|  |
| --- |
| Министерство науки и высшего образования РФ Федеральное государственное автономное образовательное учреждение высшего профессионального образования **«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»** |
| Институт космических и информационных технологий |
| Кафедра вычислительной техники |

**ОТЧЕТ О ПРЕДДИПЛОМНОЙ ПРАКТИКЕ**

|  |
| --- |
| 090301 Информатика и вычислительная техника |
| Тема: «Крипто-портфолио» |

|  |  |  |
| --- | --- | --- |
| Руководитель |  | А.П. Яблонский |
| подпись, дата |
| Студент КИ19-07Б, 031940422 |  | А.Р. Голубев |
| подпись, дата |

Красноярск 2023

СОДЕРЖАНИЕ

[Введение 3](#_Toc137116814)

[1 Спецификация требований к системе 5](#_Toc137116815)

[1.1 Существующие аналоги 5](#_Toc137116816)

[1.2 Разработка прецедентов 6](#_Toc137116817)

[1.3 Выводы по главе 11](#_Toc137116818)

[2 Проектирование 13](#_Toc137116819)

[2.1 Определение сущностей разрабатываемой системы 13](#_Toc137116820)

[2.2 Разработка диаграмм последовательности 14](#_Toc137116821)

[2.3 ER-диаграмма базы данных 15](#_Toc137116822)

[2.4 Выводы по главе 17](#_Toc137116823)

[3 Реализация и тестирование 18](#_Toc137116824)

[3.1 Создание бота в Telegram 18](#_Toc137116825)

[3.2 Установка дополнительных библиотек 19](#_Toc137116826)

[3.3 Описание констант и функций 19](#_Toc137116827)

[3.3.1 Константы 19](#_Toc137116828)

[3.3.2 Функции 20](#_Toc137116829)

[3.4 Тестирование бота 26](#_Toc137116830)

[3.5 Выводы по главе 26](#_Toc137116831)

[Заключение 27](#_Toc137116832)

[Список использованных источников 28](#_Toc137116833)

Введение

С появлением и ускоренным развитием цифровых технологий, вопросы связанные с криптовалютами становятся все более актуальными и востребованными для обсуждения. Криптовалюты постепенно переходят от непринятого статуса к общепринятым цифровым активам, вовлекая все больше и больше людей в сферу цифровых финансов.

На сегодняшний день, когда криптовалютные активы занимают существенное место в портфелях инвесторов, а иногда и торговля криптовалютами становится основным источником дохода, возникает необходимость в средствах для управления и контроля этими активами. В связи с этим, предметом исследования данной дипломной работы становится разработка инструмента для управления криптовалютным портфелем.

Быстрый рост криптовалютного рынка и все большее распространение цифровых активов в повседневной жизни подчеркивают важность и актуальность данной работы. Средства управления криптовалютами, такие как цифровые кошельки, биржи и портфолио-трекеры, уже существуют на рынке, но они часто сложны в использовании для неподготовленных пользователей или не предоставляют полной информации, которая может быть нужна для эффективного управления активами.

Тем не менее, технологический прогресс и развитие программного обеспечения открывают новые возможности для создания удобных и доступных инструментов управления криптовалютами. В данной работе исследуется возможность использования некоторых из этих технологий для создания инструмента управления криптовалютным портфелем.

Целью дипломной работы является разработка и реализация инструмента, позволяющего пользователям отслеживать состояние своего криптовалютного портфеля.

Поставленная цель обуславливает необходимость решения следующих задач:

* изучение существующих аналогов и подходов к управлению криптовалютными активами;
* анализ требований к функционалу инструментов отслеживания состояния криптовалютного портфеля;
* выбор подходящих технологий для его реализации, проектирование архитектуры приложения;
* реализация и тестирование функционала, анализ полученных результатов и возможности для дальнейшего развития проекта.

В ходе работы над выпускной квалификационной работой был изучен ряд методов и возможностей информации, связанной с криптовалютами, разработкой программного обеспечения и использованием современных технологий. Этот опыт и накопленные знания позволили реализовать проект, который, как ожидается, будет полезен для людей, интересующихся криптовалютами и ищущих удобный инструмент для управления своими активами.

# Спецификация требований к системе

Для разработки спецификации требований данного проекта, был проведен анализ уже существующих решений с открытым исходным кодом на GitHub.

## Существующие аналоги

По запросу «crypto-portfolio» [1] на GitHub найдено 1944 репозиториев.

Такие проекты как «crypto-portfolio» могут представлять собой мобильные приложения, web приложения, desktop приложения.

Для анализа были выбраны три популярных в сообществе репозитория: trentpiercy/trace [2], emmtte/Cryptocurrency-Portfolio [3], huwwp/cryptop [4]. Ниже мы сравним их.

Trentpiercy/trace: Данный проект предлагает пользователю интерфейс для отслеживания криптовалютного портфеля. Он также содержит функцию для просмотра истории транзакций и отслеживания прибыли или убытка от инвестиций в криптовалюту. Проект реализован в виде мобильного приложения на языке Dart.

