Для цього використовується веб-сайт daistats.com.

MakerDAO — лідер руху за децентралізовану фінансову систему завдяки сотням своїх партнерств та одній із найсильніших команд розробників у криптовалютній індустрії. Maker використовує потенціал блокчейну, щоб уже зараз надати своїм учасникам обіцяну економічну свободу.

Хоча блокчейн TON є ядром проекту TON, і інші компоненти можна вважати такими, що відіграють допоміжну роль для blockchain, вони мають корисну та цікаву функціональність себе. У поєднанні вони дозволяють платформі розміщувати більш універсальні додатки, ніж це було б можливо за допомогою простого використання TON Blockchain.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ТЕЛЕГРАМ-БОТА

2.1. Аналіз середовищ розробки

Python — найпопулярніша та найулюбленіша мова програмування у світі для програмістів усіх вікових груп. Новачкам, цю мову настійно рекомендують вивчити спочатку. Що ж, у програмуванні ми всі знаємо важливість редактора коду та IDE для написання нашої програми та її виконання, але вибір найкращого редактора коду чи IDE завжди викликає заплутаність. Вибір редактора коду або IDE залежить від багатьох речей, таких як мова програмування, тип проекту, розмір проекту, підтримка ОС і врахування багатьох інших особливостей. Якщо говорити про Python, то ця мова також не є винятком.

Редактор коду — це інструмент, який використовується для написання та редагування коду. Зазвичай вони легкі і чудово підходять для навчання. Однак, як тільки ваша програма стає більшою, вам потрібно протестувати та налагодити свій код, ось де на допомогу приходять IDE.

IDE (або інтегроване середовище розробки) — це програма, призначена для розробки програмного забезпечення. Як випливає з назви, IDE інтегрують кілька інструментів, спеціально розроблених для розробки програмного забезпечення. Ці інструменти зазвичай включають:

- редактор, призначений для роботи з кодом (наприклад, із підсвічуванням синтаксису та автозавершенням);
- інструменти збірки, виконання та налагодження;
- певна форма контролю джерела.

Більшість IDE підтримують багато різних мов програмування та містять набагато більше функцій. Тому вони можуть бути великими та потребувати

часу для завантаження та встановлення. Вам також можуть знадобитися передові знання, щоб правильно їх використовувати.

Навпаки, спеціальний редактор коду може бути таким же простим, як текстовий редактор із підсвічуванням синтаксису та можливостями форматування коду. Більшість хороших редакторів коду можуть виконувати код і керувати налагоджувачем. Найкращі також взаємодіють із системами керування джерелами. Порівняно з IDE хороший спеціальний редактор коду зазвичай менший і швидший, але часто менш багатий функціями.

1. Онлайн компілятор від Programiz

За допомогою Programiz можна писати та редагувати код онлайн. Щоб почати, вам потрібен лише Інтернет і браузер (рис. 2.1).

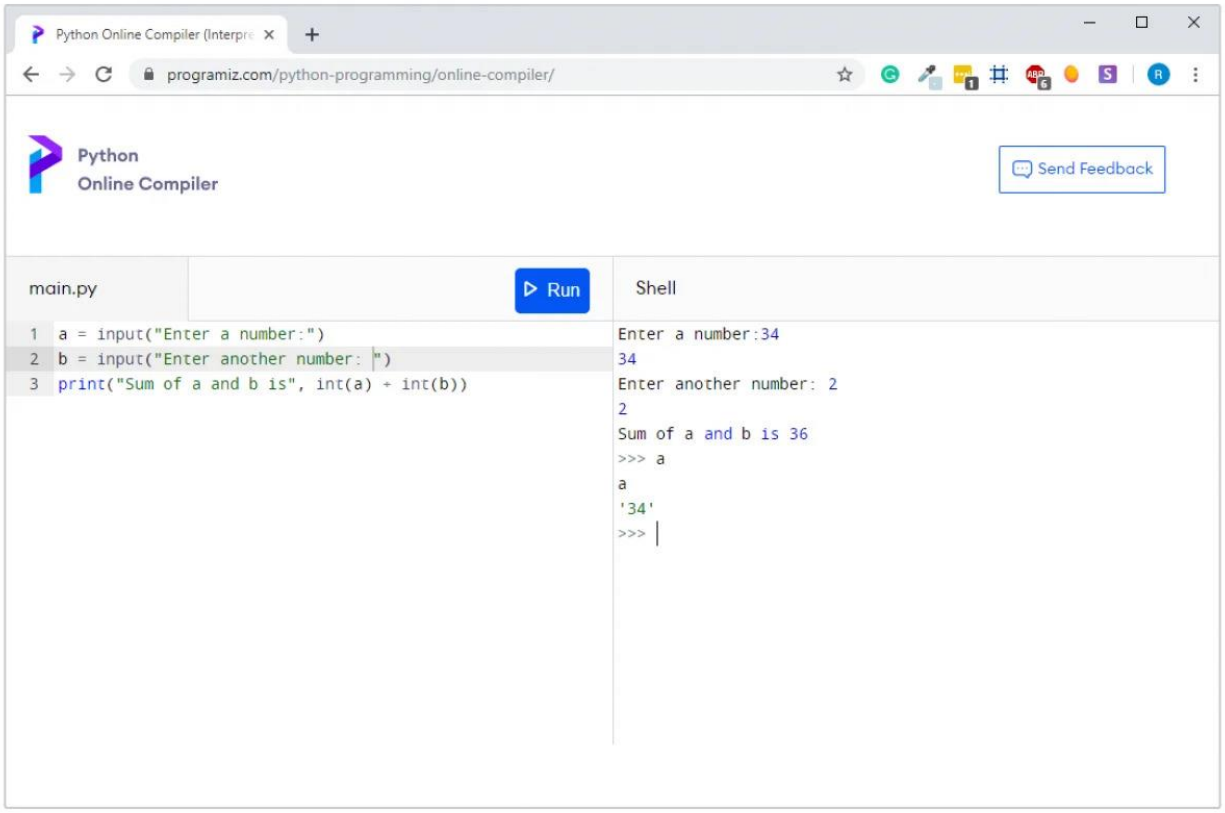

Рис. 2.1. Онлайн-компілятор Python

2. IDLE

Коли ви встановлюєте Python, IDLE(рис. 2.2) також встановлюється за замовчуванням. Це полегшує початок роботи з Python. Його основні функції

включають вікно оболонки Python (інтерактивний інтерпретатор), автозавершення, підсвічування синтаксису, розумні відступи та базовий інтегрований налагоджувач.

IDLE гарно підходить для навчання, оскільки вона легка та проста у використанні. Однак це не є оптимальним для великих проектів.

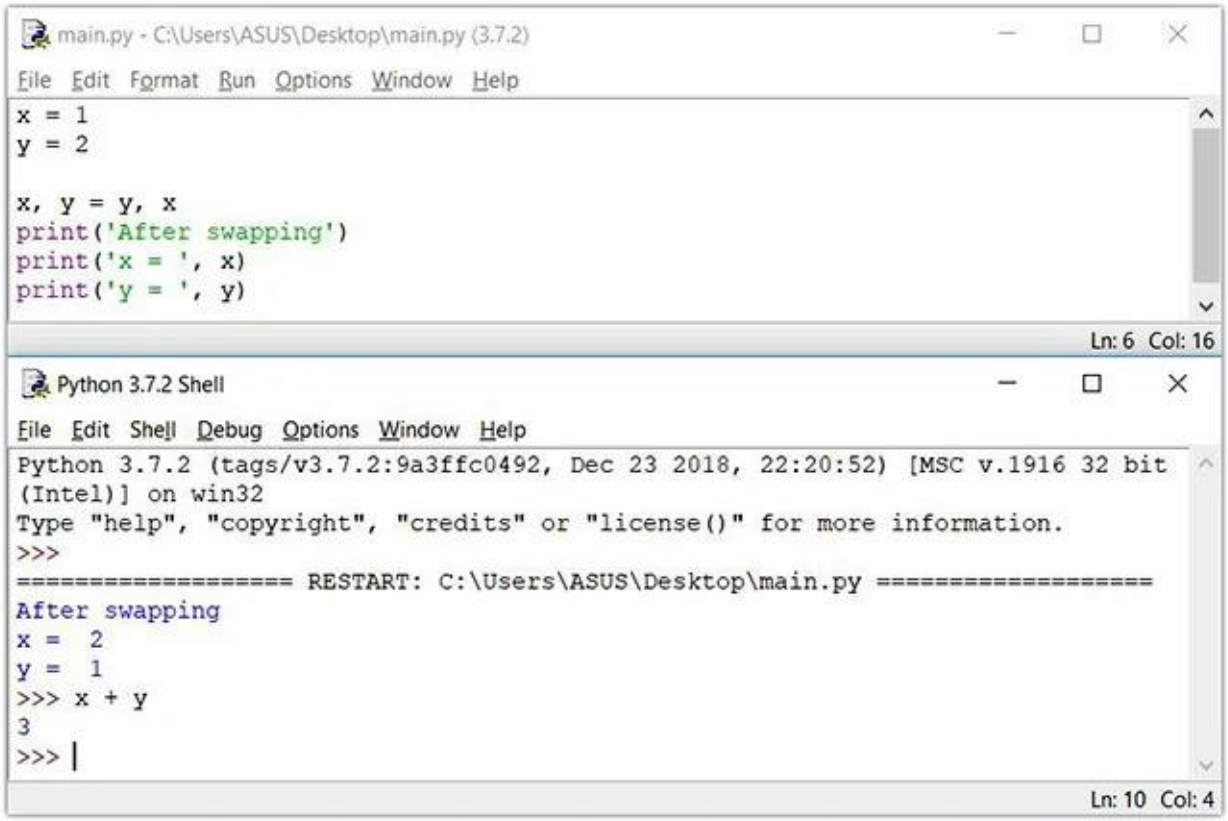

Рис.2.2. Інтегроване середовище розробки та навчання Python

3. Sublime Text 3

Sublime Text — популярний редактор коду, який підтримує багато мов, включаючи Python. Він швидкий, з широкими можливостями налаштування та має величезну спільноту.

Деякі особливості програми:

Інтерфейс

Редактор містить різні візуальні теми, з можливістю завантаження додаткових. Користувачі бачать весь свій код у правій частині екрана у вигляді міні-карти, при натисканні на яку можна активувати навігацію. Є кілька

режимів екрану. Один з них включає від 1 до 4 панелей, за допомогою яких можна показати до чотирьох файлів одночасно. Повноцінний (вільні режими) режим показує тільки один файл без будь-яких додаткових меню навколо нього.

Виділення стовпчиківі множинна правка

Виділення стовпчиків повністю або розстановка кількох вказівників по тексту, що робить можливу миттєву правку. Вказівники поводяться, начебто кожен із них — один в тексті. Команди типу переміщення на знак, переміщення на рядок, вибір тексту, переміщення слова або його частини (CamelCase, роздільний дефіс або підкреслення), переміщення в початок/кінець рядка тощо, впливають на всі вказівники незалежно і відразу, дозволяючи відправити складноструктуровий текст швидко, без використання макрокоманд або регулярних виражень.

Автодоповнення

Коли користувач набирає код, Sublime Text, залежно від використовуваної мови, запропонує різні варіанти для завершення запису. Редактор також автоматично завершує створені користувачем змінні.

Підсвітка синтаксису і висока контрастність

Темний фон Sublime Text призначений для підвищення контрастності тексту. Основні елементи синтаксису виділені різними кольорами, які краще поєднуються з темним фоном, небажаними зі світлим.

Підтримка системної збірки

Sublime Text дозволяє користувачеві збирати програми та запускати їх без необхідності переключатися в командну строку. Користувач також може будувати свою системну збірку та вмикати автоматичну збірку програми кожного разу при збереженні коду.

Заготовки (сніппети)

Збереження фрагментів часто використовуваного коду, ключові слова для їх запуску.

Перехід по файлу

Навігаційний інструмент, який дозволяє користувачам переміщатися між файлами, а також всередині них, за допомогою незначного пошуку.

Інші особливості

Додатково реалізована функція автозахисту, яка допомагає користувачам не втратити пророблену роботу.

Налаштовані комбінації клавіш і інструментів навігації дозволяють призначати свої комбінації клавіш для меню та панелей інструментів (тільки для першої версії, у другій і третій — Палітра команд).

Можливість пошуку по набору вимірювань використовується для пошуку в документі. Функція перевірки синтаксису працює подібним чином, перевіряючи коректність прямо під час введення. Є можливість автоматизації за допомогою макросів і повторення останніх дій. Команди редагування, включаючи редагування відступів, переформатування абзаців і об'єднання рядків. Є можливість завантажити evaluate Sublime text(рис. 2.3) протягом невизначеного періоду часу. Однак час від часу ви отримуватимете спливаюче вікно з повідомленням «вам потрібно придбати ліцензію для подальшого використання».

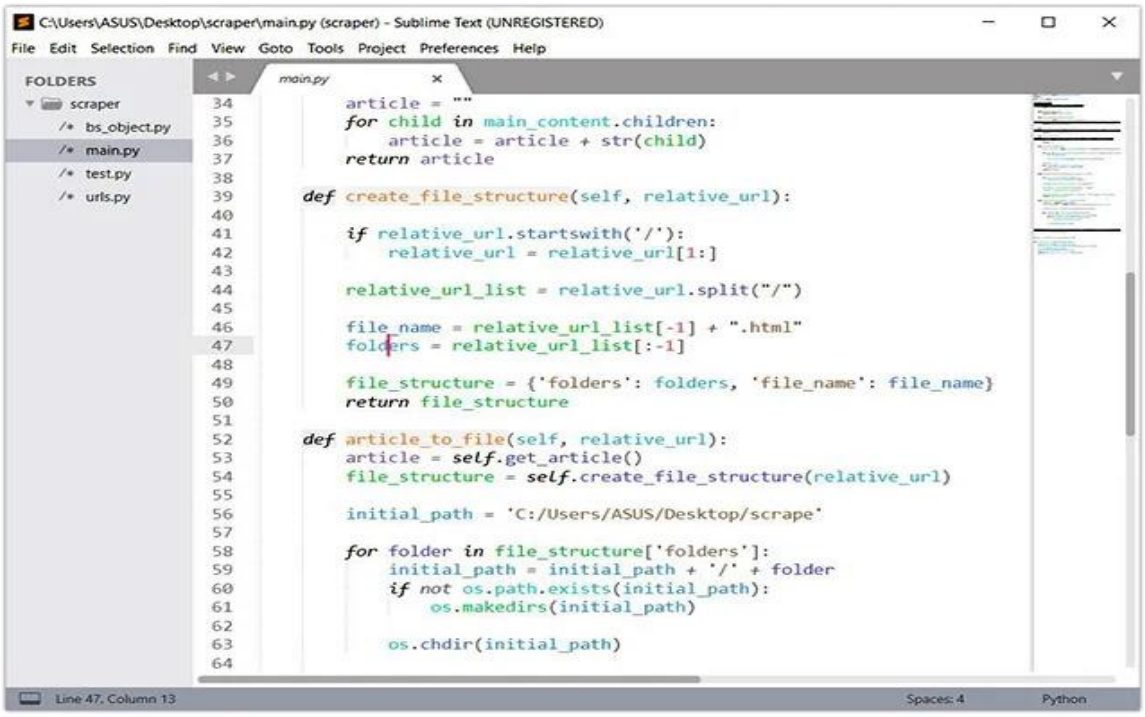

Рис. 2.3. Редактор Sublime Text

4. Atom

Atom — це редактор коду з відкритим вихідним кодом, розроблений Github, який можна використовувати для розробки на Python (подібний Sublime text).

Його функції також схожі на Sublime Text. Atom можна налаштувати (рис. 2.4). Ви можете встановити пакети відповідно до ваших потреб. Деякі з пакетів, які часто використовуються в Atom для розробки на Python, це autocomplete-python, linter-flake8, python-debugger тощо.