Emmtte/Cryptocurrency-Portfolio: Данный проект также предлагает пользовательский интерфейс для отслеживания криптовалютного портфеля. Он включает функцию, которая позволяет пользователям видеть разбивку своего портфеля по различным криптовалютам. Проект представлен в виде google sheets, использован язык JavaScript.

Huwwp/cryptop: Данный проект предлагает API для отслеживания криптовалютного портфеля. Он поддерживает интеграцию с различными криптовалютными обменами и кошельками ETH, и позволяет пользователям просматривать свой агрегированный портфель криптовалюты. Примеры использования и инструкции по установке представлены в документации проекта. Проект активен, с 33 звездами на GitHub. Данный проект представлен в виде desktop приложения на языке Python.

Проанализировав аналоги было принято решение реализовать проект в виде Telegram бота, так как приведенные аналоги требуют установку дополнительных средств для работы с крипто-портфолио. Во-первых, Telegram является одним из наиболее популярных мессенджеров в мире, что обеспечивает широкий охват аудитории. Во-вторых, боты в Telegram обладают удобным функционалом и могут быть настроены на выполнение различных задач, включая управление криптовалютными портфелями. Это позволяет пользователям управлять своими портфелями в любое время и в любом месте, где доступен интернет. В-третьих, благодаря возможности мгновенного обмена сообщениями, боты в Telegram могут предоставлять пользователю актуальную информацию о состоянии его портфеля в реальном времени. В целом, реализация проекта в виде Telegram бота может упростить управление криптовалютными активами и сделать его более доступным для широкого круга пользователей.

## Разработка прецедентов

На рисунке 1.1 представлена диаграмма вариантов использования, отражающая действия пользователя.

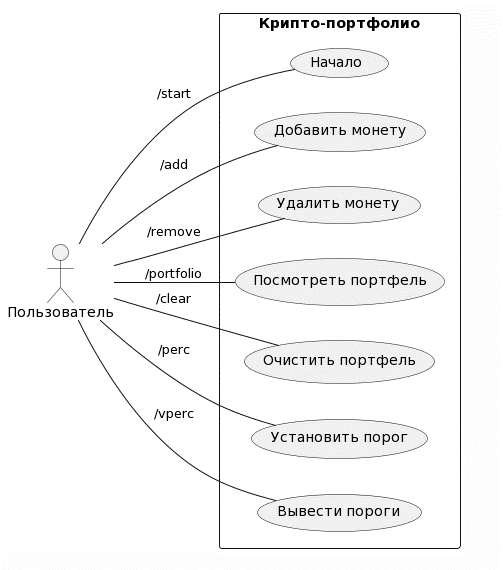


Рисунок 1.1 — Диаграмма вариантов использования

**Название прецедента:** Начало.

**Предусловие:** Пользователь начинает взаимодействовать с ботом в Telegram.

**Основная последовательность:**

1. Пользователь отправляет команду ‘/start’ (либо нажал кнопку во всплывающем меню «/start»).
2. Бот проверяет, есть уже этот пользователь в системе или нет.
3. Если пользователь уже есть в системе, бот продолжает взаимодействие с пользователем.
4. Если пользователя нет в системе, бот создает новую запись о пользователе в системе и привязывает его Telegram ID к этой записи.

**Постусловие:** Пользователь зарегистрирован в системе и может продолжить взаимодействие с ботом.

**Название прецедента:** Добавление монеты в портфель.

**Предусловие:** Пользователь уже зарегистрирован в системе и взаимодействует с ботом в Telegram.

**Основная последовательность:**

1. Пользователь отправляет сообщение о добавление с идентификатором монеты.
2. Бот получает текущую цену этой монеты.
3. Пользователь указывает количество монеты, которое хочет добавить в свой портфель.
4. Система обновляет портфель пользователя, добавляя новую монету или обновляя количество существующей.
5. Система записывает новую историю цен для этой монеты в портфеле пользователя.
6. Бот отправляет сообщение пользователю с подтверждением обновления.

**Постусловие:** Портфель пользователя обновлен, и новая история цен добавлена в систему.

**Название прецедента:** Отображение портфеля.

**Предусловие:** Пользователь уже зарегистрирован в системе и взаимодействует с ботом в Telegram. У пользователя есть монеты в его портфеле.

**Основная последовательность:**

1. Пользователь отправляет команду /portfolio.
2. Система извлекает информацию о портфеле пользователя.
3. Система рассчитывает текущую стоимость каждой монеты в портфеле, исходя из текущей цены и количества.
4. Бот отправляет пользователю сообщение с детальной информацией о его портфеле.

**Постусловие:** Пользователь получает текущую информацию о своем портфеле.

**Альтернативный сценарий**: Если у пользователя нет монет в портфеле, бот отправляет соответствующее сообщение.

**Альтернативное постусловие**: Пользователь уведомлен о том, что его портфель пуст.

**Название прецедента:** Продажа монеты из портфеля.

**Предусловие:** Пользователь уже зарегистрирован в системе и взаимодействует с ботом в Telegram. У пользователя есть монеты в его портфеле.

**Основная последовательность:**

1. Пользователь отправляет идентификатор монеты, которую хочет продать, и количество продаваемых монет.
2. Система проверяет наличие монеты и достаточное количество в портфеле пользователя.
3. Система обновляет портфель пользователя, вычитая количество продаваемых монет.
4. Система записывает историю транзакции продажи.
5. Бот отправляет пользователю сообщение с подтверждением продажи.

**Постусловие:** Портфель пользователя обновлен, и история транзакций продажи записана в системе.

**Альтернативный сценарий**: Если у пользователя недостаточно монет для продажи, бот отправляет соответствующее сообщение.

**Альтернативное постусловие**: Пользователь уведомлен о том, что у него недостаточно монет для продажи.

**Название прецедента:** Очистка портфеля.

**Предусловие:** Пользователь уже зарегистрирован в системе и взаимодействует с ботом в Telegram. У пользователя есть монеты в его портфеле.

**Основная последовательность:**

1. Пользователь отправляет команду /clear.
2. Система удаляет все монеты из портфеля пользователя.
3. Бот отправляет пользователю сообщение с подтверждением очистки.

**Постусловие:** Портфель пользователя полностью очищен.

**Альтернативный сценарий**: Если у пользователя нет монет в портфеле, бот отправляет соответствующее сообщение.

**Альтернативное постусловие**: Пользователь уведомлен о том, что его портфель уже пуст.

**Название прецедента:** Установка порога изменения цены.

**Предусловие:** Пользователь уже зарегистрирован в системе и взаимодействует с ботом в Telegram. У пользователя есть монеты в его портфеле.

**Основная последовательность:**

1. Пользователь отправляет команду /perc.
2. Бот отображает встроенную клавиатуру с выбором промежутка времени.
3. Пользователь выбирает промежуток времени.
4. Бот предлагает пользователю ввести процентное изменение.
5. Пользователь вводит процентное изменение.
6. Система обновляет пороговое значение для выбранного временного промежутка.
7. Бот подтверждает обновление порогового значения.

**Постусловие:** Пользователь успешно установил порог изменения цены для выбранного промежутка времени.

**Альтернативный сценарий**: Если пользователь вводит недействительное число, бот просит ввести допустимое значение.

**Альтернативное постусловие**: Если ввод недействителен, пользователь получает запрос на повторный ввод.

**Название прецедента:** Просмотр установленных порогов изменения цены.

**Предусловие:** Пользователь уже зарегистрирован в системе и взаимодействует с ботом в Telegram. У пользователя уже установлены пороги изменения цены.

**Основная последовательность:**

1. Пользователь отправляет команду /vperc.
2. Система извлекает все установленные пороги изменения цены пользователя.
3. Бот отправляет пользователю сообщение со всеми установленными порогами изменения цены.

**Постусловие:** Пользователь видит все свои установленные пороги изменения цены.

**Альтернативный сценарий**: Если у пользователя не установлены пороги изменения цены, бот отправляет соответствующее сообщение.

**Альтернативное постусловие**: Пользователь уведомлен о том, что не установлены пороги изменения цены.

## Выводы по главе

В первой главе было проведено комплексное исследование в области управления криптовалютными портфелями. Был осуществлен анализ существующих аналогов, что позволило выявить недостатки, такие как отсутствие удобных оповещений о значительных изменениях в стоимости активов.

Была выделена актуальность разработки инструмента для управления криптовалютным портфелем в Telegram, который решит указанные проблемы. Было представлено описание предполагаемого использования бота с помощью прецедентов.

Вывод первой главы подтверждает значимость создания данного бота, который способен предложить более усовершенствованные функции уведомлений и обеспечить удобство использования для широкого круга пользователей криптовалют.

# Проектирование

## Определение сущностей разрабатываемой системы

В процессе разработки и реализации крипто-портфеля были выделены основные сущности, которые отражают ключевые аспекты работы с системой. Это сущности TelegramUser, Portfolio, UserCoin и CoinHistory.

TelegramUser представляет пользователя Telegram, который регистрируется в системе для управления своим крипто-портфелем. Каждый пользователь имеет уникальный идентификатор в системе Telegram (telegram\_id), который используется для идентификации пользователя в нашей системе. Также для каждого пользователя создается связанная с ним Django-модель User, которая обеспечивает дополнительные возможности управления пользователями в системе.

Portfolio представляет крипто-портфель пользователя, в котором хранится информация о всех криптовалютах, которые пользователь решил добавить в свой портфель. Каждый пользователь имеет только один портфель, который может содержать информацию о множестве криптовалют.

UserCoin представляет информацию о конкретной криптовалюте в портфеле пользователя. В каждом объекте UserCoin хранится информация о символе криптовалюты, ее текущей цене, цене покупки, количестве монет, которые пользователь хранит в своем портфеле, и времени добавления этой монеты в портфель.