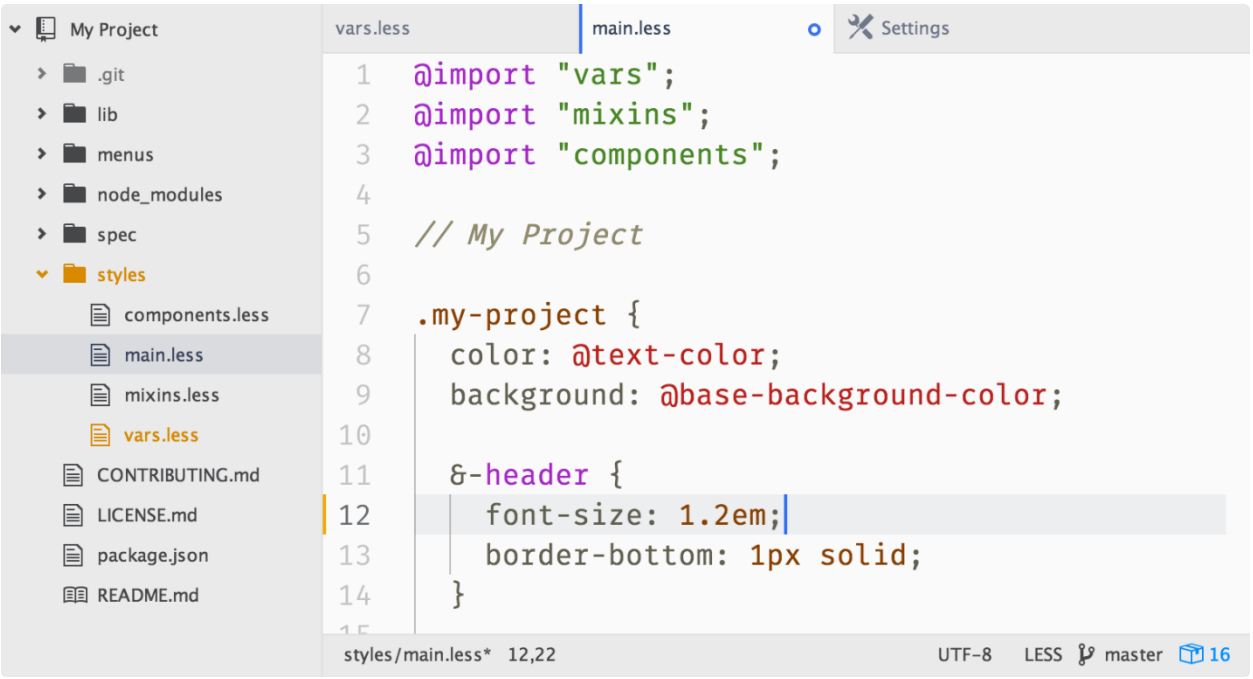

Рис. 2.4. Atom для розробки Python

5. Thonny

Thonny — це спеціальне середовище розробки Python, яке постачається з вбудованим Python 3. Після встановлення ви можете почати писати код Python.

Thonny призначений для новачків (рис. 2.5). Інтерфейс користувача зберігається простим, тому новачкам буде легко розпочати роботу. Хоча Thonny призначений для початківців, він має кілька корисних функцій, які також роблять його хорошим IDE для повноцінної розробки Python. Деякі з

його функцій – підсвічування синтаксичних помилок, налагоджувач, завершення коду, покрокова оцінка виразу тощо.

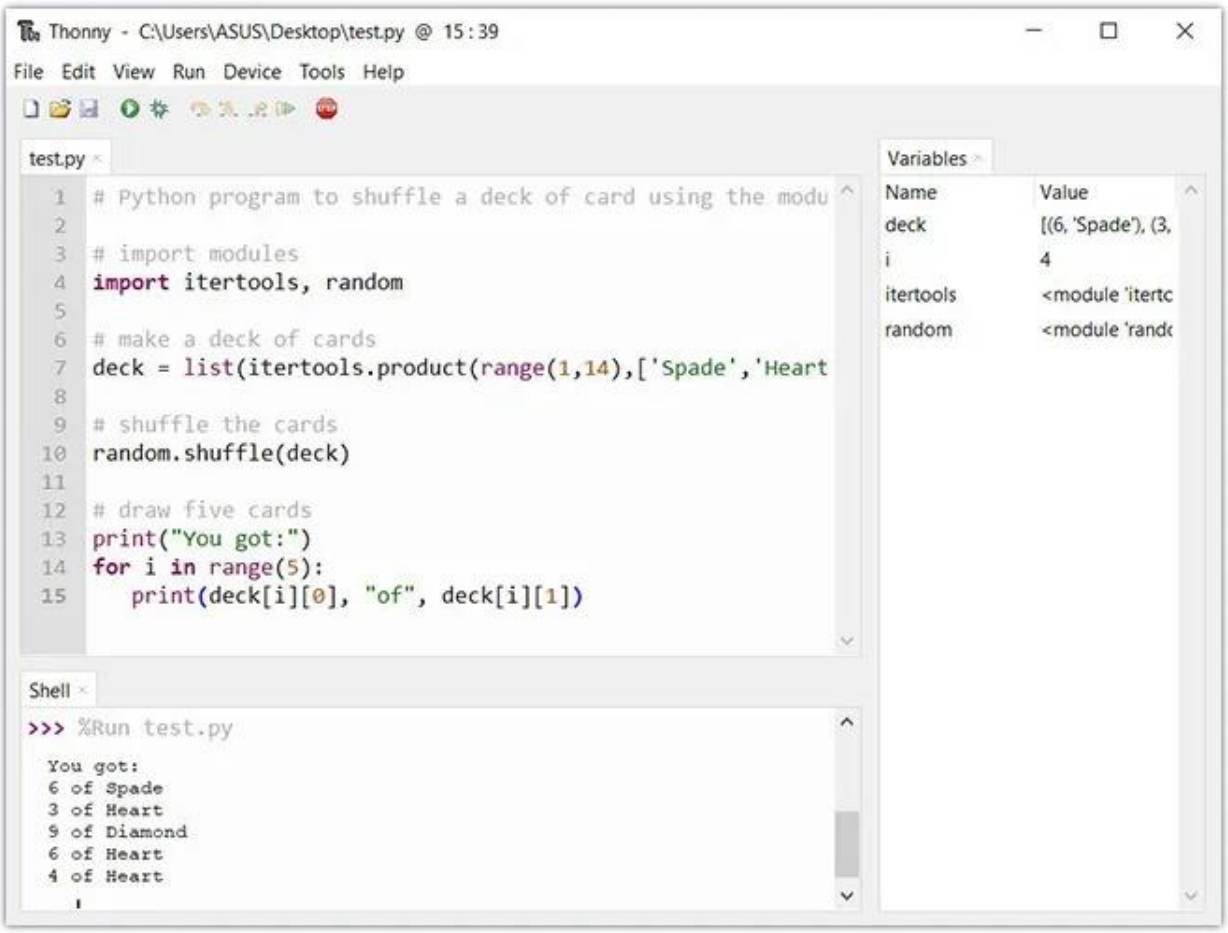

Рис. 2.5. Thonny IDE

6. PyCharm

PyCharm — це IDE для професійних розробників. Він створений JetBrains, компанією, відомою створенням чудових інструментів розробки програмного забезпечення (рис. 2.6).

PyCharm доступний у трьох версіях:

- спільнота (безкоштовна та з відкритим вихідним кодом): для розумної та інтелектуальної розробки Python, включаючи допомогу в коді, рефакторинг, візуальне налагодження та інтеграцію контролю версій;

- професійний (оплачується) : для професійної розробки Python, Інтернету та науки про дані, включаючи допомогу в коді, рефакторинг,

візуальне налагодження, інтеграцію контролю версій, віддалене налаштування, розгортання, підтримку популярних веб-фреймворків, таких як Django та Flask, підтримку баз даних, наукову інструменти (включаючи підтримку блокнотів Jupyter), інструменти для великих даних;

- edu (безкоштовний і з відкритим кодом): для вивчення мов програмування та пов'язаних технологій за допомогою інтегрованих навчальних інструментів.

PyCharm підтримує такі версії Python:

- Python 2: версія 2.7;
- Python 3: від версії 3.6 до версії 3.11.

Крім того, у версії Professional можна розробляти програми Django, Flask і Pyramid. Крім того, він повністю підтримує HTML (включно з HTML5), CSS, JavaScript і XML: ці мови включено в IDE через плагіни та ввімкнено за замовчуванням. Підтримку інших мов і фреймворків також можна додати за допомогою плагінів.

PyCharm надає всі основні функції, які повинна надавати хороша IDE: доповнення коду, перевірки коду, підсвічування та виправлення помилок, налагодження, система контролю версій і рефакторинг коду. Усі ці функції виходять «із коробки».

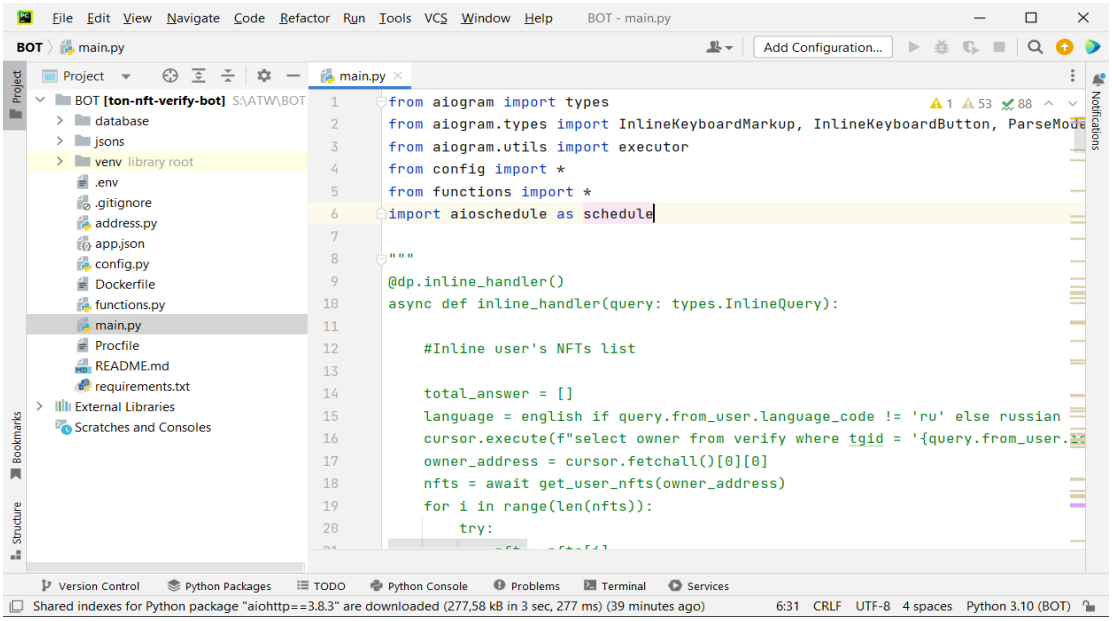

Рис. 2.6. Інтегроване середовище розробки PyCharm

7. Код Visual Studio

Visual Studio Code (VS Code) — це безкоштовна IDE із відкритим вихідним кодом, створена Microsoft, яку можна використовувати для розробки на Python (рис. 2.7).

Ви можете додати розширення для створення середовища розробки Python відповідно до ваших потреб у коді VS. Він надає такі функції, як інтелектуальне завершення коду, пошук потенційних помилок, налагодження, модульне тестування тощо. VS Code легкий і наповнений необхідними функціями. Це причина, чому він стає популярним серед розробників Python.

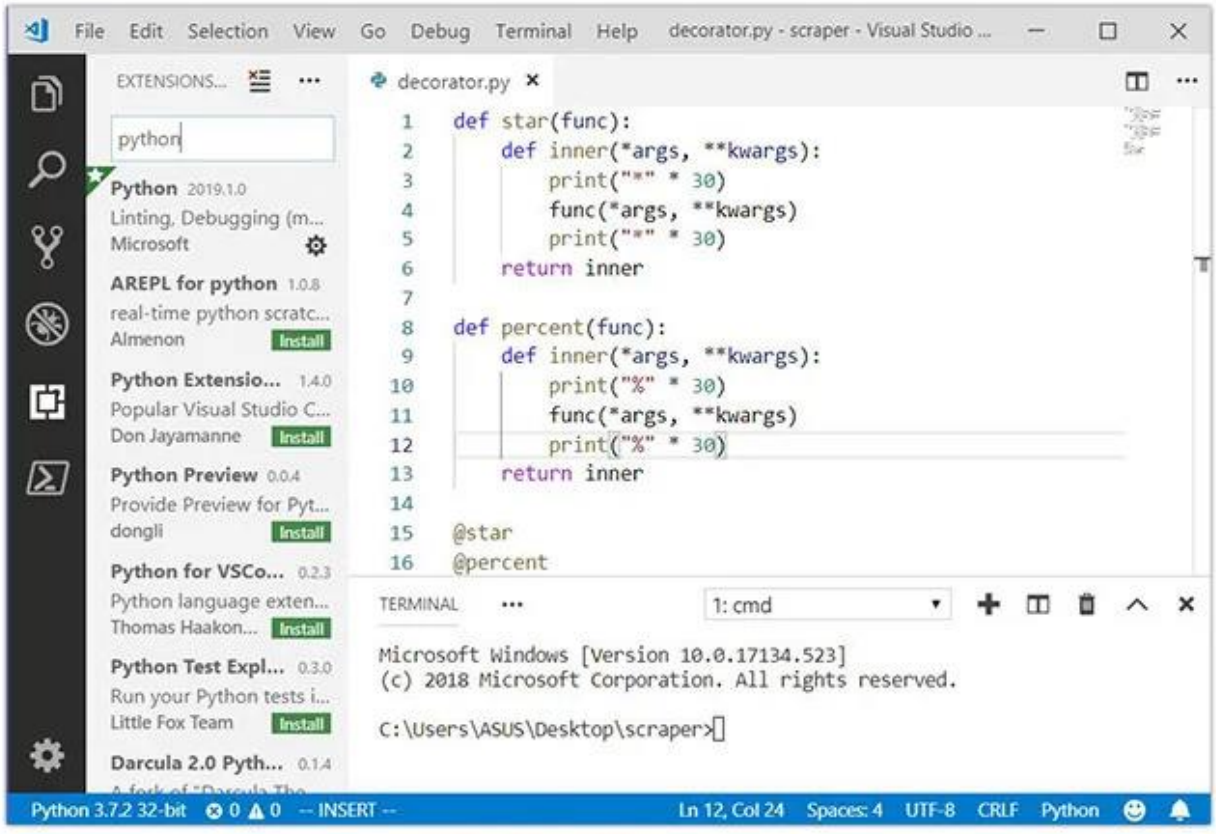

Рис. 2.7. Інтегроване середовище розробки VS Code

8. Spyder

Spyder — це IDE з відкритим кодом, який зазвичай використовується для наукових розробок (рис. 2.8).

Найпростіший спосіб налагодити роботу зі Spyder — це встановити дистрибутив Anaconda. Якщо ви не знаєте, Anaconda — це популярний

дистрибутив для науки про дані та машинного навчання. Дистрибутив Anaconda включає сотні пакетів, включаючи NumPy, Pandas, scikit-learn, matplotlib тощо.

Spyder має кілька чудових функцій, таких як автозавершення, налагодження та оболонка iPython. Однак йому бракує функцій порівняно з PyCharm.