CoinHistory представляет историю изменения цены конкретной монеты. В каждом объекте CoinHistory хранится информация о цене монеты и времени, когда эта цена была актуальна. Эта информация может быть полезна для анализа динамики изменения цены монеты и принятия решений о покупке или продаже.

## Разработка диаграмм последовательности

Диаграммы последовательности помогают визуализировать и лучше понять процессы, происходящие в системе. В рамках этой работы были разработаны диаграммы последовательности для двух основных прецедентов: "Покупка монеты пользователем" и "Продажа монеты пользователем".

В прецеденте "Покупка монеты пользователем" отражены основные шаги, которые происходят при покупке монеты: пользователь отправляет команду /buy с указанием идентификатора монеты и количества купленных монет. Бот обращается к API для получения текущей цены монеты. Система добавляет информацию о монете в портфель пользователя и обновляет историю цен монеты. Диаграмма представлена на рисунке 2.1.

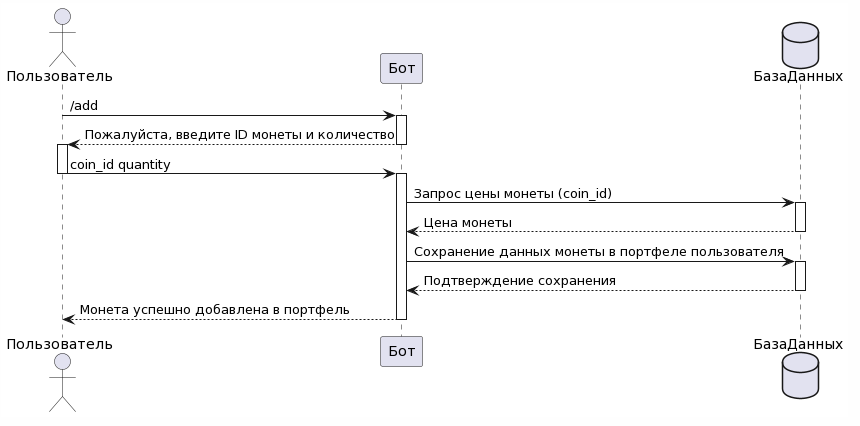


Рисунок 2.1 – Диаграмма последовательности покупки монеты пользователем

В прецеденте "Продажа монеты пользователем" отражены основные шаги, которые происходят при продаже монеты: пользователь отправляет команду /sell с указанием идентификатора монеты и количества. Бот проверяет, есть ли такая монета в портфеле пользователя и достаточно ли монет для продажи. Если все условия выполнены, система удаляет информацию о монете из портфеля пользователя. Диаграмма представлена на рисунке 2.2.

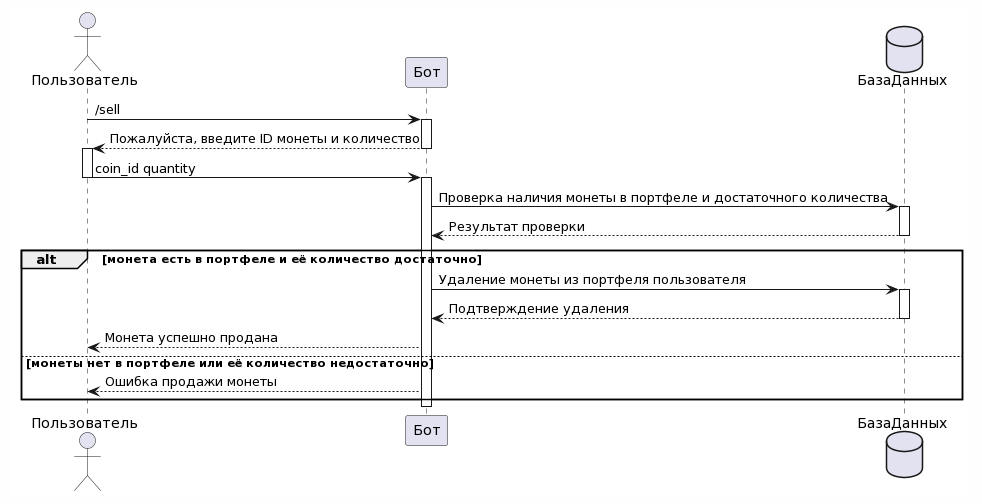


Рисунок 2.2 – Диаграмма последовательности продажи монеты пользователем

## ER-диаграмма базы данных

ER-диаграмма представляет собой визуальную модель структуры данных, которая облегчает понимание и анализ используемых в проекте сущностей и отношений между ними.

В данном проекте база данных была создана с использованием SQLite3. Основным преимуществом данного выбора является то, что SQLite3 является компактной встраиваемой СУБД. В отличие от большинства других СУБД, SQLite не требует отдельного сервера - это упрощает развертывание и поддержку приложения. К тому же, SQLite3 встроено в Django, что обеспечивает дополнительное удобство использования и сокращает количество необходимых настроек.

Сам Django был выбран в качестве основного фреймворка для разработки приложения, и одной из причин этого выбора является его ORM система. С помощью ORM и моделей Django мы можем эффективно взаимодействовать с базой данных, используя Python-код вместо прямых SQL-запросов. Это упрощает процесс разработки и делает код более читаемым.

Визуальное представление сущностей и связей между ними помогает лучше понять структуру данных, которую мы используем для реализации функциональности нашего крипто-портфельного бота. ER-диаграмма для базы данных проекта представлена ниже на рисунке 2.3.

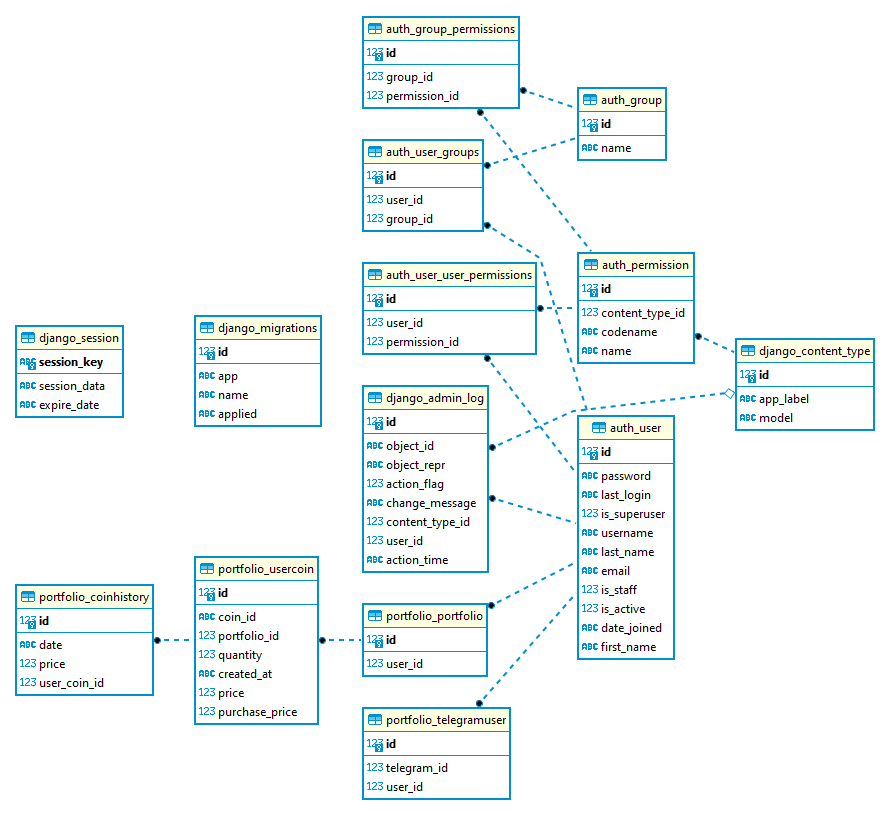


Рисунок 2.3 — ER-диаграмма базы данных

## Выводы по главе

Во второй главе работы были проведены ключевые этапы проектирования проекта - криптовалютного портфеля в виде бота для Telegram. Были исследованы основные сущности, с которыми будет взаимодействовие, и их связи, что было успешно представлено в виде ER-диаграммы.

SQLite3 был выбран в качестве системы управления базами данных, так как он надежен, быстр, удовлетворяет требованиям необходимым для реализации крипто-портфолио, что особенно удобно, он встроен в Django. Использование Django позволило упростить работу с базой данных благодаря его ORM, позволяющей оперировать высокоуровневыми моделями вместо прямых SQL-запросов.

В ходе проектирования были созданы диаграммы, отражающие основные процессы взаимодействия пользователя с ботом. Они помогают в визуализации ключевых сценариев использования бота и облегчают процесс разработки, предоставляя ясное представление о функциональности системы.

В заключение, вторая глава стала основой для дальнейшей реализации проекта, предоставив подробную и в то же время структурированную картину функционала криптовалютного портфеля.

# Реализация и тестирование

## Создание бота в Telegram

На этом этапе мы рассмотрим, как мы создали нашего бота в Telegram. Процесс был исключительно простым благодаря встроенной платформе Telegram для создания ботов, известной как BotFather. BotFather - это бот, разработанный командой Telegram для создания, управления и настройки пользовательских ботов. Мы следовали инструкциям, предоставленным BotFather, включая выбор имени для бота, создание уникального username, а затем получение уникального токена, который будет использован для идентификации нашего бота при общении с Telegram Bot API. Процесс создания бота представлен на рисунке 3.1.