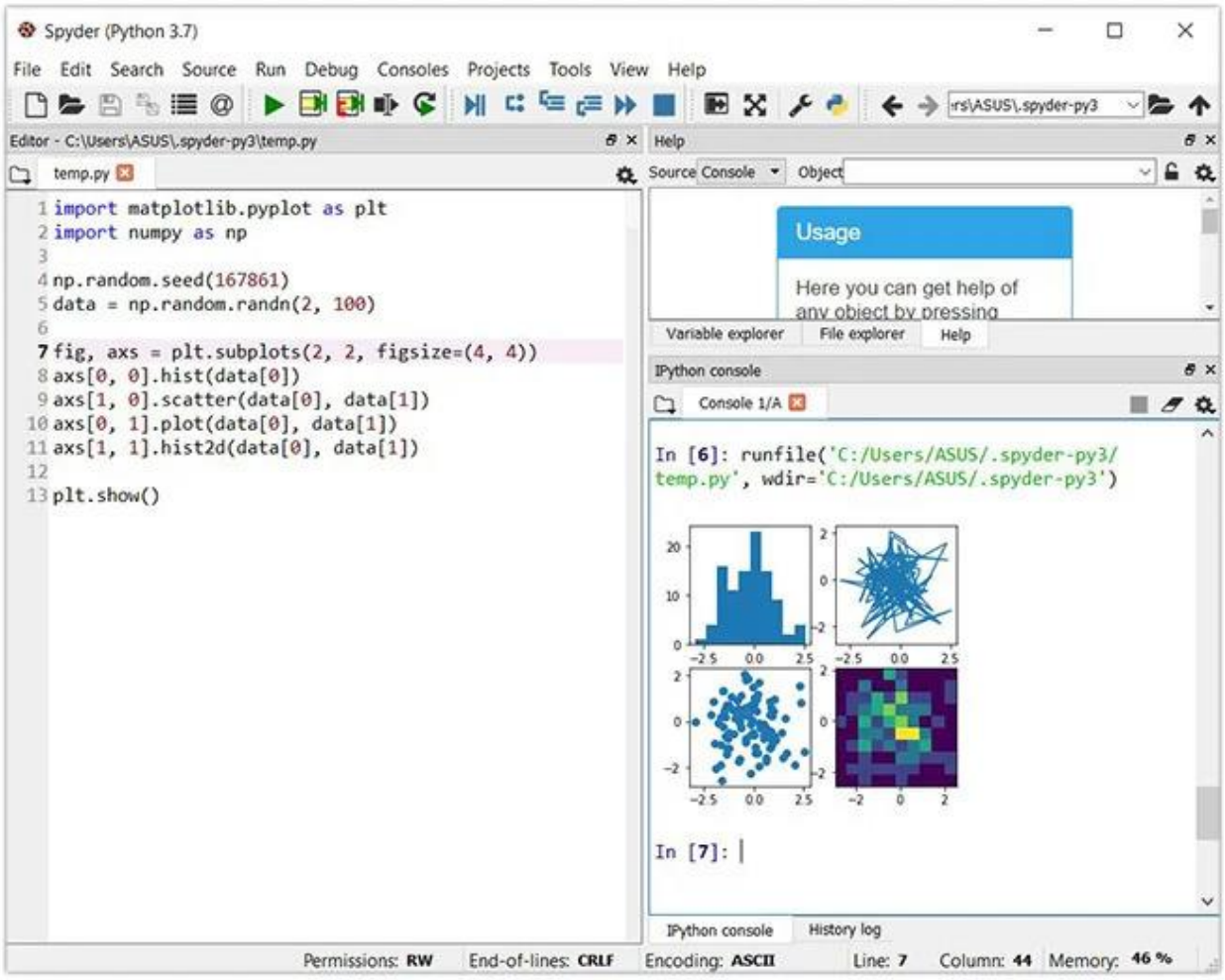

Рис. 2.8. Інтегроване середовище розробки Spyder IDE

Щоб полегшити вибір середовища, в дипломній роботі, було створено таблицю. Вона відображає різницю між середовищами, поділяється на 4 стовпчики, серед яких окрім назви є помітки, існує у безкоштовному варіанті, чи тільки платно, а також для якого рівня підготовки програмістів підходить

Таблиця 2.1

Порівняльна таблиця середовищ розробки

	Підходить для	безкоштовно	платно
Programiz	Новачків	+	-
IDLE	Новачків	+	-
Sublime Text	Новачків та професіоналів	+	+
Atom	Новачків та професіоналів	+	-
Thonny	Новачків	+	-
PyCharm	Професіоналів	+	+
Visual Studio	Професіоналів	+	-
Spyder	Новачків та професіоналів	+	-

2.2. PyCharm

Інтелектуальна допомога в кодуванні

PyCharm забезпечує інтелектуальне завершення коду, перевірку коду, підсвічування помилок на льоту та швидкі виправлення, а також автоматизовану рефакторинг коду та широкі можливості навігації.[12]

2.2.1. Інтелектуальний редактор коду

Підсвічування синтаксису

Читати ваш код легше завдяки настроюваним кольорам для коду Python і шаблонів Django. Виберіть із кількох попередньо визначених колірних тем.

Автоматичний відступ і форматування коду

Автоматичні відступи вставляються на новому рядку. Перевірка відступів і переформатування коду відповідають параметрам стилю коду проекту.

Настроювані стилі коду

Виберіть попередньо визначений стиль кодування, щоб застосувати його до конфігурації стилю коду для різних підтримуваних мов.

Доповнення коду

Доповнення коду для ключових слів, класів, змінних тощо під час введення або за допомогою Ctrl+Пробіл (рис. 2.9). Пропозиції редактора залежать від контексту та пропонують найбільш прийнятні варіанти.

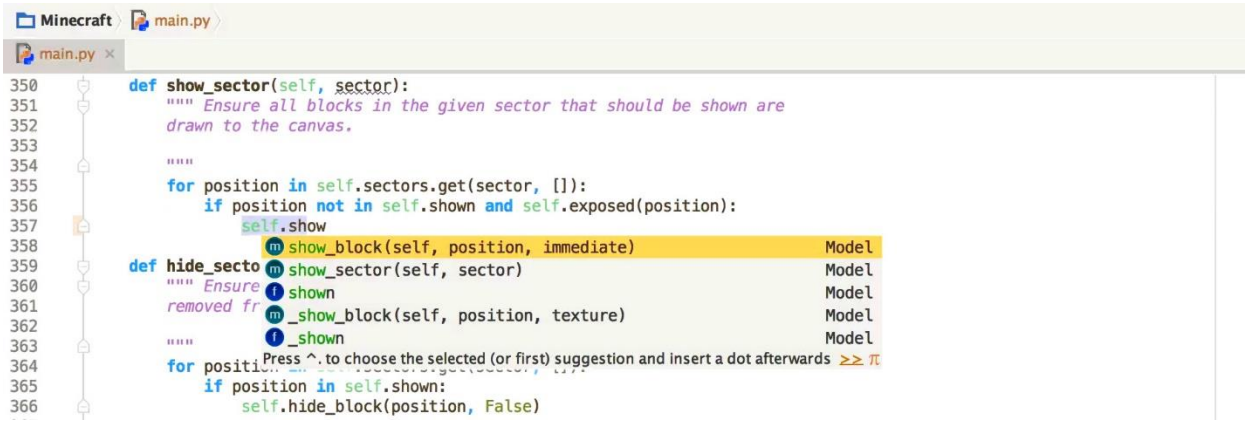

Рис. 2.9. Приклад інтелектуального редактора коду

Вибір коду та коментарі

Виберіть блок коду та розгорніть його до виразу, до рядка, до логічного блоку коду тощо за допомогою ярликів. Одне натискання клавіші для коментування/розкоментування поточного рядка чи виділення.

Форматувальник коду

Форматувальник коду з конфігурацією стилю коду та іншими функціями допоможе вам написати акуратний код, який легко підтримувати. PyCharm містить вбудований PEP-8 для форматування коду Python та інших стандартів для підтримуваних мов.

Фрагменти коду

Економте час, використовуючи розширені настроювані та параметризовані живі шаблони коду та фрагменти.

Складання коду

Згортання коду, автоматичне вставлення дужок, дужок і лапок, підсвічування дужок/дужок тощо.

Підсвічування помилок на льоту

Помилки відображаються під час введення. Вбудований засіб перевірки орфографії перевіряє ваші ідентифікатори та коментарі на орфографічні помилки.

Кілька вставок і виділень

З кількома вставками ви можете редагувати кілька місць у файлі одночасно.

Аналіз коду

Численні перевірки коду перевіряють код Python під час введення, а також дозволяють перевіряти весь проект на наявність можливих помилок або запахів коду.

Швидкі виправлення

Швидкі виправлення для більшості перевірок дозволяють легко виправити або миттєво покращити код. Alt+Enter показує відповідні параметри для кожної перевірки.

Детектор повторюваного коду

Розумний детектор повторюваного коду аналізує ваш код і шукає скопійований/вставлений код. Вам буде запропоновано список кандидатів на рефакторинг — і за допомогою рефакторингу ваш код буде легко залишатися сухим.

Настроювані мовні ін’єкції

Нативно редагуйте код, відмінний від Python, вбудований у рядкові літерали, із доповненням коду, підсвічуванням помилок та іншими функціями допомоги при кодуванні.

Автоматична генерація коду

Автоматична генерація коду з використанням із швидкими виправленнями; рядки документів і перевірка відповідності коду, а також автоматичне оновлення під час рефакторингу. Автоматичне створення заглушки рядка документа (reStructuredText, Epytext, Google і NumPy).

Намір дії

Дії намірів допомагають застосувати автоматичні зміни до правильного коду, щоб покращити його або спростити програмування.

Швидкий і безпечний рефакторинг

Вносьте глобальні зміни в проект легко та безпечно. Локальні зміни вносяться миттєво на місці (рис. 2.10). Рефакторинг працює у звичайному Python та інших типах проектів, таких як Django, Flask, Pyramid тощо.

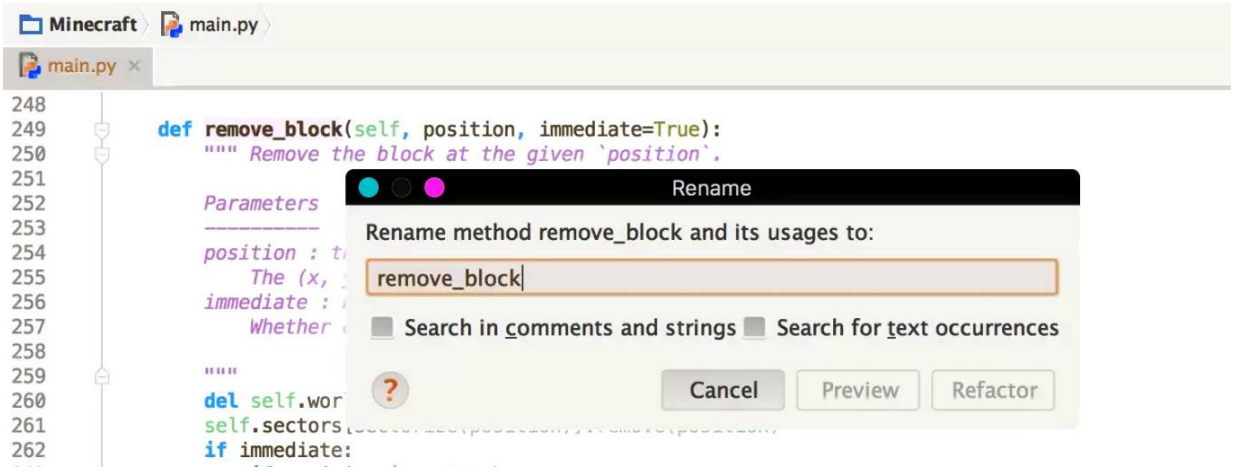

Рис. 2.10. Швидкий і безпечний рефакторинг

Перейменувати та перемістити

Рефакторинг Rename і Move працює для файлів, функцій, констант, класів, властивостей, методів, параметрів, а також локальних і глобальних змінних.

Витягти рефакторинг

Використовуйте Extract Variable/Field/ Constant/Parameter і Inline Local для покращення структури коду в методі.

Метод вилучення

Використовуйте Extract Method, щоб розбити довші методи, Extract Superclass, Push Up, Pull Down, щоб перемістити методи та класи.

Інші рефакторинги

Також доступні наступні рефакторинги: зміна підпису, перетворення на пакет Python/перетворення на модуль Python, створення функції верхнього рівня тощо.

Документація

Швидке визначення / перегляд документації

Переглядайте визначення об’єкта або документацію на місці, не втрачаючи контексту (рис. 2.11).

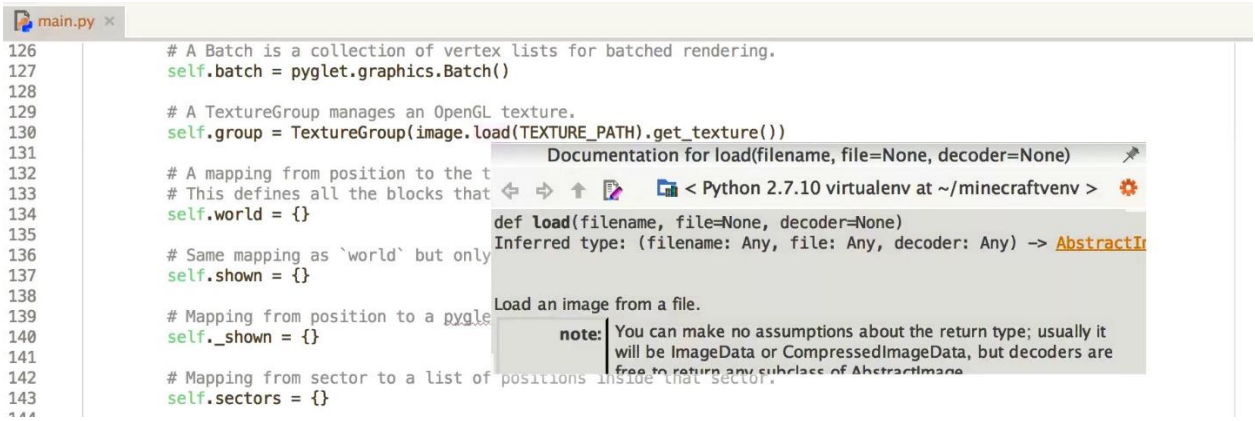

Рис. 2.11. Приклад документація

Зовнішній вигляд документації

Перегляд зовнішньої документації доступний у браузері для стандартної бібліотеки Python, Django, Google App Engine, пакетів Scientific тощо (використовуйте Shift+F1).

Генерація заглушки рядка документації

Автоматично генерувати заглушку рядка документації (Epydoc, reStructuredText, Google, NumPy) для методу.

Виділення і завершення

PyCharm забезпечує підсвічування синтаксису та доповнення коду для тегів і параметрів тегів у Docstrings, а також перевірку відповідності коду та автоматичне оновлення під час рефакторингу.

2.2.2. AIOGRAM

Aiogram — це досить проста та повністю асинхронна структура для API Telegram Bot, написана на Python 3.7 з asyncio та aiohttp. Це допоможе вам зробити ваших ботів швидшими та простішими.[13]

Особливості:

- асинхронний;
- робить роботу швидше;
- має FSM;
- можна відповісти у вебхук. (Іншими словами, робити запити у відповідь на оновлення).

Технічна документація бібліотеки містить розділи:

- об'єкт бота:
  - API низького рівня;
  - Telegram Bot;
  - API Helpers;
- типи даних Telegram:
  - бази:
    - базовий TelegramObject:
      - MetaTelegramObject;
      - TelegramObject;
    - поля:
      - BaseField;
      - Field;
      - ListField;
      - ListOfLists;
      - DateTimeField;
      - TextField;
    - міксини:
      - можна завантажити;
  - типи:

- набір наклейок;
- EncryptedCredentials;
- CallbackQuery;
- успушна оплата;
- MessageEntity:
  - MessageEntityType;
- ShippingQuery;
- паспортні дані;
- InlineKeyboardMarkup:
  - InlineKeyboardButton;
- користувач;
- відео;
- EncryptedPassportElement;
- гра;
- файл;
- LabeledPrice;
- CallbackGame;
- ReplyKeyboardMarkup:
  - KeyboardButton;
  - CallbackKeyboardRemove;
- чат:
  - ChatType;
  - ChatActions;
- документ;
- аудіо;
- ForceReply:
  - PassportElementError;
  - PassportElementErrorDataField;
  - PassportElementErrorFile;
  - PassportElementErrorFiles;

    - PassportElementErrorFrontSide;
    - PassportElementErrorReverseSide;
    - PassportElementErrorSelfie;
- адреса доставки;
- параметри відповіді;
- інформація про замовлення;
- GameHighScore;
- наклейка;
- InlineQuery;
- місцезнаходження;
- анімація;
- InputMedia:
    - InputMediaAnimation;
    - InputMediaDocument;
    - InputMediaAudio;
    - InputMediaPhoto;
    - InputMediaVideo;
    - MediaGroup;
- InlineQueryResult:
    - InlineQueryResultArticle;
    - InlineQueryResultPhoto;
    - InlineQueryResultGif;
    - InlineQueryResultMpeg4Gif;
    - InlineQueryResultVideo;
    - InlineQueryResultAudio;
    - InlineQueryResultVoice;
    - InlineQueryResultDocument;
    - InlineQueryResultLocation;
    - InlineQueryResultVenue;

    - InlineQueryResultContact;
    - InlineQueryResultGame;
    - InlineQueryResultCachedPhoto;
    - InlineQueryResultCachedGif;
    - InlineQueryResultCachedMpeg4Gif;
    - InlineQueryResultCachedSticker;
    - InlineQueryResultCachedDocument;
    - InlineQueryResultCachedVideo;
    - InlineQueryResultCachedVoice;
    - InlineQueryResultCachedAudio;
- InputFile;
- PreCheckoutQuery;
- голос;
- InputMessageContent:
    - InputContactMessageContent;
    - InputLocationMessageContent;
    - InputTextMessageContent;
    - InputVenueMessageContent;
- оновлення:
    - AllowedUpdates;
- PhotoSize;
- місце проведення;
- ChosenInlineResult;
- VideoNote;
- WebhookInfo;
- PassportFile;
- ChatMember:
    - статус учасника чату;
- ShippingOption;
- ChatPhoto;

- контакт;
- повідомлення:
  - ContentType;
  - ContentTypes;
  - ParseMode;
- MaskPosition;
- фотографії профілю користувача;
- рахунок-фактура;
- AuthWidgetData;

- диспетчер:
  - фільтри:
    - основи;
    - завод фільтрів;
    - вбудовані фільтри:
      - команда;
      - CommandStart;
      - CommandHelp;
      - параметри команди;
      - CommandPrivacy;
      - текст;
      - HashTag;
      - регулярний вираз;
      - RegexpCommandsFilter;
      - ContentTypeFilter;
      - IsSenderContact;
      - фільтр стану;
      - фільтр винятків;
      - IDFilter;
      - AdminFilter;
      - IsReplyFilter;

- ForwardedMessageFilter;
- ChatTypeFilter;
- MediaGroupFilter;

- створення власних фільтрів (користувацькі фільтри):
  - AbstractFilter;
  - фільтр;
  - BoundFilter;

○ кінцевий автомат:

- зберігання:
  - доступне сховище:
    - зберігання пам'яті;
    - сховище Redis;
    - сховище Mongo;
    - переосмислити зберігання БД;
    - створення власного сховища;
- держави;
- державні комунальні послуги:
  - держава;
  - група держав;

○ проміжне програмне забезпечення:

- бази;
- створення власного проміжного програмного забезпечення;
- доступне проміжне програмне забезпечення;

○ вебхук:

- бази;
- безпека;
- створення запитів при отриманні оновлень;

○ основи;

○ доступні обробники:

- клас обробника;
  - особливості;
  - клас диспетчера;
    - utils:
      - віджет авторизації;
      - виконавець;
      - винятки;
      - уцінка;
      - помічник;
      - застаріле;
      - корисне навантаження;
      - запчастини;
      - JSON;
      - emoji;
      - глибинне посилання;
    - приклади:
      - ехо-бот;
      - вбудований бот;
      - розширений приклад виконавця;
      - проксі та емодзі;
      - приклад кінцевого автомата;
      - приклад дроселювання;
      - I18n приклад;
      - приклад фільтра команд регулярного виразу;
      - перевірте мову користувача;
      - проміжне програмне забезпечення та антифлуд;
      - приклад вебхуку;
      - старий приклад вебхуку;
      - платежі;
      - приклад трансляції;

- медіа група;
- локальний сервер;

- внесок;
- посилання.