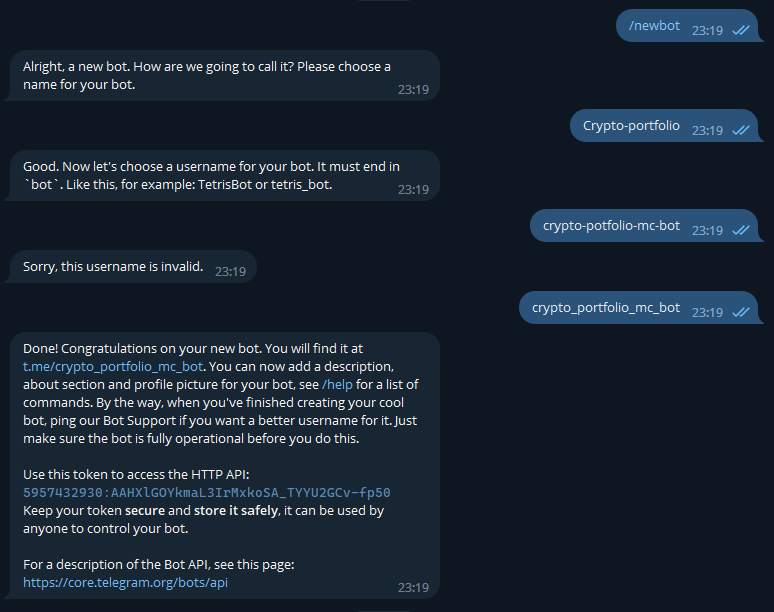


Рисунок 3.1 – Создание бота с помощью инструмента BotFather

## Установка дополнительных библиотек

На этапе реализации чат-бота стало необходимым использование дополнительных библиотек для обеспечения различных аспектов функциональности. Ключевыми библиотеками для нашего проекта были:

aiohttp: Эта асинхронная HTTP-библиотека используется для обеспечения параллельных HTTP-запросов к различным API, что позволяет обрабатывать взаимодействия пользователя с ботом быстрее и более эффективно.

aiogram: Это современная и мощная библиотека для создания ботов для Telegram на Python. Она предоставляет удобные и гибкие инструменты для работы с API Telegram, что позволяет создавать сложные сценарии взаимодействия с пользователем.

asgiref: Эта библиотека является частью ASGI (Asynchronous Server Gateway Interface), нового стандарта асинхронных веб-приложений Python, и используется для работы с асинхронными вызовами в Django.

requests: Эта библиотека используется для отправки HTTP-запросов. Мы использовали ее для взаимодействия с внешними сервисами для получения информации о криптовалютах.

## Описание констант и функций

В этом разделе описываются ключевые константы и функции, использованные в процессе разработки чат-бота.

### Константы

Основные константы представлены в моделях, определенных в файле models.py. Это классы, описывающие структуру базы данных приложения:

TelegramUser: Эта модель представляет уникального пользователя нашего бота. Он связан с моделью User Django через поле OneToOneField, что позволяет каждому пользователю Django иметь одного ассоциированного пользователя Telegram. Важное поле в этой модели - telegram\_id, которое хранит уникальный идентификатор пользователя Telegram.

Portfolio: Эта модель представляет портфель пользователя, который хранит его вложения в различные криптовалюты. Каждый портфель привязан к конкретному пользователю Django.

UserCoin: Эта модель представляет собой конкретное вложение пользователя в криптовалюту в его портфеле. Он хранит информацию о типе криптовалюты (coin\_id), текущей цене (price), цене покупки (purchase\_price), количестве (quantity) и времени создания (created\_at).

CoinHistory: Эта модель представляет историю изменения цены криптовалюты, вложенной пользователем. Она содержит информацию о вложении пользователя (user\_coin), дате и времени изменения цены (date) и самой цене (price).

### Функции

Команда "/start": эта команда выводит приветственное сообщение и информацию о доступных командах. Данная функция не имеет никаких сложных операций, просто возвращает текстовое сообщение. Выполнение функции представлено на рисунке 3.2.