У кваліфікафйній роботі будуть використані не всі доступні методи, а тільки необхідні для роботи бота.

Перевагою даної бібліотеки є можливість її встановлення різними методами та способами.

Наприклад:

1. Використання PIP

   У командному рядку написати: *pip install -U aiogram*

2. Використання Pipenv

   У командному рядку написати: *pipenv install aiogram*

3. Використання Pacman

   Aiogram також доступний у репозиторії Arch Linux, тож існує можливість встановити цю структуру на будь-якому дистрибутиві на основі Arch, наприклад Arch Linux, Antergos, Manjaro тощо. Для цього просто потрібно скористатись pacman, щоб установити пакет python-aiogram: *pacman -S python-aiogram*

4. З джерел

   Версії розробки:

   ```
   $ git clone https://github.com/aiogram/aiogram.git
   $ cd aiogram
   $ python setup.py install
   ```

   Або якщо потрібно встановити стабільну версію (те саме з версією з PyPi):

   ```
   $ git clone https://github.com/aiogram/aiogram.git
   $ cd aiogram
   $ git checkout master
   $ python setup.py install
   ```

ВИСНОВКИ ДО РОЗДІЛУ 2

У даному теоретичному розділі кваліфікаційної роботи було наведено ряд середовищ для розробки на Python, а також створена порівняльна таблиця з найважливішими зіставленнями. Перелічені розділи технічної документації дозволяють ознайомитись з основними положеннями та поділами використаної бібліотеки для асинхронного програмування, а також варіанти встановлення модулів цієї бібліотеки.

PyCharm — одна з найкращих, якщо не найкраща, повнофункціональна, спеціальна та універсальна IDE для розробки на Python. Він пропонує масу переваг, заощаджуючи багато часу, допомагаючи виконувати рутинні завдання.

PyCharm надає такі можливості:

- написання коду;
- запуск вашого коду;
- налагодження та тестування вашого коду;
- редагування існуючого проекту;
- використання контролю версій;
- використання плагінів і зовнішніх інструментів;
- використання функцій PyCharm Professional, таких як підтримка Django та науковий режим.

РОЗДІЛ 3
РОЗРОБКА ТЕЛЕГРАМ-БОТА

3.1. Мова програмування

Python — це мова програмування високого рівня загального призначення. Його філософія дизайну наголошує на читабельності коду з використанням значних відступів.

Python динамічно типізується та збирає сміття. Він підтримує кілька парадигм програмування, включаючи структуроване (зокрема процедурне), об’єктно-орієнтоване та функціональне програмування. Його часто описують як мову «батареї включені» через його повну стандартну бібліотеку.

Гвідо ван Россум почав працювати над Python наприкінці 1980-х як наступником мови програмування ABC і вперше випустив її в 1991 році як Python 0.9.0. Python 2.0 був випущений у 2000 році та представив нові функції, такі як розуміння списків, збирання сміття з визначенням циклу, підрахунок посилань і підтримка Unicode. Python 3.0, випущений у 2008 році, був основною версією, яка не повністю сумісна з попередніми версіями. Python 2 було припинено з версією 2.7.18 у 2020 році.[14]

Python незмінно вважається однією з найпопулярніших та найпростіших мов програмування, саме тому для написання телеграм бота був вибраний саме Python.

3.2. Telegram MTProto

Telegram використовує власний протокол шифрування MTProto. MTProto API (він же Telegram API) — це API, через який ваш додаток Telegram спілкується із сервером. API Telegram повністю відкритий, тому будь-який розробник може написати власний клієнт месенджера.

Для написання ботів був створений інтерфейс Telegram Bot API – надбудова для Telegram API. Переклад з офіційного сайту:
«Щоб використовувати Bot API, вам не потрібно нічого знати про те, як працює протокол шифрування MTProto — наш бекенд піклується про шифрування та зв’язок із самим Telegram API. Ви підключаєтесь до сервера через простий інтерфейс HTTPS, який надає просту версію Telegram API.»

Серед спрощень Bot API: робота через веб-хуки, спрощене тегування повідомлень тощо.

Чомусь мало хто знає, що боти можуть працювати безпосередньо через API Telegram. Крім того, таким чином можна навіть обійти деякі обмеження, які надає API бота.

Уся інформація, наведена нижче, за замовчуванням застосовуватиметься як до API ботів, так і до API Telegram. Я згадаю про відмінності. Використовуючи локальний сервер, ви можете позбутися деяких обмежень Bot API.[15]

3.3. Розробка Бота

До початку розробки необхідно виконати декілька простих кроків:

Частина 1: Реєстрація бота

Найпростіша та описана частина. Дуже коротко: потрібно знайти бота @BotFather, ввести в нього /start або /newbot, заповнити необхідні поля (ім’я бота та його коротке ім’я) і отримати повідомлення з маркером бота та посиланням на документацію. Токен потрібно зберігати, бажано надійно, оскільки це єдиний ключ для авторизації та взаємодії з ботом.

При створенні бота необхідно вибрати ім'я користувача. Тоді змінити буде дуже важко.(рис. 3.1, 3.2)

Ім’я користувача бота виглядає як звичайне ім’я користувача, але має закінчуватися на «бот».

Можна було бачити ботів з іменами @pic, @vid, @sticker, @gamee — це офіційні боти Telegram.

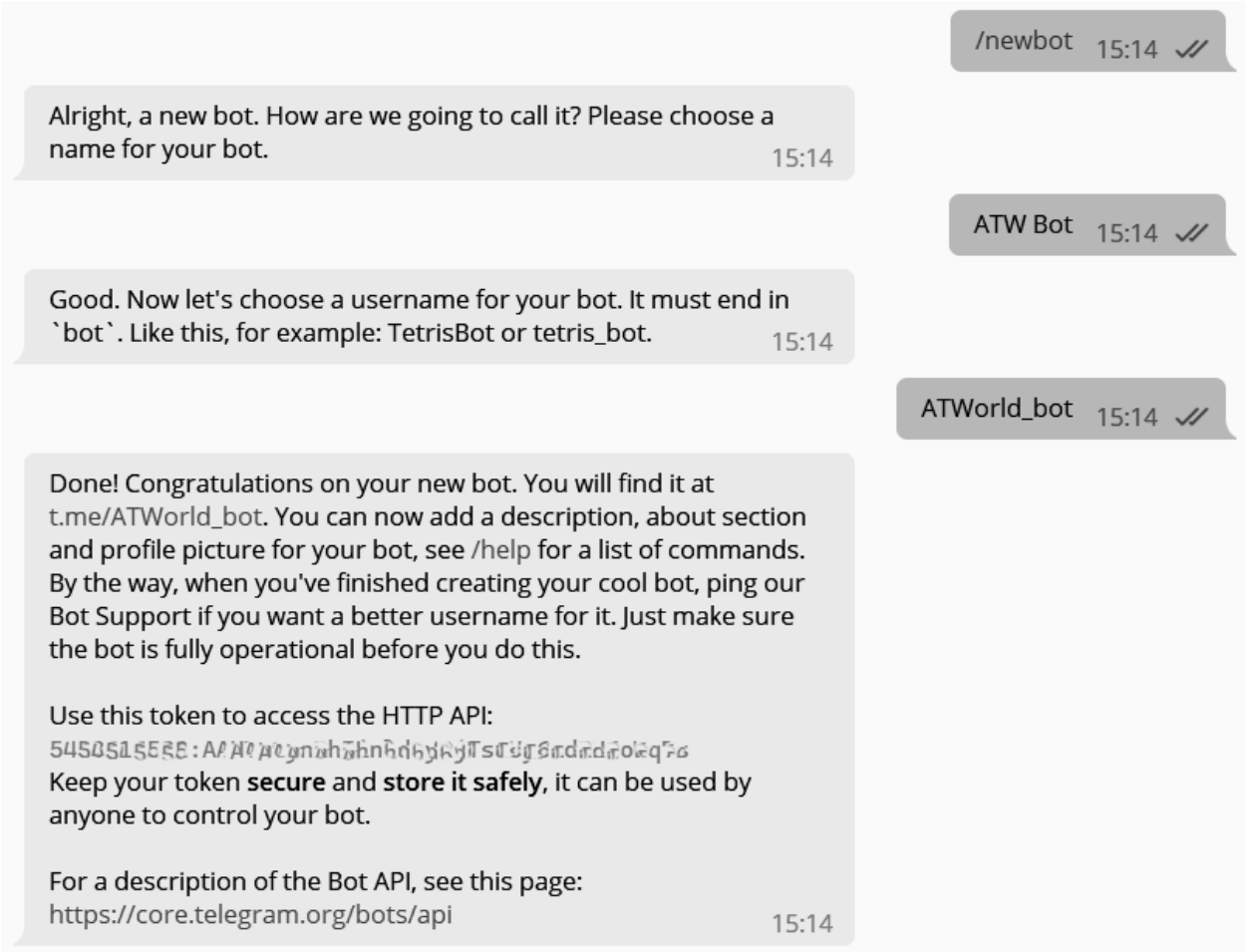

Рис. 3.1. Приклад заповнення початкової інформації

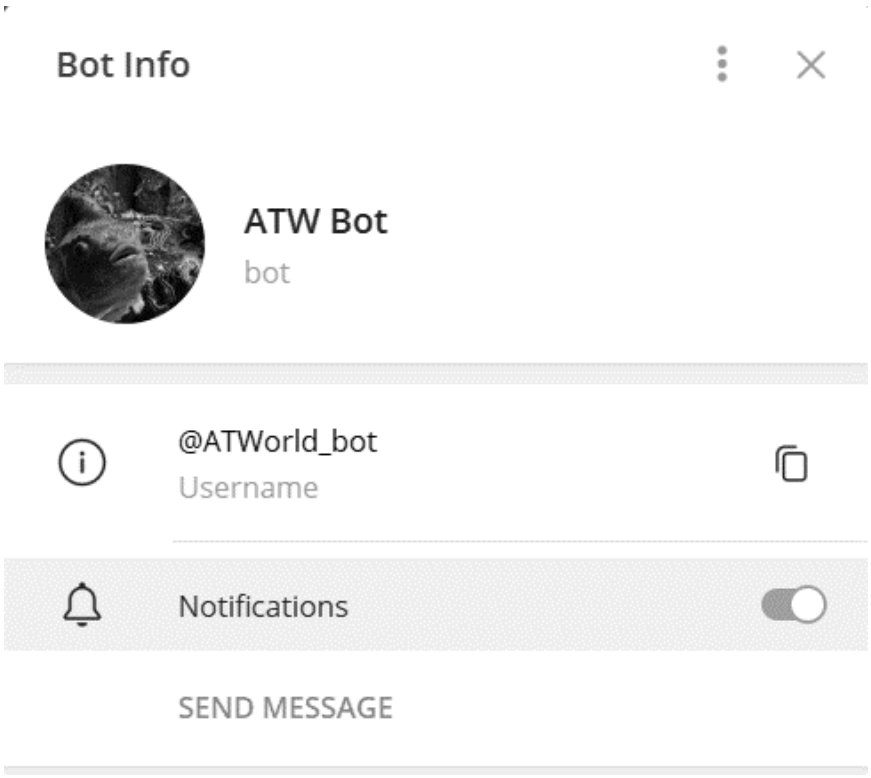

Рис. 3.2. Приклад профілю бота

Дизайн бота встановлюється в BotFather: меню /mybots → Редагувати бота. Там ви можете змінити:

- ім'я робота;
- опис (Description)— це текст, який бачать користувачі на початку розмови з ботом під заголовком «Що може цей бот?»;
- про бота (About) — це текст, який буде видно в профілі бота;
- аватар. На відміну від аватарів користувачів і чатів, аватари ботів не можна анімувати. Тільки картинки;
- команди - тут маються на увазі командні рядки в боті.

Перед початком написання основного коду бота потрібно підключити проект до Telegram та створити необхідну структуру коду (рис. 3.3).

config.py | main.py — ATW\BOT | functions.py | secbot.py

```python
import json
import os
from aiogram import Bot
from aiogram.dispatcher import Dispatcher
import sqlite3
from address import detect_address
from dotenv import load_dotenv

load_dotenv()

TOKEN = os.getenv('BOT_TOKEN')
BOT_ID = TOKEN.split(':')[0]
CHAT_ID = int(os.getenv('CHAT_ID'))
COLLECTION = detect_address(os.getenv('COLLECTION'))['bounceable']['b64url']
SUPPORT_CHAT_ID = int(os.getenv('SUPPORT_CHAT_ID'))

API_TOKEN = os.getenv('TONCENTER_API_KEY')

TONCENTER_BASE = 'https://toncenter.com/api/v2/'
headers = {
    'cookie': 'csrftoken=vO44viQbEDlbcKiLz56aaE8tLmjzc5XtVqUTZo6zgItxRNehF7VRGlCVdQR3dul9',
    'x-csrftoken': 'vO44viQbEDlbcKiLz56aaE8tLmjzc5XtVqUTZo6zgItxRNehF7VRGlCVdQR3dul9',
    'referer': 'https://beta.disintar.io/object/',
    'user-agent': 'Mozilla/5.0 (Windows NT 10.0; rv:91.0) Gecko/20100101 Firefox/91.0'}

bot = Bot(token=TOKEN)
dp = Dispatcher(bot)

connect = sqlite3.connect('database/atw_base.db', check_same_thread=False)
cursor = connect.cursor()

english = json.loads(open('jsons/en.json', 'r').read())

ukrainian = json.loads(open('jsons/ua.json', 'rb').read())

config = json.loads(open('jsons/configurations.json', 'rb').read())

```

Рис. 3.3. Створення структури

Для використання доступних стандартних функцій необхідно підключити бібліотеки, що будуть використовуватись на моменті синхронізації бота (рис. 3.4):

- Бібліотека json:

JSON — це синтаксис для зберігання та обміну даними.

JSON — це текст, написаний за допомогою нотації об’єктів JavaScript.

Python має вбудований пакет під назвою json, який можна використовувати для роботи з даними JSON.

- Бібліотека os:

Модуль os забезпечує багато функцій для роботи з операційною системою, і їх поведінка зазвичай не залежить від ОС, тому програми залишаються портативними.

- Бібліотека sqlite3:

SQLite — це бібліотека C, яка надає легку дискову базу даних, яка не потребує окремого серверного процесу та дозволяє отримати доступ до бази даних за допомогою нестандартного варіанту мови запитів SQL. Деякі програми можуть використовувати SQLite для внутрішнього зберігання даних. Також можна створити прототип програми за допомогою SQLite, а потім перенести код у більшу базу даних, таку як PostgreSQL або Oracle.

- Бібліотека dotenv:

Python-dotenv зчитує пари ключ-значення з файлу .env і може встановити їх як змінні середовища. Це допомагає в розробці програм.

При розробці веб-додатку або бота ми часто маємо справу з якимись секретними даними, різними токенами та паролями (ключі API, секрети веб-форм). Жорстке кодування цієї інформації, а тим більше зберігання її в загальнодоступній системі контролю версій, є дуже поганою ідеєю.

Найпростіший спосіб вирішити цю проблему — створити окремий файл конфігурації з усією конфіденційною інформацією та додати його до .gitignore. Недоліком цього підходу є те, що вам також потрібно зберігати шаблон файлу конфігурації в git і не забувати періодично оновлювати його.

Більш просунутий підхід полягає у використанні змінних середовища. Змінні середовища — це іменовані змінні, які містять текстову інформацію, яка може використовуватися запущеними програмами.

Щоб уникнути ручного налаштування змінних середовища кожного разу, коли ви перезапускаєте термінал, ви можете використовувати пакет python-dotenv. Дозволяє завантажувати змінні середовища з файлу .env у кореневій папці програми.

- Бібліотека aiogram:

Aiogram — це досить проста та повністю асинхронна структура для API Telegram Bot, написана на Python 3.7 з asyncio та aiohttp.

```
1  import json
2  import os
3  from aiogram import Bot
4  from aiogram.dispatcher import Dispatcher
5  import sqlite3
6  from address import detect_address
7  from dotenv import load_dotenv
8
```

Рис. 3.4. Імпорт бібліотек

Далі потрібно прописати всі необхідні змінні середовища. Файл зі змінними створюється окремо, та інформація імпортується покроково (рис. 3.5):

- BOT_TOKEN – Telegram bot API токен взятий у @BotFather;

Приклад: 1234567890:ABCDEFGHIJKLMNOPQRSTUVWXYZ.

- CHAT_ID – Telegram приватний чат для власників NFT;

Приклад: -1001234567890".

- COLLECTION – TON адреса NFT колекції;

Приклад: EQDwxUWb1ZxUarpj-mUF1gzD_jT4yiScHi_VL5AezWjsMT88".

- TONCENTER_API_KEY – toncenter.com API ключ з @tonapibot;
- CHAT_ID_SUPPORT – Telegram приватний чат для обробки запитів у службу підтримки;

Приклад: -1001234567891".

```
load_dotenv()

TOKEN = os.getenv('BOT_TOKEN')
BOT_ID = TOKEN.split(':')[0]
CHAT_ID = int(os.getenv('CHAT_ID'))
COLLECTION = detect_address(os.getenv('COLLECTION'))['bounceable']['b64url']
SUPPORT_CHAT_ID = int(os.getenv('SUPPORT_CHAT_ID'))

API_TOKEN = os.getenv('TONCENTER_API_KEY')
```

Рис. 3.5. Файл зі змінними

Прописуються локальні необхідні загальновідомі змінні (рис. 3.6).

```
TONCENTER_BASE = 'https://toncenter.com/api/v2/'
headers = {
    'cookie': 'csrftoken=vO44viQbEDlbcKiLz56aaE8tLmjzc5XtVqUTZo6zgItxRNehF7VRGlCVdQR3dul9',
    'x-csrftoken': 'vO44viQbEDlbcKiLz56aaE8tLmjzc5XtVqUTZo6zgItxRNehF7VRGlCVdQR3dul9',
    'referer': 'https://beta.disintar.io/object/',
    'user-agent': 'Mozilla/5.0 (Windows NT 10.0; rv:91.0) Gecko/20100101 Firefox/91.0'}
```

Рис. 3.6. Необхідні змінні

Далі потрібно прописати шлях доступу до бази даних, для збереження переліку користувачів (рис. 3.7).

```
bot = Bot(token=TOKEN)
dp = Dispatcher(bot)

connect = sqlite3.connect('database/atw_base.db', check_same_thread=False)
cursor = connect.cursor()
```

Рис. 3.7. Підключення бази даних

Останнім кроком буде під’єднання файлів для подальшого розділення користувачів з різних країн, та файл з режимами доступу до посилань на зовнішні ресурси (рис. 3.8).

```
32  english = json.loads(open('jsons/en.json', 'r').read())
33
34  ukrainian = json.loads(open('jsons/ua.json', 'rb').read())
35
36  config = json.loads(open('jsons/configurations.json', 'rb').read())
37
```

Рис. 3.8. Файли тексту

Весь текст з будь-якої частини бота записується в окремі файли, що полегшує подальше редагування та спрощує сприйняття коду. Поділяється текст на:

- *англомовну аудиторію:*

"no_longer_owner": "You don't have any NFT\nGo through the verification again if you want to access the chat again",

"start_message": "Welcome to ATW",

"start_verify_m": "Verification allows you to join chat of holders and reveal other useful functions",

"start_verify_b": "Verify ✅",

"start_verification": "Start verification",

"send_nft": "Send me your TON address",

"owner_verified": "You are verified!",

"is_it_ton": "Error! Try again.",

"done": "Done",

"no_nfts": "There are no NFTs here",

"send": "Send",

"from": "from",

"to": "to",

"comment": "with comment",

"scan_qr": "Or scan QR code below via Tonkeeper\n\nAfter the transfer, wait 30 seconds before clicking on the \"Done\" button",

"verified": "You are now verified!\n\nYour link: ",

"nft_verified": "Error! The verification might already been completed",

"dont_see": "No transaction! Please try again in few seconds",

"market": "GetGems page 💎",

"nft_share": "Share your NFTs",

"already_verified": "You are verified ✅",

"get_privileges": "What does it give?",

"privileges": "The following features are now available to you:",

"join_chat": "Join DAO chat 📜",

"i": "I",

"faq_mes": "Detailed information about the project 📄",

"start_faq": "FAQ",

"main_menu": "Menu",

"button_verification": "Verification",

"button_mint": "Buy",

"button_faq": "FAQ",

"button_question": "Make a question",

"question_mes": "Support center is under maintenance now",

"mint_mes_button": "Buy NFT",

"faq_mes_button": "How to buy?",

"mint_mes": "Want to buy NFT? Check our website",

"short_faq_mes_button": "Short FAQ 📋",

"main_menu": "Main menu",

"text_question": "Write your question in one message, then click 'Send'",

"send_question": "Send",

"thank_quest": "Thanks for your question!",

"big_faq_button": "Questions & Answers 📄",

"support_respond": "Support respond:"

- *україномовну аудиторію:*

"no_longer_owner": "Ви більше не володієте NFT\nПройдіть верифікацію, щоб отримати доступ до чату",

"start_message": "Вітаю Вас у боті проекту Around The World",

"start_verify_m": "Верифікація дозволить Вас увійти в чат холдерів та відкриє інші цікаві функції",

"start_verify_b": "Верифікуватись ✅",

"start_verification": "Почати верифікацію",

"send_nft": "Відправте адресу свого TON-гаманця:",

"owner_verified": "Власник цього гаманця вже пройшов верифікацію.",

"is_it_ton": "Виникла помилка! Спробуйте ще.",

"done": "Готово",

"no_nfts": "NFT не знайдені",

"send": "Відправте",

"from": "з",

"to": "на",

"comment": "з коментарем",

"scan_qr": "Або відскануйте QR код нижче за допомогою Tonkeeper\n\nПісля переказу почекайте 30 секунд перед тим, як натиснути кнопку «Готово»",

"verified": "Ви успішно пройшли верифікацію!\n\nПосилання на чат: ",

"nft_verified": "Помилка. Можливо, верифікація вже була пройдена",

"dont_see": "Я не бачу Ваш переказ. Будь ласка, спробуйте ще раз за кілька секунд",

"market": "Переглянути на GetGems 💎",

"nft_share": "Поділитись своїми NFT",

"already_verified": "Ви верифіковані ✅",

"get_privileges": "Що це значить?",

"privileges": "Вам тепер доступні наступні можливості:",

"join_chat": "Увійти в чат ДАО 📜",

"i": "Я",

"faq_mes": "Детальна інформація про проект 📄",

"start_faq": "FAQ",

"main_menu": "Головне меню",

"button_verification": "Пройти верифікацію",

"button_mint": "Купити",

"button_faq": "FAQ",

"button_question": "❓ Задати питання ❓",

"mint_mes_button": "Купити NFT",

"faq_mes_button": "Як купити?",

"mint_mes": "Хочете купити NFT? Ви можете зробити це у боті",

"short_faq_mes_button": "Короткий FAQ📖",

"main_menu": "Головне меню",

"text_question": "Напишіть Ваше питання в одному повідомленні, потім натисніть 'Надіслати'",

"send_question": "Надіслати",

"thank_quest": "Дякую за Ваше питання!",

"big_faq_button": "Детальний FAQ📚",

"support_respond": "Відповідь підтримки:"

У цих мовних файлах допускається використання тегів HTML (стандартизована мова гіпертекстової розмітки для перегляду веб-сторінок у браузері. Веб-браузери отримують HTML-документ із сервера через протоколи HTTP/HTTPS або відкривають його з локального диска, а потім інтерпретують код в інтерфейс, який буде відображатися на екрані монітора.) для розмітки тексту, та його оформлення, а також символи Юнікод (стандарт кодування символів, який включає символи майже всіх письмових мов світу. Стандарт зараз домінує в Інтернеті).

Основний код програми знаходиться у файлі main.py. Для запуску конкретного головного файлу у мультифайловому проекті використовується конструкція if __name__ == '__main__' (рис. 3.9).

```
306    if __name__ == '__main__':
307        logging.basicConfig(level=logging.INFO)
308        executor.start_polling(dp, on_startup=on_startup)
```

Рис. 3.9. Конструкція if __name__ == '__main__'

Також необхідно записувати логи для контролю виникнення можливих помилок програми.

Start_polling з рисунку вище означає початок «прослуховування» ефіру на рахунок змін стану бота та запитів користувачів. Насамперед, потрібно створити базу даних, якщо її не було ще створено, та запустити модуль scheduler (для регулярної перевірки користувачів з таблиці на наявність NFT у під'єднаному гаманці)(рис. 3.10). База даних поділяється на 4 таблиці:

1. verify: основна таблиця зі зв'язками користувач-адреса та код для верифікації через транзакці;

2. contest: таблиця для зберігання адрес, які не потребують перевірки на наявність NFT;

3. joined: список усіх користувачів бота;

4. random: тимчасовий список ідентифікаційних номерів користувачів та їх кодів, задля неможливості впізнання акаунта у Telegram за адресою гаманця.

Модуль scheduler дозволяє виконувати функції з певною періодичністю, наприклад, один раз за годину (рис. 3.11).

```
259  async def on_startup(dispatcher):
260      """
261      Створення таблиць у БД, якщо цього було зроблено раніше
262      verify: основна таблиця зі зв'язками користувач-адреса та код для верифікації через транзакцію
263      contest: таблиця для зберігання адрес, які не потребують перевірки на наявність NFT
264      joined: список усіх користувачів бота
265      """
266      try:
267          cursor.execute(
268              'CREATE TABLE IF NOT EXISTS verify (tgid int8 NOT NULL, "owner" text NULL, CONSTRAINT verify_tgid_key UNIQUE (tgid));')
269          cursor.execute(
270              'CREATE TABLE IF NOT EXISTS contest ("owner" text NOT NULL);')
271          cursor.execute(
272              'CREATE TABLE IF NOT EXISTS joined ("tgid" int8 NOT NULL);')
273          cursor.execute(
274              'CREATE TABLE IF NOT EXISTS random ("tgid" int8 NOT NULL, "code" int8 NULL, CONSTRAINT random_tgid_key UNIQUE (tgid));')
275          connect.commit()
276      except:
277          pass
278      asyncio.create_task(scheduler())
```

Рис. 3.10. Створення таблиць бази даних

```python
async def scheduler():
    schedule.every().hour.do(check_holders)
    while True:
        await schedule.run_pending()
        await asyncio.sleep(1)

```

Рис. 3.11. Модуль scheduler

Один раз за годину буде відбуватись перевірка усіх користувачів, що пройшли верифікацію на наявність NFT на їх гаманці, та користувач автоматично буде видалений у разі непройденої перевірки (рис. 3.12).

```python
async def check_holders():
    """
    Регулярна перевірка на наявність користувачів NFT. У разі продажу та/або передачі всіх NFT, верифікація знімається
    """
    cursor.execute(f"select * from verify")
    response = cursor.fetchall()
    owners = [x for x in response if x[1] is not None and x[1] != '']
    for owner in owners:
        try:
            tgid, owner_address = owner
            nfts = await get_user_nfts(owner_address)
            cursor.execute(f"select * from contest where owner = '{owner_address}'")
            contest = cursor.fetchall()
            if len(nfts) + len(contest) == 0:
                await kick_user(tgid)
        except Exception as e:
            logging.error(f"Failed to Check {owner} on NFTs\n"
                          f"{e}")
        await asyncio.sleep(1)

```

Рис. 3.12. Модуль видалення користувача

Алгоритм проходження перевірки:

1. З бази даних беруться адреси гаманців користувачів.

2. Створюється список з гаманців.

3. Запитом до блокчейну збираються NFT на гаманці (рис. 3.13).

4. Користувач видаляється, якщо його немає у таблиці збережених адрес, що не потребують перевірки, та кількість його NFT колекції дорівнює 0(рис. 3.14).

```python
async def get_user_nfts(address):
    address = (await get_ton_addresses(address))['b64url']
    await asyncio.sleep(1)
    all_nfts = json.loads(requests.get(f"https://tonapi.io/v1/nft/searchItems",
                                       params={"owner": address,
                                               "collection": COLLECTION,
                                               "include_on_sale": "true",
                                               "limit": "50",
                                               "offset": 0}).text)['nft_items']
    nfts = []
    for nft in all_nfts:
        try:
            name = nft['metadata']['name']
            address = (await get_ton_addresses(nft['address']))['b64url']
            image = nft['metadata']['image']
            nfts += [{'address': address, 'name': name, 'image': image}]
        except:
            pass
    return nfts

```

Рис. 3.13. Запит до блокчейну

```python
async def kick_user(tgid):
    """
    Позбавлення користувача верифікації та права бути присутнім у чаті.
    """
    try:
        cursor.execute(
            f"delete from verify where tgid = '{tgid}'")
        connect.commit()
    except Exception as e:
        logging.error(f"Failed to Delete {tgid} from Database\n"
                      f"{e}")
    try:
        await bot.ban_chat_member(CHAT_ID, tgid)
        await bot.unban_chat_member(CHAT_ID, tgid, only_if_banned=True)
    except Exception as e:
        logging.warning(f"Failed to Kick {tgid} from Chat {CHAT_ID}\n"
                        f"{e}")
    try:
        await bot.send_chat_action(tgid, 'typing')
        await bot.send_message(tgid,
                               f'{ukrainian["no_longer_owner"]}\n\n{english["no_longer_owner"]}')
    except Exception as e:
        logging.warning(f"Failed to Send Message to {tgid}\n"
                        f"{e}")

```

Рис.3.14. Видалення користувача

Послідовність виконання верифікації

Перше, що бачить користувач, коли заходить до бота – кнопку START, механізм оброблення натиснення кнопки передбачає додавання окремої клавіатури, що вміщає в собі 4 кнопки(рис. 3.15, 3.16):

1. Початок верифікації.

2. Кнопка мінту (де купити).
3. Кнопка FAQ (питання, що часто задають).
4. Кнопка щоб задати питання службі підтримки.