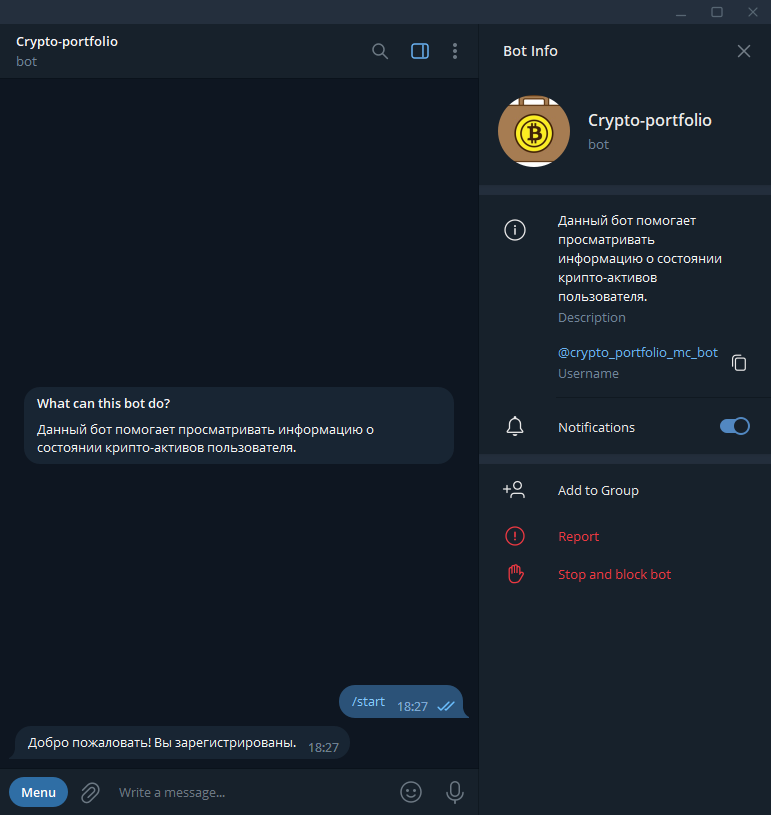


Рисунок 3.2 – команда start

Команда "/add": начинается с обработчика команды '/add', который анализирует введенную пользователем команду и сохраняет идентификатор монеты в состоянии. Затем переходит к следующему состоянию, где запрашивает количество монет для добавления. После введения количества монет, функция 'process\_quantity' обновляет портфель пользователя, добавляя указанное количество монет с текущей ценой. Команда представлена на рисунке 3.3.

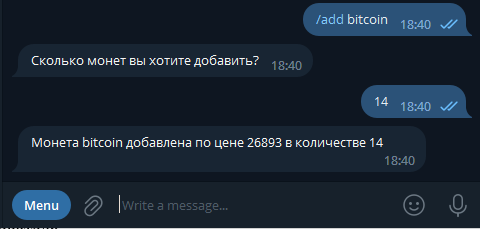


Рисунок 3.3 – команда add

Команда "/sell": похожа на команду '/add', но вместо добавления монет, она отвечает за их продажу. Это начинается с сохранения идентификатора монеты и перехода к следующему состоянию, где запрашивается количество монет для продажи. Затем функция 'process\_sell\_quantity' проверяет, достаточно ли у пользователя монет для продажи, и если да, то обновляет его портфель, вычитая указанное количество монет. Команда представлена на рисунке 3.4.

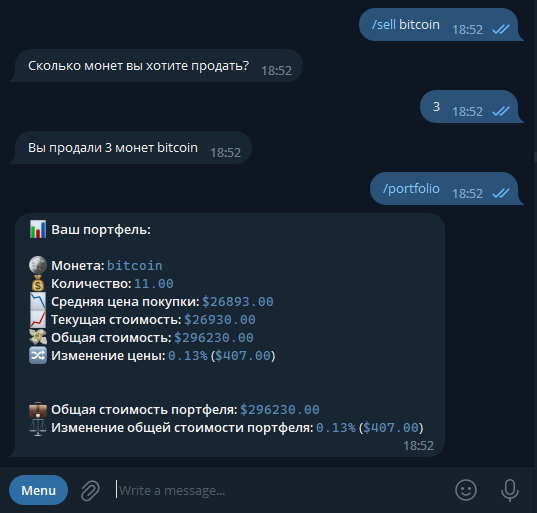


Рисунок 3.4 – команда sell

Команда "/portfolio": данная команда обрабатывает портфель пользователя и выводит информацию о каждой монете в портфеле, включая идентификатор монеты, количество, среднюю цену покупки, текущую стоимость, общую стоимость и процент изменения цены. В конце сообщения также выводится общая стоимость портфеля и изменение общей стоимости. Команда представлена на рисунке 3.5.

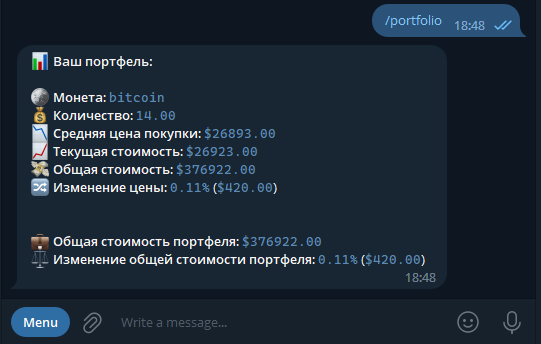


Рисунок 3.5 – команда portfolio

Команда "/clear": эта команда удаляет все монеты из портфеля пользователя и выводит сообщение о том, что все монеты были удалены. Команда представлена на рисунке 3.6.



Рисунок 3.6 – команда clear

Команда "/perc": эта команда позволяет пользователю установить порог изменения цены для конкретной криптовалюты. После ввода команды и указания порогового значения, пользователь будет получать уведомления, когда цена криптовалюты изменится на указанный процент. Эта функциональность полезна для отслеживания важных изменений в стоимости криптовалют и своевременного реагирования на них. Команда представлена на рисунке 3.7.