Рис. 3.15. Початкова сторінка Бота

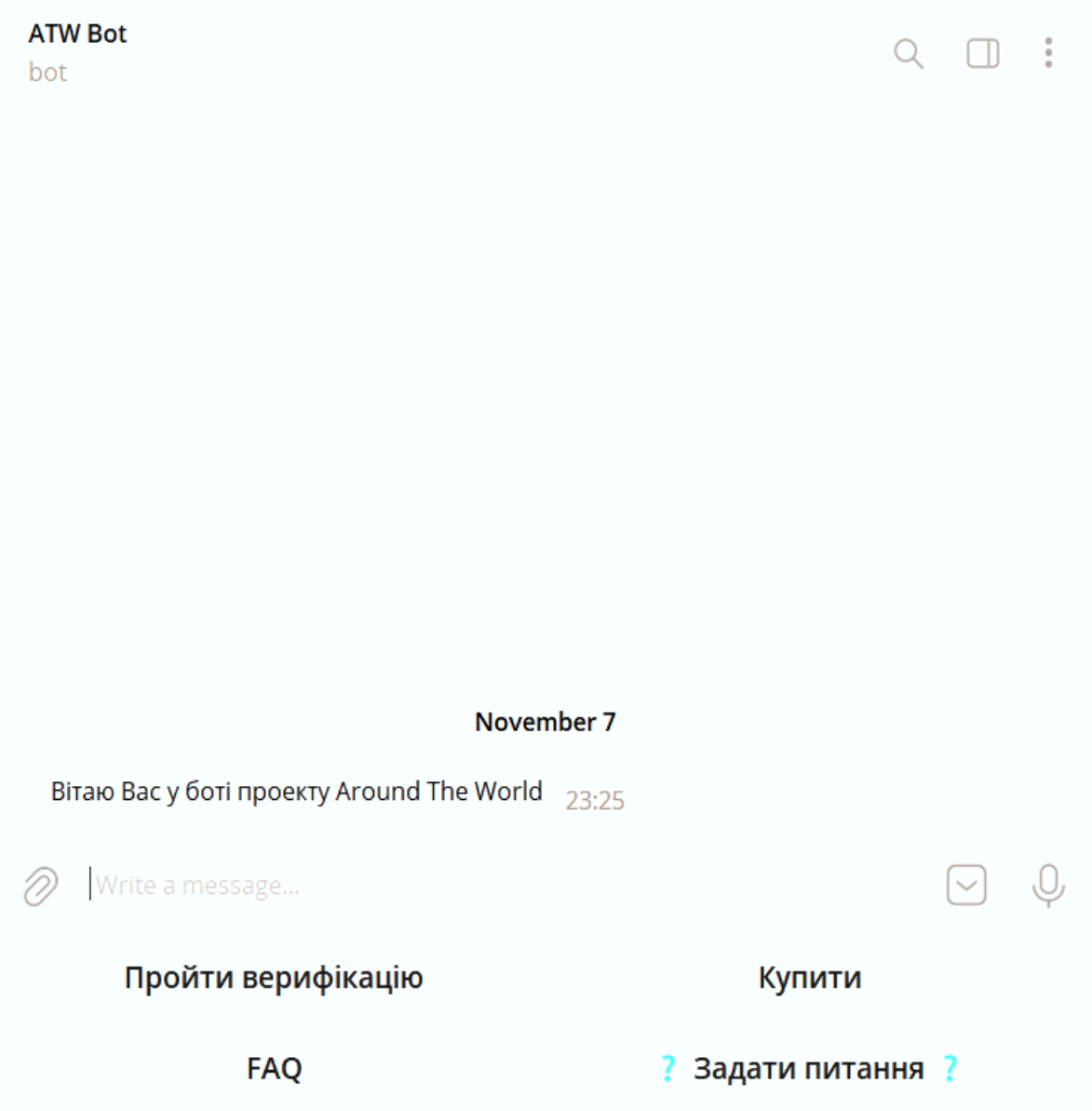

Рис. 3.16. Головне меню

Процес верифікації

Процес верифікації починається з перевірки користувача, записаний він вже у базу даних чи ні, якщо ні, то буде доступна кнопка отримання реквізитів (реквізити необхідні для отримання мінімальної транзакції 0.01ton як доказ належності гаманця цьому користувачу), або кнопка з доступними привілегіями від пройденої верифікації (рис. 3.17, 3.18).

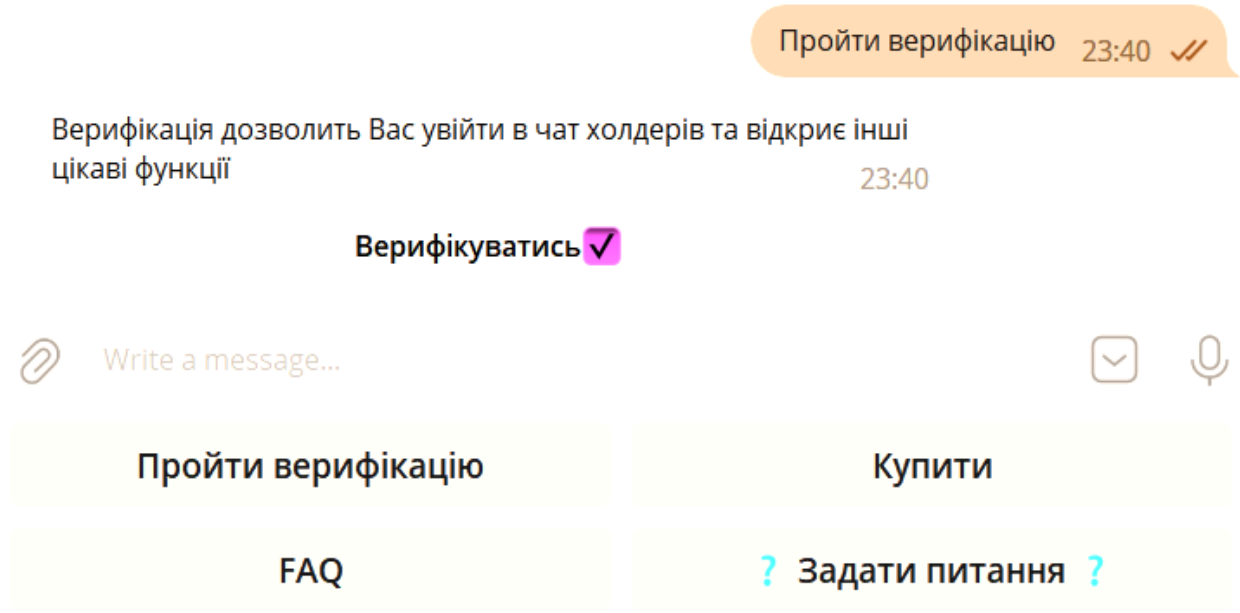

Рис. 3.17 Кнопка верифікації

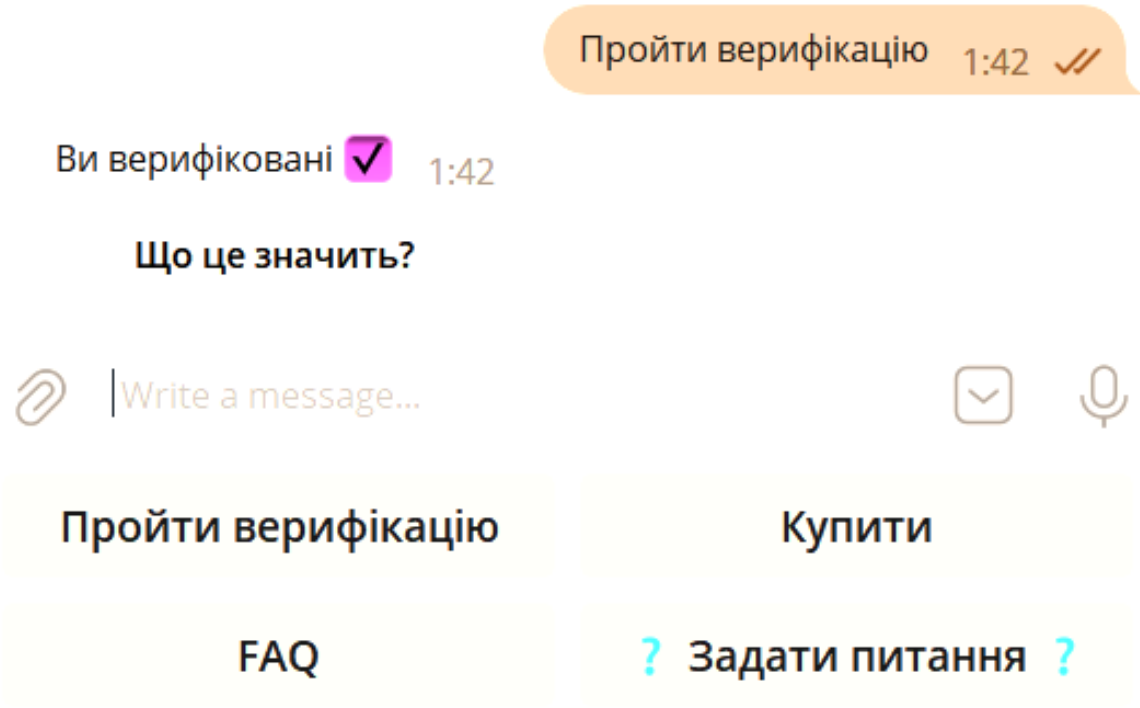

Рис. 3.18. Кнопка верифікації

Функція, що обробляє кнопку «Пройти верифікацію» відправляє різні повідомлення від імені бота, текст та зворотній зв’язок яких залежить від наявності верифікації у користувача (рис. 3.19)

```python
@dp.callback_query_handler(lambda c: c.data == 'verification')
async def start_ver_message(callback_query: types.CallbackQuery):

    await bot.send_chat_action(callback_query.message.chat.id, 'typing')
    language = english if callback_query.from_user.language_code != 'ua' else ukrainian
    cursor.execute(f"select * from verify where tgid = '{callback_query.message.from_user.id}'")
    verified = len(cursor.fetchall())
    if verified:
        await bot.send_message(
            callback_query.message.chat.id, language['already_verified'], reply_markup=InlineKeyboardMarkup().add(
            InlineKeyboardButton(text=language['get_privileges'], callback_data='privileges')))
    else:
        await bot.send_message(
            callback_query.message.chat.id, language['start_verify_m'],
                                reply_markup=InlineKeyboardMarkup().add(
                                    InlineKeyboardButton(text=language['start_verification'], callback_data='verify')))

```

Рис. 3.19. Функція, що обробляє кнопку «Пройти верифікацію»

Для початку верифікації необхідно надіслати адресу свого гаманця у блокчейні TON, бот повідомить коли саме це потрібно зробити, у який момент процесу (рис. 3.20).

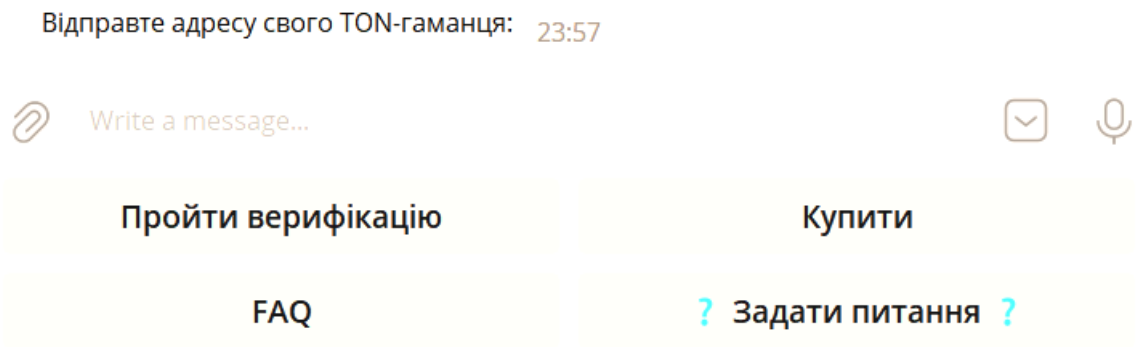

Рис. 3.20. Запит гаманця

Після перевірки правильності введеного гаманця бот опише звідки (введений гаманець) та куди (гаманець NFT колекції у блокчейні TON) потрібно надіслати транзакцію, а також з яким коментарем. Код коментаря буде згенеровано спеціально для кожного користувача, та видалено після отримання транзакції (рис 3.21, 3.22).

```
    elif len(msg.text) == 48:
        try:
            await asyncio.sleep(1)
            owner_address = (await get_ton_addresses(msg.text))['b64url']
            cursor.execute(f"select * from verify where owner = '{owner_address}'")
            response = cursor.fetchall()
            cursor.execute(f"select * from contest where owner = '{owner_address}'")
            contest = cursor.fetchall()
            if len(response) == 0:
                await asyncio.sleep(1)
                nfts = await get_user_nfts(owner_address)
                if len(nfts) != 0 or len(contest) > 0:
                    rand_id = await get_random_id(msg.from_user.id)
                    await bot.send_message(msg.chat.id,
                                           f'{language["send"]} <a href="http://qrcoder.ru/code/?ton%3A%2F%2Ftransfer% \
                                           2FEQAEKjxqAk-fwb07HV06VhEJKt2-qEm_HwBxVXCvzP-o666_%3Famount%3D10000000%26te \
                                           xt%3Dverify{rand_id}&4&0"> </a><b>0.01 TON</b>\n\n{language["from"]} <code> \
                                           {owner_address}</code>\n\n{language["to"]} <code>EQAEKjxqAk-fwb07HV06VhEJKt \
                                           2-qEm_HwBxVXCvzP-o666_</code>\n\n'
                                           f'{language["comment"]} <code>verify{rand_id}</code>\n\n{language["scan_qr"]}'
                                           parse_mode=ParseMode.HTML,
                                           reply_markup=InlineKeyboardMarkup().add(
                                               InlineKeyboardButton(text=language['done'],
                                                                    callback_data=owner_address)))
                else:
                    await msg.reply(language['no_nfts'])
            else:
                await msg.reply(language['owner_verified'])
        except Exception as e:
            print(e)

```

Рис. 3.21. Після перевірки гаманця

Рис. 3.22. Реквізити оплати

Функція генерації коду, та дешифрування дозволять унеможливити впізнання власника гаманця у Telegram (рис. 3.23, 3.24).

```
66  async def get_random_id(tgid):
67      random_id = random.randint(10000000, 99999999)
68      print(random_id)
69      try:
70          cursor.execute(f"delete from random where tgid = '{tgid}'")
71          connect.commit()
72      except:
73          pass
74      try:
75          cursor.execute(
76              f"insert into random (tgid, code) values ('{tgid}', '{random_id}') on conflict do nothing")
77          connect.commit()
78      except:
79          print("get_random_id error")
80      return random_id
81
```

Рис. 3.23. Функція генерації коду

```
83  async def check_random_id(tgid):
84      try:
85          cursor.execute(
86              f"SELECT code FROM random WHERE tgid = '{tgid}'")
87          table_number = cursor.fetchall()
88          print(table_number)
89      except Exception as e:
90          raise e
91      return table_number[0][0]
92
```

Рис. 3. 24. Функція дешифрування коду

Надсилання транзакції

Криптовалютний гаманець — це інструмент для взаємодії з криптовалютами в блокчейні. Він дозволяє створювати та керувати адресами для зберігання та передачі цифрових активів. По суті, це програма з інтерфейсом і різними функціями для управління адресою та збереженими в ній криптоактивами.

Гаманці бувають кастодіальні та некастодіальні:

Кастодіальний гаманець – це додаток для зберігання та переказу криптовалют, особливість якого полягає в тому, що його адміністратор (кастодіан) керує адресами користувачів або має доступ до їх особистих

ключів. Крім того, довірені особи повинні пройти процес перевірки особи (KYC).

Основним недоліком кастодіальних криптогаманців є можливість для зберігача отримати доступ до криптоактивів клієнтів.

Керівництво біржі володіє закритими та відкритими ключами адрес, де зберігаються криптовалюти клієнтів. Завдяки централізованому зберіганню великої кількості коштів, торгові майданчики є частою мішенню хакерів. Зломи можуть призвести до того, що окремі користувачі можуть втратити свої криптовалюти.

Крім того, криптобіржа є легальною організацією, яка зобов'язана дотримуватися законодавства та вимог правоохоронних органів. За їхнім запитом він може надати інформацію про клієнтів і заморозити кошти в їх гаманці. Наприклад, у разі застосування санкцій чи конфіскації майна за рішенням суду.

Також неприємним моментом може стати відсутність доступу до власних ресурсів під час технічної роботи, адже кастодіан може за потреби відключити доступ користувачам.

Некастодіальний гаманець – криптовалютний гаманець, який не підлягає зберіганню, забезпечує повний контроль над коштами автора адреси, не передаючи нікому свої особисті ключі. Такий додаток не може заморожувати або управляти коштами користувачів, але не несе відповідальності за їх безпеку.

Зазвичай це програма, яку можна завантажити на ПК, мобільний пристрій або браузер. Створення блокчейн-адреси через програму, яка не підлягає опіці, не потребує KYC.

Популярні некастодіальні гаманці додатково захищені шляхом публікації їх вихідного коду. Це дозволяє незалежним експертам перевірити, чи програма справді безпечна. Крім того, такі проекти часто підтримуються всією спільнотою програмістів.

Найпопулярнішими криптогаманцями для блокчену TON є:

- TonKeeper;
- TonWallet;
- TonHub.

Для оплати верифікації був використаний інтерфейс TonWallet розширення для браузеру Google Chrome. Надсилання транзакції поділяється на кроки:

- заповнення реквізитів (рис. 3.25):
  - адреса отримувача;
  - кількість монет;
  - коментар (зазвичай не є обов'язковим).
- перевірка адреси отримувача та підтвердження транзакції (рис. 3.26);
- очікування проведення транзакції валідаторами мережі (рис. 3.27);
- отримання позитивного результату проведення платежу (рис. 3.28).

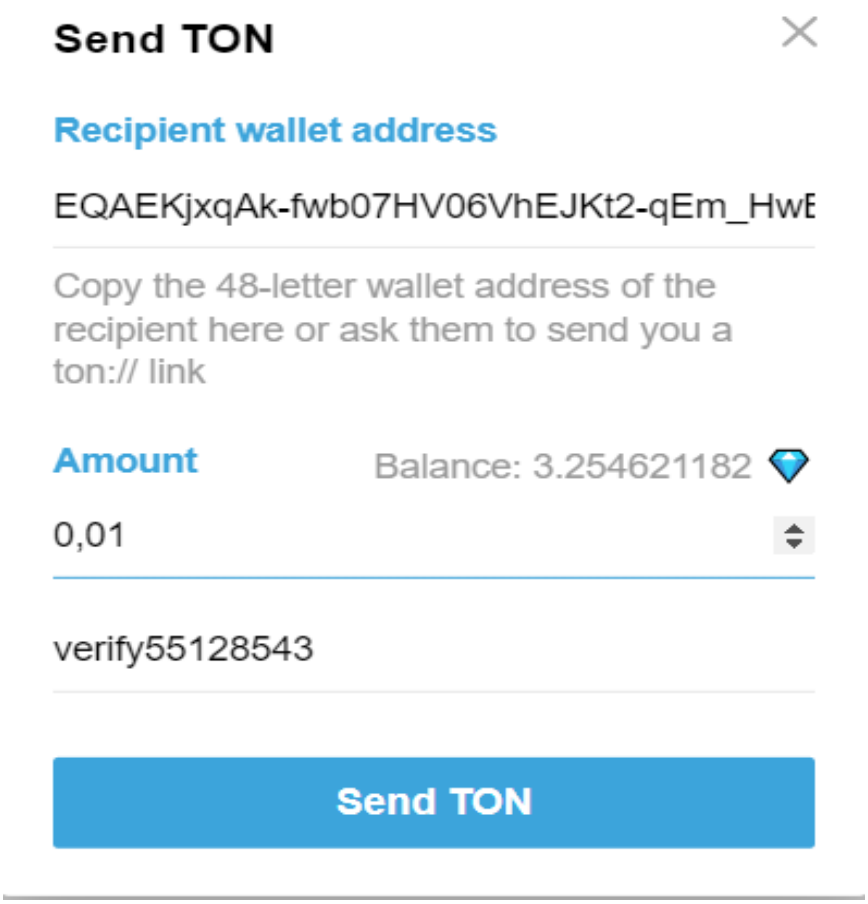

Рис. 2. 25. Заповнення адреси

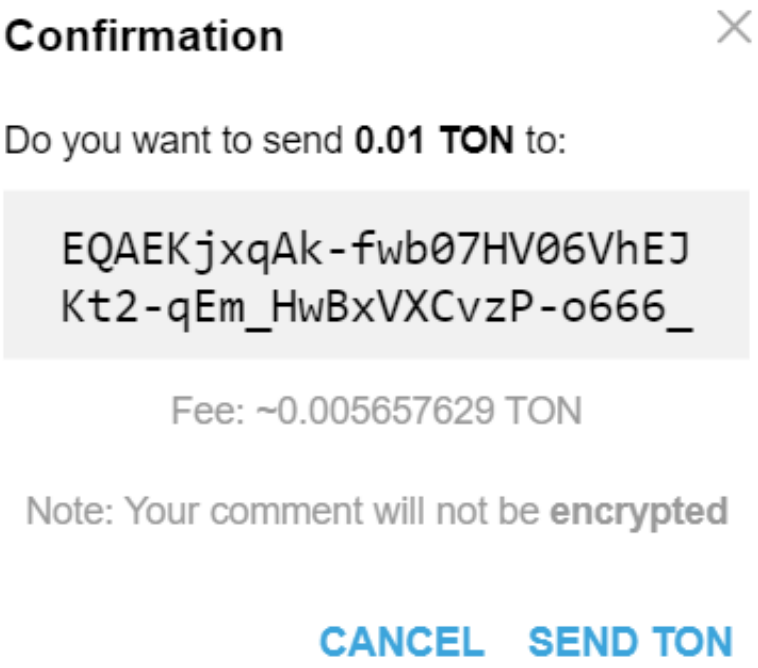

Рис. 3.26. Перевірка заповнення

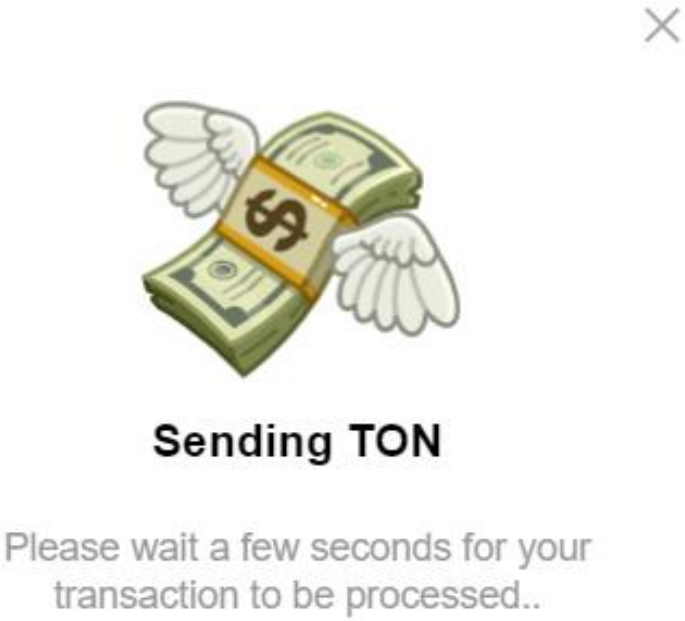

Рис. 3.27. Очікування проведення транзакції

Done!

0.01 TON have been sent

CLOSE

Рис. 3.28. Отримання позитивного результату

Після проведення транзакції бажано почекати 30 секунд, та натиснути кнопку «Готово». Бот перевірить транзакцію та надішле персоналізоване посилання для доступу в закритий чат (рис. 3.29, 3.30, 3.31).

```python
@dp.callback_query_handler(lambda c: c.data != 'verify')
async def verify2(callback_query: types.CallbackQuery):
    """
    Перевірка переказу для верифікації
    """
    await bot.send_chat_action(callback_query.message.chat.id, 'typing')
    language = english if callback_query.from_user.language_code != 'ua' else ukrainian
    owner_address = callback_query.data
    cursor.execute(f"select * from verify where owner = '{owner_address}'")
    owner_verify = len(cursor.fetchall())
    owner_addresses = await get_ton_addresses(owner_address)
    try:
        if owner_verify != 0:
            raise AssertionError
        result = json.loads(requests.get(
            f'{TONCENTER_BASE}getTransactions?address=EQAEKjxqAk-fwb07HV06VhEJKt2-qEm_HwBxVXCvzP-o666_&limit=10&to_lt=0&archival=false',
            headers=headers).text)['result']
        is_verified = False
        for i in result:
            transaction = i['in_msg']
            value = transaction['value']
            msg = transaction['message']
            source = transaction['source']
            rand_id = await check_random_id(callback_query.from_user.id)
            if (int(value) >= 10000000) and (msg == f"verify{rand_id}") and (
                    source == owner_addresses['b64url']):
                is_verified = True
                break
        if is_verified:
            cursor.execute(
                f"insert into verify (tgid, owner) values ('{callback_query.from_user.id}', '{owner_addresses['b64url']}') on conflict d
            cursor.execute(f"delete from random where tgid = '{callback_query.from_user.id}'")
            connect.commit()
            link = await bot.create_chat_invite_link(CHAT_ID, name=f'{callback_query.from_user.id}',
                                                     creates_join_request=True)
```

Рис. 3.29. Перевірка переказу

```
            await bot.send_message(callback_query.message.chat.id,
                                   f'{language["verified"]}{link.invite_link.split("://")[1]}',
                                   reply_markup=InlineKeyboardMarkup())
        else:
            try:
                await callback_query.answer(language['dont_see'], show_alert=True)
            except:
                await bot.send_message(callback_query.message.chat.id, language['dont_see'])
    except Exception as e:
        try:
            await callback_query.answer(language['owner_verified'], show_alert=True)
        except:
            await bot.send_message(callback_query.message.chat.id, language['owner_verified'])
        logging.info(f"{owner_address} is already verified")
    try:
        await callback_query.answer()
    except:
        pass

```

Рис. 3. 30. Перевірка переказу

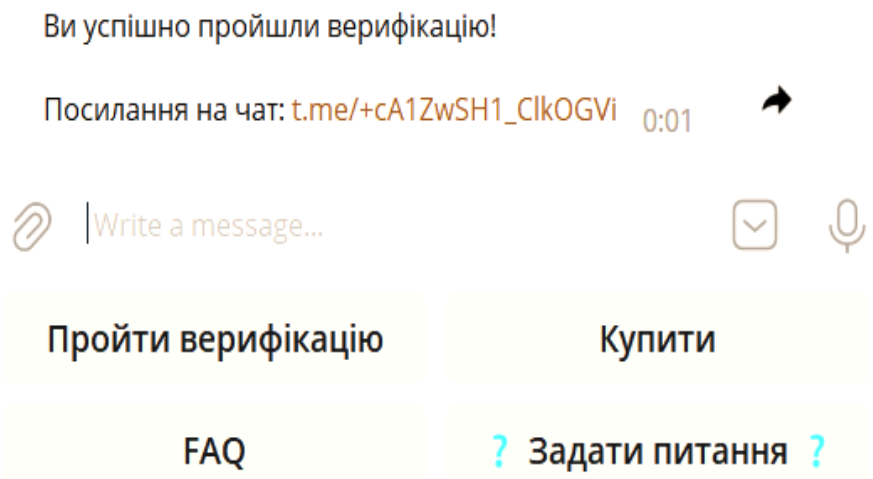

Рис. 3. 31. Персоналізоване посилання

Враховуючи, що посилання на доступ до чату приватне – натиснувши на нього створиться запит на доступ, та буде автоматично підтверджений ботом (рис. 3.32).

```python
@dp.chat_join_request_handler()
async def islegal(update: types.ChatJoinRequest):
    """
    Обробка запитів на вступ до чату. Якщо посилання не призначалося користувачеві, заявка відхиляється.
    """
    if update.invite_link.creator.id == int(BOT_ID):
        if str(update.invite_link.name) == str(update.from_user.id):
            await update.approve()
        else:
            await update.decline()
        await bot.revoke_chat_invite_link(CHAT_ID, update.invite_link.invite_link)

```

Рис. 3.32. Автоматичне підтвердження запиту

Після натиснення на кнопку FAQ користувач може ознайомитись з питаннями, що часто задають, або дізнатись варіанти, як можна купити NFT (рис. 3.33).

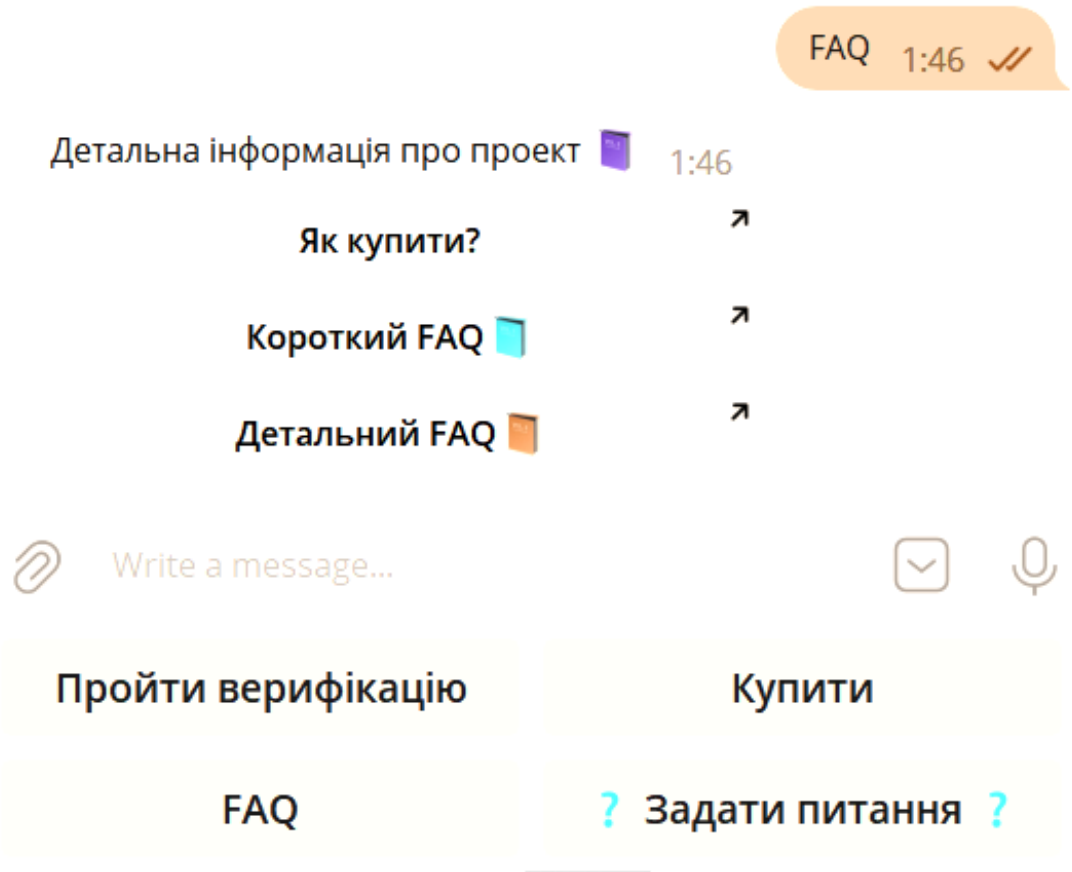

Рис. 3.33. Питаннями, що часто задають

Кнопка «Купити» відкриє доступ до можливих варіантів придбання NFT (рис. 3.34).

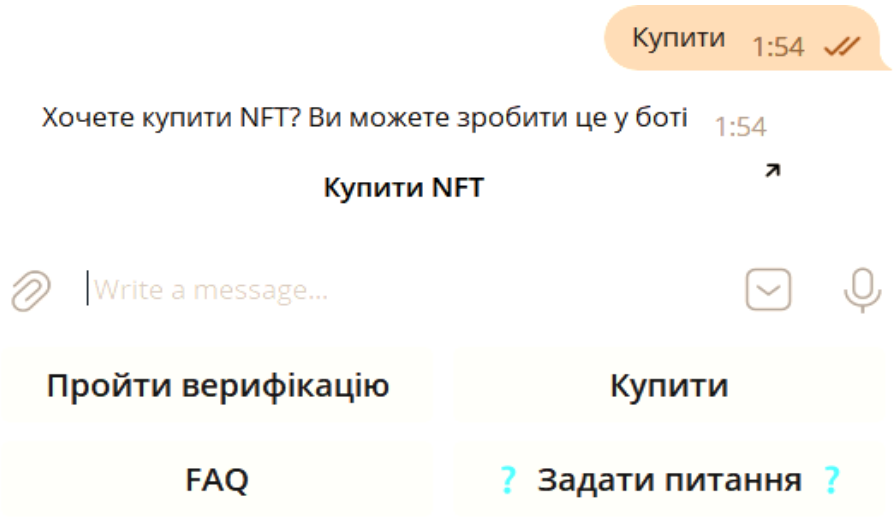

Рис. 3.34. Кнопка «Купити»

Також реалізований механізм спілкування зі службою підтримки. Кнопка «Задати питання» відкриває до нього доступ. Необхідно написати в одному повідомленні питання, або описати проблему та натиснути «Надіслати» (рис. 3.35, 3.36).