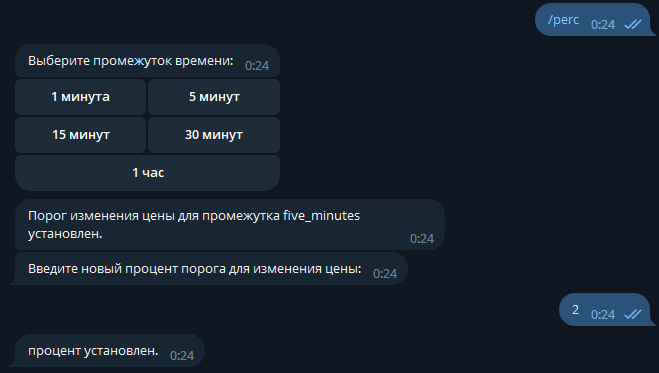


Рисунок 3.7 – команда perc

Команда "/vperc": команда позволяет просмотреть все установленные пороги изменения цены. При ее использовании пользователь получает список всех пороговых значений, на которые он подписался для получения уведомлений. Это позволяет легко управлять и контролировать все установленные пороги и, при необходимости, изменить их. Команда представлена на рисунке 3.8.



Рисунок 3.8 – команда vperc

Таким образом, все эти команды вместе обеспечивают полную функциональность бота, позволяя пользователям добавлять, продавать и просматривать монеты в их портфеле, очищать их портфель при необходимости, устанавливать пороги изменения цены а так же просматривать их.

## Тестирование бота

После реализации основного функционала бота, было проведено его тестирование. Это был важный этап, на котором было проверено, что все функции бота работают должным образом и что бот является стабильным и надежным. В процессе тестирования был проведен ряд единичных и интеграционных тестов, которые обеспечили проверку корректности работы бота.

## Выводы по главе

В данной главе были описаны все этапы реализации чат-бота для отслеживания криптовалют. Начиная с создания бота в Telegram были подробно описаны: каждый этап разработки, установки необходимых библиотек, настройки структуры проекта и описания основных модулей, реализации функционала бота и его тестирования. В результате был получен работоспособный бот, который отвечает всем требованиям и способен помочь пользователям в отслеживании их портфеля криптовалют.

Заключение

В результате выполнения дипломной работы был создан инструмент для мессенджера Telegram, предоставляющий пользователю ряд функций для управления криптовалютным портфелем. Этот инструмент позволяет выполнять следующие действия: добавление новых монет в портфель, продажа монет из портфеля, просмотр текущего состояния портфеля, полное его очищение, установка порога изменения цены для получения уведомлений и просмотр установленных порогов.

В ходе исследования были изучены ключевые аспекты функционирования мессенджера и его API, а также применяемые технологии и инструменты для разработки подобных инструментов.

Проект был выполнен на языке Python с использованием фреймворков aiogram и Django. Эти технологии предоставляют мощные инструменты для создания асинхронных приложений и управления базой данных. В рамках работы были подробно рассмотрены и применены различные аспекты работы с этими технологиями и инструментами.

Таким образом, результатом данной дипломной работы стал функциональный инструмент, способный помочь пользователям в управлении и отслеживании своих инвестиций в криптовалюты.

Основная цель проекта – разработка и реализация криптовалютного инструмента – была успешно достигнута. Однако, существуют возможности для дальнейшего развития и улучшения проекта, например, интеграция с другими криптовалютными биржами для предоставления более широкой картины рынка или добавление функционала для анализа трендов на основе исторических данных.

Список использованных источников

1. Результаты запросов по ключевому словам «crypto-portfolio» / GitHub : сайт. – URL: https://github.com/search?q=crypto-portfolio (дата обращения: 09.05.2023).
2. Репозиторий «trentpiercy/trace» / GitHub : сайт. – URL: https://github.com/ trentpiercy/trace (дата обращения: 09.05.2023).
3. Репозиторий «emmtte/Cryptocurrency-Portfolio» / GitHub : сайт. – URL: https://github.com/emmtte/Cryptocurrency-Portfolio (дата обращения: 09.05.2023).
4. Репозиторий «huwwp/cryptop» / GitHub : сайт. – URL: https://github.com/huwwp/cryptop (дата обращения: 09.05.2023).
5. Репозиторий «Toxich2012/kursovoi» / GitHub : сайт. – URL: https://github.com/Toxich2012/kursovoi (дата обращения: 15.05.2023).
6. Django documentation: The official documentation for Django. — Текст : электронный // Django : [сайт]. — URL: https://docs.djangoproject.com/ (дата обращения: 16.05.2023).
7. Coingecko API: A web service for retrieving cryptocurrency data. — Сайт. – URL: https://www.coingecko.com/en/api/documentation (дата обращения: 16.05.2023).
8. Telegram Bot API: Official documentation for the Telegram Bot API. — Текст : электронный // Telegram : [сайт]. — URL: https://core.telegram.org/bots/api (дата обращения: 16.05.2023).
9. Git / A free and open source distributed version control system : сайт. – URL: https://git-scm.com/ (дата обращения: 18.05.2023).