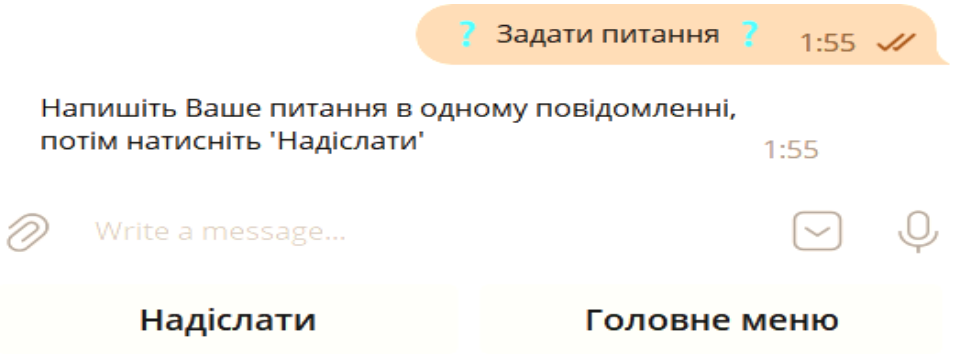

Рис. 3.35. Задати питання

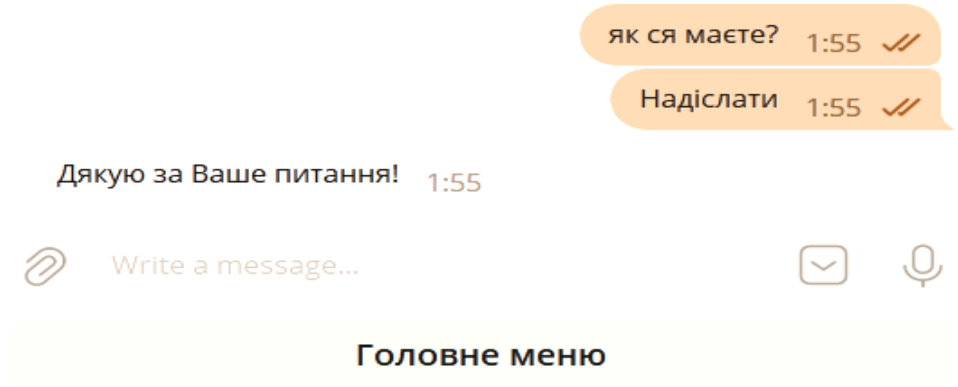

Рис. 3.36. Надіслати питання

Повідомлення буде переслано ботом у чат підтримки, де спеціально навчені адміністратори зможуть написати відповідь, та за допомогою відповіді на конкретно взяте повідомлення допомогти потрібному користувачу (рис. 3.37, 3.38, 3.39).

```
219  @dp.message_handler(content_types='text')
220  async def reply_to_user(msg: types.Message):
221      if (msg.chat.id == SUPPORT_CHAT_ID):
222          try:
223              language = english if msg.from_user.language_code != 'ua' else ukrainian
224              replyText = language['support_respond']+ "\n" + str(msg.text)
225              try:
226                  await bot.send_message(msg.reply_to_message.forward_from.id, replyText)
227              except:
228                  userId = int(msg.reply_to_message.text)
229                  await bot.send_message(userId, replyText)
230          except:
231              pass
232
```

Рис. 3.37. Функція пересилання повідомлення

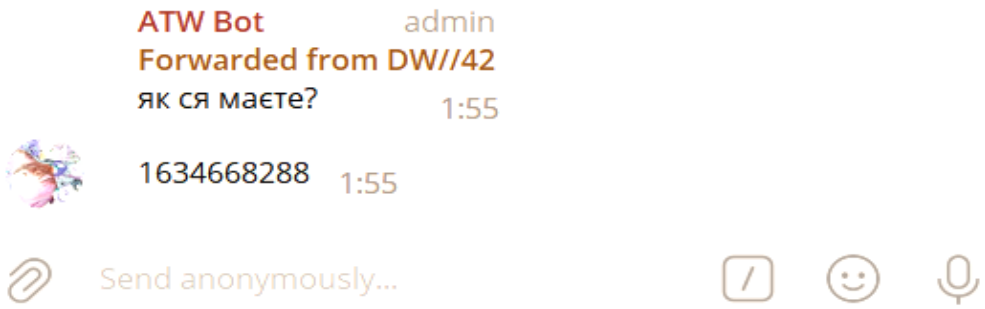

Рис. 3. 38. Повідомлення надійшло

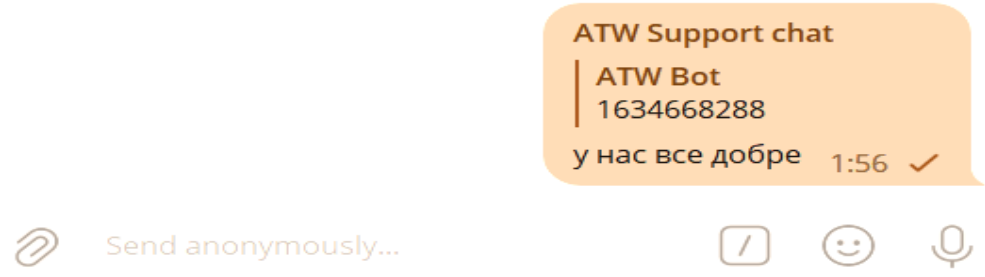

Рис. 3.39. Пересилання відповіді

Користувач отримає відповідь на своє питання (рис. 3.40).

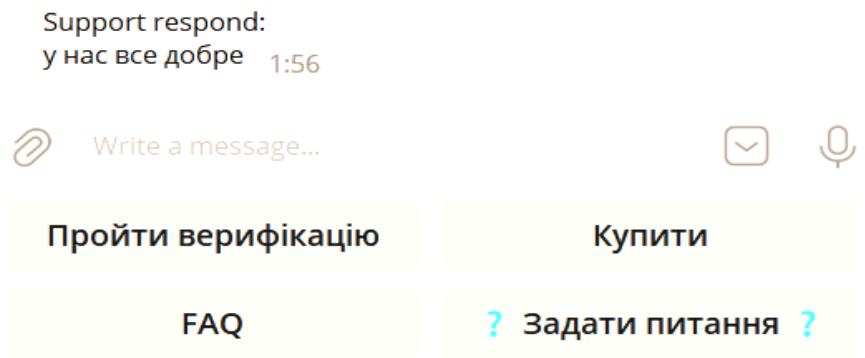

Рис. 3.40. Відповідь від служби підтримки

Для повноцінного запуску коду, та розміщення на сервері потрібно створити контейнер використовуючи Docker.

Контейнери — це спосіб стандартизувати розгортання програми та відокремити її від решти інфраструктури. Екземпляр програми працює в ізольованому середовищі, яке не впливає на операційну систему хоста. Розробникам не потрібно думати про середовище, в якому працюватиме їхній додаток, чи будуть існувати необхідні налаштування та залежності. Вони просто створюють додаток, упаковують усі залежності та налаштування в одне зображення. Потім ви можете запустити цей образ на інших системах, не турбуючись про те, що програма не запуститься.

Docker — це платформа для розробки, доставки та запуску контейнерних програм. Docker дозволяє створювати контейнери, автоматизувати їх запуск і розгортання, а також керувати їхнім життєвим циклом. Дозволяє запускати кілька контейнерів на одній хост-машині.

Контейнерізація схожа на віртуалізацію, але вони не однакові. Віртуалізація запускає повноцінний хост на гіпервізорі з власним віртуальним обладнанням та операційною системою. У цьому випадку ви можете запустити іншу ОС в одній ОС. У випадку контейнеризації процес виконується безпосередньо з ядра основної операційної системи і не віртуалізує апаратне забезпечення. Це означає, що програма-контейнер може працювати лише в тій самій операційній системі, що й основна. Контейнери не віртуалізують обладнання, тому споживають менше ресурсів.

Створюється проект через командний рядок командою: «***docker build -t test-bot .***»(рис. 3.41).

```
S:\ATW\BOT>docker build -t test-bot .
[+] Building 1.4s (10/10) FINISHED
 => [internal] load build definition from Dockerfile                                                    0.0s
 => => transferring dockerfile: 32B                                                                     0.0s
 => [internal] load .dockerignore                                                                       0.0s
 => => transferring context: 2B                                                                         0.0s
 => [internal] load metadata for docker.io/library/python:alpine3.10                                    1.1s
 => [1/5] FROM docker.io/library/python:alpine3.10@sha256:152b1952d4b42e360f2efd3037df9b645328c0cc6fbe9c63decbffb  0.0s
 => [internal] load build context                                                                       0.1s
 => => transferring context: 223.05kB                                                                   0.1s
 => CACHED [2/5] WORKDIR /bot                                                                           0.0s
 => CACHED [3/5] COPY requirements.txt .                                                                0.0s
 => CACHED [4/5] RUN  apk add --no-cache python3 postgresql-libs &&  apk add --no-cache --virtual .build-deps gcc  0.0s
 => CACHED [5/5] COPY . .                                                                               0.0s
 => exporting to image                                                                                  0.1s
 => => exporting layers                                                                                 0.0s
 => => writing image sha256:bcd5f0956e49589e5378be8098b129f0993ad8b6e217f563f6540be2d815e478            0.0s
 => => naming to docker.io/library/test-bot                                                             0.0s
```

Рис. 3.41. Створені контейнерів

Також після створення контейнеру потрібно його запустити командою: «***docker run --name=BOT -it test-bot:latest***»(рис. 3.42).

```
S:\ATW\BOT>docker run --name=BOT -it test-bot:latest
INFO:aiogram:Bot: ATW Bot [@ATWorld_bot]
INFO:aiogram.dispatcher.dispatcher:Start polling.
```

Рис. 3.42. Запуск Бота

ВИСНОВКИ ДО РОЗДІЛУ 3

У цьому розділі було розглянуто етапи створення бота у месенджері Telegram, рисунками наведено приклади основних функцій та код, що дозволяє розмістити їх у правильному порядку відносно один одного.

Проект можливо зібрати у контейнері, або розмістити на сайті хостингу. Різні типи представлення проекту та модульний тип розміщення функцій дозволяють виявляти помилки при додаванні нових функцій та полегшують сприйняття коду у разі необхідності зміни функцій (модулів), або додавання нових сторонніх процесів у логіку роботи програми бота.

ВИСНОВКИ

Метою кваліфікаційної роботи була розробка робочого Telegram бота з основною функцією верифікації власників невзаємозамінних токенів у блокчейні The Open Network, який на початку розробки та до конфліктів з комісією з цінних паперів міг називатись Telegram Open Network. Таке підтвердження володіння має намір спростити керування доступом до інформації серед довірених користувачів та «холдерів» (власників) токенів. Також за допомогою інформації, що збирає бот, є можливість надсилання персоналізованих пропозицій довіреним аккаунтам.

У процесі написання пояснювальної записки до кваліфікаційної роботи було розглянуто по порівняно проклади середовищ розробки, виявлено основні їх функції, переваги та недоліки, а також обрано серед них найкращий, що підходив би цілям проектування та розробки основного програмного забезпечення Telegram бота.

Розробку біткойна Сатоші Накамото у 2009 році часто сприймали як радикальну розробку грошей і валюти, оскільки це був перший приклад цифрового активу, який водночас не має ні підтримки, ні «внутрішньої цінності», ні централізованого емітента чи контролера. Однак іншою, мабуть, більш важливою частиною експерименту з біткойнами є базова технологія блокчейну як інструмент розподіленого консенсусу, і увага швидко починає переключатися на цей інший аспект біткойна. Зазвичай згадувані альтернативні застосування технології блокчейн включають використання цифрових активів на блокчейні для представлення користувальницьких валют і фінансових інструментів («кольорові монети»), право власності на основний фізичний пристрій («розумна власність»), незамінні активи, такі як доменні імена. («Namecoin»), а також більш складні програми, які включають пряме керування цифровими активами за допомогою частини коду, що реалізує довільні правила («розумні контракти») або навіть «децентралізовані автономні організації» (DAO) на основі блокчейну.

Код бота написаний однією з найкращих мов програмування Python. Перш за все, Python — це поширена мова програмування, яка широко

використовується для веб-розробки, машинного навчання, штучного інтелекту, наукових обчислень тощо. Перш ніж перейти до кращих бібліотек цієї мови, коротко опишемо його основні переваги, завдяки яким Python став таким популярним серед програмістів.

Переваги включають: простий синтаксис високого рівня, особливо в порівнянні з конкурентами; елементарність використання - Python успішно використовують як експерти, так і новачки; величезна бібліотека – вона має велику бібліотеку для всіх мов програмування.

Ця мова програмування зосереджена на читабельності коду за допомогою англійських ключових слів. Використовується на таких відомих джерелах як Reddit, NASA, PBS. Свою популярність мова набула в основному завдяки своїй простоті. Саме ця якість спонукає розробників постійно створювати популярні бібліотеки Python.

Отже, розроблений Telegram бот має не тільки сучасну модульну архітектуру, а й працює з одним із найуспішніших та найпопулярніших месенджерів, що безумовно є найголовнішою його перевагою серед інших можливих майбутніх аналогів. Також такий вид архітектури дозволяє легко масштабувати бота та розширювати його функціонал. Алгоритм може бути використаний під час розробки ботів для інших проектів на основі колекцій невзаємозамінних токенів в інших структурах заснованих на технології блокчейну.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. What Is A Telegram Bot [Електронний ресурс] – Режим доступу до ресурсу: https://chatbotslife.com/what-is-a-telegram-bot-reasons-to-use-bot-for-telegram-46b0d0579337

2. Telegram displays the power of bots [Електронний ресурс] – Режим доступу до ресурсу: https://www.messengerpeople.com/the-10-best-telegram-bots-in-2022/

3. Bitcoin As A State Transition System [Електронний ресурс] – Режим доступу до ресурсу: https://ethereum.org/en/whitepaper/

4. What is blockchain technology [Електроний ресурс] – Режим доступу до ресурсу: https://www.ibm.com/topics/what-is-blockchain#anchor-2038331631

5. Kesonpat N. Organization Legos: The State of DAO Tooling. [Електроний ресурс] – Режим доступу до ресурсу: https://medium.com/1kxnetwork/organization-legos-the-state-of-dao-tooling-866b6879e93e

6. Mining of Bitcoin [Електронний ресурс] – Режим доступу до ресурсу: https://ethereum.org/en/whitepaper/

7. All aspects of DAO: types, commonly used tools and evolution history. [Електроний ресурс] – Режим доступу до ресурсу: https://blockcast.cc/news/all-aspects-of-dao-types-commonly-used-tools-and-evolution-history

8. Usman W., Chohan, The Decentralized Autonomous Organization and Governance Issues [Електроний ресурс] – Режим доступу до ресурсу: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3082055

9. The DAO (organization) [Електроний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/The_DAO_(organization)

10. The Maker Protocol: MakerDAO's Multi-Collateral Dai (MCD) System [Електроний ресурс] – Режим доступу до ресурсу: https://makerdao.com/ru/whitepaper

11. Brief Description of TON Components [Електронний ресурс] – Режим доступу до ресурсу: https://ton-blockchain.github.io/docs/ton.pdf

12. Intelligent Coding Assistance [Електронний ресурс] – Режим доступу до ресурсу: https://www.jetbrains.com/pycharm/features/coding_assistance.html

13. Aiogram [Електронний ресурс] – Режим доступу до ресурсу: https://docs.aiogram.dev/en/latest/

14. Python [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Python_(programming_language)

15. Telegram [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/en/post/543676/

16. Pyrogram [Електронний ресурс] – Режим доступу до ресурсу: https://snyk.io/advisor/python/pyrogram

17. Top 5 aiogram [Електронний ресурс] – Режим доступу до ресурсу: https://snyk.io/advisor/python/aiogram/example

18. Decentralized Autonomous Organizations: Concept, Model, and Applications [Електронний ресурс] – Режим доступу до ресурсу: https://ieeexplore.ieee.org/abstract/document/8836488

19. Blockchain technology [Електронний ресурс] – Режим доступу до ресурсу: https://www.sciencedirect.com/science/article/abs/pii/S0065245819300257

20. PyCharm [Електронний ресурс] – Режим доступу до ресурсу: https://de.wikipedia.org/wiki/PyCharm

21. Bot [Електронний ресурс] – Режим доступу до ресурсу: https://www.myrasecurity.com/de/bot/

22. Положення про дипломні роботи [Електронний ресурс] – Режим доступу до ресурсу: http://aem.nau.edu.ua/%D0%9F%D0%BE%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%BD%D1%8F%20%D0%BF%D1%80%D0%BE%20%D0%B4%D0%B8%D0%BF%D0%BB%D0%BE%D0%BC%D0%BD%D1%96%20%D1%80%D0%BE%D0%B1%D0%BE%D1%82%D0%B8%202017.pdf